Integrating eGovernment Services using Semantic Web Technologies

Christian Drumm
SAP Research
SAP Research Center CEC Karlsruhe
cristian.drumm@sap.com

Abstract

In this paper, we will describe a prototypical application that shows how Semantic Web Services and current state-of-the-art Enterprise Application Integration software can be used to integrate eGovernment services across different service providers. We will describe the architecture as well as the challenges faced when integrating research prototypes with industrial-strength solutions.

Introduction

In many European countries, the government organizations have a very distributed structure. Different agencies that are organized at different levels (e.g. national vs. communal vs. local agencies) provide different services to the citizens. The operation of each of these agencies is supported by mostly proprietary legacy systems. In cases where no central population registration office is available (e.g. UK), each of these agencies even has to keep track of citizen records individually.

This setting poses severe difficulties on possible integration projects. For example in order to provide comprehensive eGovernment service to citizens, information and applications of different agencies need to be combined and integrated.

In this paper, we will describe a prototypical application that shows how Semantic Web Services [SWS] and current state-of-the-art Enterprise Application Integration [EAI] software can be combined in order to integrate government applications and information across different agencies enabling the creation of integrated eGovernment solutions.

The remainder of the paper is organized as follows. First, we will shortly present the use case that motivated the implementation of the system. After that, we will describe the prototype architecture followed by a discussion of the challenges faced when developing the prototype. Finally, we will summarize and provide an outlook on further work.

Use case overview

The prototype application, which will be called Change-of-Circumstance application in the following, was developed to solve a very specific use case at Essex County Council [ECC]. Whenever the circumstances in which a given citizen lives change, he might be eligible for a set of services and benefits provided by ECC and other governmental agencies together with public service providers. An example of such a change of circumstances is, if an elderly, partly disabled woman moves in together with her daughter. This changes the circumstances of both, the mother and the daughter. The mother might for example no longer receive a “Meals-on-Wheel” service whereas the daughter might get financial support for caring for her mother.

Even very simple processes like this require the interaction of many different government agencies. Each of the involved agencies has different legacy systems in place to keep track of citizen records, provided services, third-party service providers, etc. In ECC, this legacy system is called SWIFT. The system of one of Essex’s partner agencies is called ELMS. In order to integrate these different systems and to provide an integrated solution for providing services to the citizens in Essex, the Change-of-Circumstance application was developed. In the following section we will describe the architecture of the prototype.

Architecture

Figure 1 shows the high level overview of the prototype architecture. The prototype consists of four layers, namely i) the legacy system layer, ii) the service abstraction layer, iii) the Semantic Web Service layer and iv) the presentation layer.

The legacy system layer contains the legacy applications available in each of the agencies involved in the integration project. In the current version of the prototype, this layer contains two databases, the database of Essex County Council and the database of the Chelmsford housing department. Due to privacy and data security issues, we were not able to use the real databases running at the two agencies for the prototype. Therefore, we developed test databases containing dummy data. These databases have a similar design as

1A county located in the east of London in the UK
2Proprietary data base used by ECC and other local authorities to manage client case records e.g. in social services
3Proprietary data base used by Chelmsford District Council to manage client case records and provide housing services
the real databases. Furthermore, we created dummy data that mimic the real data available in the systems by including duplicate, inconsistent and conflicting records. As a result we were confronted with the same challenges when developing the application based on the test databases as if we had used the real productive databases.

In the following, we will explain each of the other three layers in more detail, as they show how different legacy systems can be integrated using SWS.

The Service Abstraction Layer

The lowest level of the prototype consists of the service abstraction layer. This layer is responsible for providing low level functionality available in the involved legacy systems to the SWS layer as well as for abstracting from the given implementation details of these systems. The SAP Exchange Infrastructure [SAP XI] (SAP 2005), is used to enable this abstraction. SAP XI is an Enterprise Application Integration (EAI) software. It is capable of integrating heterogeneous applications by acting as a middleware for the message exchange. In our prototype, we used the so called Adapter Framework to connect to the different existing databases (i.e. SWIFT DB and ELMS DB) and provide standard CRUD\(^4\) functions as web services to the next layer. Examples of Web services provided in this layer are:

- Create citizen record
- Update citizen record
- Delete citizen record

Note that for each of the two involved systems (i.e. SWIFT and ELMS), a different set of services is developed depending on the information stored in these systems.

The Semantic Web Service Layer

The Semantic Web Service layer consists of two main parts, namely i) a set of SWSs and ii) a set of goal templates. Both the goal templates and the SWSs are backed by an eGovernment domain ontology (Gutierrez, Domingue, & Cabral 2004).\(^5\)

Simply speaking, the Semantic Web Service layer contains two different sets of SWSs. Firstly, there are basic SWSs that fulfill very basic goals like e.g. “create a citizen record in the SWIFT system” or “order new equipment via the ELMS system”. These SWSs are directly mapped to the underlying CRUD services of the service abstraction layer. The purpose of these SWSs is to make the underlying Web service available in the Semantic Web Service layer for goal based invocation. Secondly, there are complex SWSs that fulfill more complex goals like e.g. “provide suitable equipment for citizen.” To fulfill these complex goals, several of the simple SWS need to be executed. Note that the execution sequence of the different basic SWSs is not hard-coded into the complex SWSs but is dynamically created using goal-based discovery and invocation.

