

# Integrating eGovernment Services using Semantic Web Technologies

**Christian Drumm**

SAP Research  
SAP Research Center CEC Karlsruhe  
christian.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.

These settings pose 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]<sup>1</sup>. 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 a 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-Wheels" service whereas the daughter might get financial support for caring for her mother.

Even very simple processes like that 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*<sup>2</sup>. The system of one of Essex's partner agencies is called *ELMS*<sup>3</sup>. In order to integrate these different systems and to provide a 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

<sup>1</sup>A county located in the east of London in the UK

<sup>2</sup>Proprietary data base used by ECC and other local authorities to manage client case records e.g. in social services

<sup>3</sup>Proprietary data base used by Chelmsford District Council to manage client case records and provide housing services

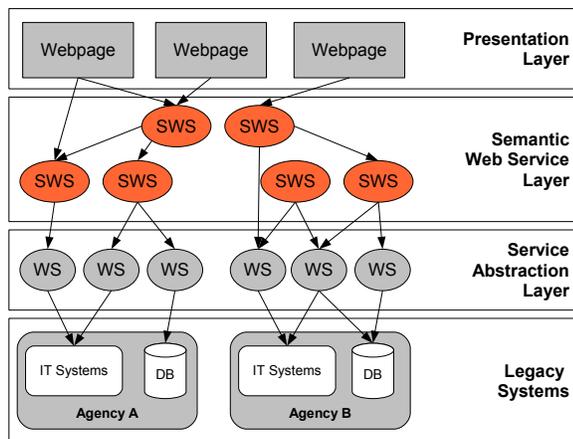


Figure 1: The high level architecture of the prototype.

the real databases. Furthermore we created dummy data that mimicked 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 layer 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<sup>4</sup> functions as web services to the next layer. Examples of web service provided in this layer are:

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

Note that for each of 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 Services layer consists of two main parts, namely i) a set of SWSs and ii) as set of goal templates.

Both the goal templates and the SWSs are backed by a eGovernment domain ontology (Gutierrez, Domingue, & Cabral 2004)<sup>5</sup>.

Simply speaking the Semantic Web Service layer contains two different sets of SWSs. Firstly there are *basic* SWSs that fulfill very basic goal 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-code 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 service 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 a the standard Java API provided by the IRS III server to communicate with the Semantic Web Service layer of the prototype. Figure 2 show a screen-shot 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 set-up

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

As depicted in figure 3 we chose a highly distributed set-up for the prototype in order to show the feasibility of

<sup>4</sup>Create, Read, Update, Delete

<sup>5</sup>The eGovernment ontology is available online at <http://villapark.open.ac.uk/dip/egovernment/>

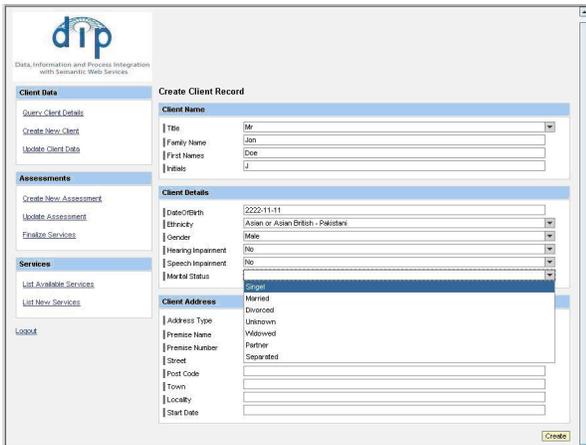


Figure 2: The graphical user interface of the Change-of-Circumstance application.

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.

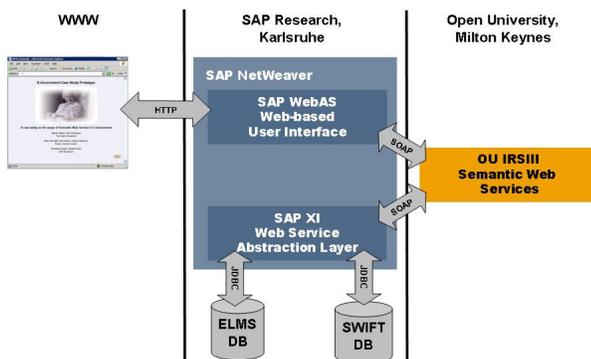


Figure 3: The distributed set-up of the prototype.

The rationale behind the presented architecture and the set-up of the prototype is twofold. First we wanted to show the feasibility of running a SWS based solution across several physically distributed locations, thereby enabling agencies to benefit from an integrated solution without the need to change their existing systems. The enabling factor for this feature are the integration capabilities of current state-of-the-art EAI applications. Secondly we wanted to demonstrate the possibility to non-intrusively integrate

SWS frameworks that are currently mainly in the state of research prototypes into current enterprise application software stacks using well known industry standards like e.g. SOAP. This integration enables access to the flexibility of SWS based application inside current solutions without any major changes to the underlying software stack or the programming model. As a result this integration shows a nice transition path from current technologies to SWSs.

## Challenges

The development of the Change-of-Circumstance application mainly posed challenges on two layers. The main challenges at 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 will not focus on these problems in the context of this paper. Instead we will discuss the technical problems encountered during the development of the prototype.

The main technical problems encountered where due to the break between Java world and the SWSs inside the IRS III server. IRS III uses unambiguous name to identify instances of ontology concepts. As long as the execution of goals is triggered from within IRS III<sup>6</sup> it is easy to use these names to specify the input for the goals. However initially there was no way to get at these names based on values of the properties of an instance. This means that it was not possible to find out all instances of the concept "citizen" with the first name "John" and the family name "Doe". Therefore IRS III had to be extended in order to allow the execution of the goal templates based on the data entered in the user interface.

On a lower level the re-engineering of the SWIFT and ELMS system where quite difficult. Even though we did not use the productive version of the databases that are way more complex, understanding the databases and the interdependencies between the data took quite some time and significant development effort.

## 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 the distributed set-up of it.

The prototype demonstrated 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 overall was not as complex as expected. 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

Gutierrez, L.; Domingue, J.; and Cabral, L. 2004. D9.3 - e-government ontology. Deliverable of the DIP project.

<sup>6</sup>e.g. by using the IRS III browser which can be downloaded at <http://kmi.open.ac.uk/projects/irs/>

Open University. 2004. IRS III - the internet reasoning service. <http://kmi.open.ac.uk/projects/irs/>.

SAP. 2005. Sap exchange infrastructure: The integration solution for process-centric collaboration. <http://www.sap.com/xi>.

Sun Microsystems. 1994-2005. Jdbc technology. <http://java.sun.com/products/jdbc/overview.html>.

### **Acknowledgments**

The work described in this document is 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 when setting up the database server.