The goal templates described earlier provide the ability to invoke the SWSs available in the Semantic Web Service layer from the User Interface layer. The goal templates are filled with the data entered by the user through the user interface and sent to the Semantic Web Service layer where they trigger the execution of different SWSs which in turn after several steps trigger the execution of web services in the Service abstraction layer.

Technically the SWS were developed in and are executed by the IRS-III server (Open University 2004). The IRS-III server is an SWSs runtime environment developed by the Open University.

The User Interface Layer

The user interface is a web application available using a standard Web browser. It uses the standard Java API provided by the IRS-III server to communicate with the Semantic Web Service layer of the prototype. Figure 2 shows a screenshot of the user interface. The screen mainly consists of two parts: The navigation bar on the left hand side of the screen and the main screen.

The navigation bar on the left is used to select a goal the user wants to achieve (like e.g. create a new citizen record). Based on the selection, different input screens are shown in the main screen area. After the user has entered the required data, he triggers the execution of the goal. Depending on the goal, different goal templates are filled with the data entered and are sent to the Semantic Web Service layer.

Technically, the User Interface layer is based on SAP Web DynPro which provides a comprehensive environment for the model-driven design of user interfaces.

Prototype Setup

After describing the design of the Change-of-Circumstance application we now describe the setup we chose for developing the prototype.

As depicted in Figure 3, we chose a highly distributed setup for the prototype in order to show the feasibility of

\(^{4}\)Create, Read, Update, Delete

\(^{5}\)The eGovernment ontology is available online at http://villapark.open.ac.uk/dip/egovolution/
operating a SWS-based application across several physically distributed locations. The two databases are running in a database server at the SAP Research Center in Karlsruhe, Germany. They are accessed by the Service Abstraction layer through the JDBC (Sun Microsystems 1994-2005) standard. The Service Abstraction layer runs also at the SAP Research Center in Karlsruhe. The Semantic Web Service layer, which is running inside the IRS-III server at the Open University in Milton Keynes, UK, connects to the Service Abstraction layer using SOAP over HTTP. The same protocol is used to connect the User Interface layer, which again is running in Karlsruhe, to the Semantic Web Service layer.

Challenges

The development of the Change-of-Circumstance application mainly posed challenges on two layers. The main challenge in the beginning of the project was the communication with the domain experts. It was necessary to understand their processes and to gather their knowledge of the government domain in order to enable the development of the eGovernment ontology and to develop the necessary SWSs. However, we are not focusing on these problems in the context of this paper. Instead we will discuss the technical problems encountered during the development of the prototype.

From a technical point of view, the presented generic architecture proved capable of developing SWS-based solutions on it. Especially the layering of the architecture proved very useful when developing the prototype application. The development of the ontologies, Semantic Web Services, goal descriptions and necessary mediators could be decoupled from the implementation of the user interface and the technical integration. Using a state-of-the-art EAI software for coping with integration problems on the technical level (e.g. providing Web services on top of different database systems) enabled us to focus on the development of semantic descriptions in the SWS layer. Furthermore, the advanced tools for wrapping legacy systems into Web services simplified this development process. In addition to the simplification of the development process, the usage of current EAI software also allows to achieve the security and data privacy constraints necessary in productive environments and eGovernment in particular. Security is one of the topics not yet solved in the context of SWS. But using SAP XI the access to the involved legacy systems can be controlled on a fine granularity.

From an implementation point of view, basing cross-organizational eGovernment applications on SWS technology seems to be a suitable approach. First of all, the usage of an ontology as a central hub format for the data available in the different legacy systems enables developers to focus on the functional aspects of the (semantic) services rather than on data interoperability problems. In addition, the ontologies simplified the communication between developers and domain experts as there was an agreed conceptualization of the domain available. However, there are still a number of challenges that need to be solved prior to running our prototypical application in a productive environment. Most important, neither of the currently available SWS infrastructures are available as an industrial strength infrastructure, but all of them are research prototypes that are still under heavy development. As a result of that, none of the existing SWS infrastructures is capable of coping with the tough requirements on uptime, performance, etc. usually necessary for productive applications. Therefore, the transition of the
currently available systems into a stable and robust infrastructure is the major challenge that need to be solved before a SWS-based solution can be deployed into a productive environment.

**Summary and Conclusions**

In this paper, we presented the architecture of the prototypical Change-of-Circumstance application. We described the different layers that make up the prototype and described its distributed setup.

The prototype demonstrates the possibility of integrating current SWS environments with industrial strength enterprise software. Also, there were quite a number of nitty-gritty details that needed to be solved in order to build the prototype. The integration was not as complex as expected in the end. Currently, we are in the state of finalizing the application in order to use it as a demonstrator for the integration capabilities of SWSs.

**References**


**Acknowledgments**

The work described in this document was partly funded by the European Commission under the project DIP. Furthermore, the work described in this paper was a joint effort conducted together with John Domingue and Liliana Cabral from the Open University and Leticia Gutierrez from Essex County Council. Last but not least, I want to thank Harald Fuchs for his efforts in setting up the database server.