Semantic Requirements for eGovernment Services Interoperability

Claude Moulin  
Compiègne University of Technology  
UMR CNRS 6599, Heudiasyc, France  
claude.moulin@utc.fr

Marco Sbodio  
Hewlett Packard – Italy  
Innovation Center  
C.so Trapani 16, 10139  
Torino, Italy  
marco.sbodio@hp.com

Fathia Bettahar  
Compiègne University of Technology  
UMR CNRS 6599, Heudiasyc, France  
fbettaha@utc.fr

Abstract

This paper considers the required knowledge elements taken into account for ensuring the interoperability between the different modules of the eGovernment platform that the TerreGov project has the objective to model and implement.

The glue of the system is a unique and multimodal ontology used both for describing the domain knowledge, adding semantics to agency services, indexing various documents from the civil servants’ knowledge base and building automatic dialogs between users and the system.

The ontology is also a support for the module in charge of managing complex tasks in the system. This module can dynamically request to find Web Services from semantic characteristics.

Introduction

Research issues have emerged from the rapid introduction of new technologies in government services, mainly Web Services making eGovernment a real application domain. It also appears that slightly modifying the current processes could not lead to the necessary modifications of the administration services.

The Terregov (see section 6 for more information) project focuses on the semantic requirements of governments at local, intermediate and regional levels for building flexible and interoperable tools to support the change towards eGovernment services. The strategic research area targeted by the TERREGOV Project is the “Usability and Applicability in large, complex and real-life environments for coherent development of interoperable government services”.

With a diversity of in-house information systems (information sources, databases, legacy applications) and the increasing availability of 3rd Party Web Services, the “Web Services” paradigm appears as a major brick for applications interoperability and integration. However, the implementation of complex and flexible government processes with Web Services still requires additional effort in terms of “eGovernment process support” facilities for both design and enactment.

In a growing pool of Web Services, technical interoperability is not sufficient and mechanisms should be set up for ensuring that semantics becomes the glue between applications. The enhancement of Web Services with semantics is a crucial step for making as easy as possible access to these services both by 3rd party applications and human users. Some efforts propose methods and methods for standardizing basic e-government services development at the national and international levels [12], [13], [14].

This paper tries to highlight some issues occurring at different levels, in the building of a platform and proposes some directions for bringing some answers.

Use Case

As an introduction to the global semantic issue of such platforms, we will present a
typical use case from which we based our research. This use case is intended to represent actual situations in a close future. Civil servants in local agencies are the closest contacts with the public. They are not only specialized in one type of problems but can also answer to various questions, from simple administrative requests, like address modification, to more complex problems like allowance demands in relation with health care services.

The system interface allows the civil servant to insert free text like the following requests:

a) What is the last law texts concerning specific allowances?

b) What is the state of the allowance dossier of Mr. Smith?

The civil servant is expecting answers appropriate to the questions:

a) The system returns URLs of documents in the knowledge base, associated to the question. This base can be automatically and manually updated.

b) The system discovers the Web Service managing the allowance dossiers. Mr. Smith’s data are found and the civil servant can explain the state of the dossier to the citizen.

From the questions, we can focus on at least three main entries: law texts, allowance, dossier. These words and expressions are item of an index. This index references concepts and relations of a global ontology describing the knowledge of the system. The same ontology is used for several purposes: i) indexing documents in the system knowledge base, ii) describing the semantic part of the web services. The definitions of concepts and relations in the ontology can also directly serve to answer some questions.

The ontology is also used to build dialogs between the system and the civil servant when some questions are not precise enough and need complements.

These answers illustrate some modules of the system architecture:

- The knowledge base allows to automatically index different types of documents, such as legal texts, messages, and agency internal documents.
- The semantic registry stores references to the published Web Services, along with their semantic description. It allows the research of services according to semantic criteria.
- Some procedures are more complex than a simple call to a Web Service. In this case, the “eProcedure” module (a workflow engine) is activated, which can run processes containing dynamic calls to Web Services from semantic requests. These processes implement a composition of semantic Web Services.

### Ontology Description

The ontology is intended to represent knowledge of the social care domain, which is the application domain of the TerreGov project. In order to present the ontology itself, we have created nine main categories of concepts, even though some of them are difficult to classify. It is not necessary in this section to use any strict formalism to present the ontological elements.

#### Concepts Categories

The main categories of concepts are the following. Some concepts are given, but obviously, this list is not exhaustive.

- Geographical areas, Territories and Administrative zones
- Authorities (elected bodies and not elected bodies)
- Social services (employment, health care, etc.)
- Documents
- Facilities, buildings
- Organizations
- People
- Time
- Information technology

#### Some Relations

Relations are divided in object and data type properties, according to the OWL
denomination. We give some examples using the triples syntax: Subject, property, Object “Municipality” “mayor” “Person” “Elected Body” “jurisdiction” “Territory” “Program” “funding agency” “Organization” “Financial Assistance” “assistance domain”

{“Housing” U “Health” U “Inclusion”}

Figure 2 shows an excerpt of the ontology.

Access to the Ontology

As described previously, concepts, relations and individuals have labels in each language. These labels are considered index items, referring to the corresponding elements of the ontology. Thus, it is possible to automatically index documents from the ontology concepts in each application country.

The formalism chosen for transferring and implementing the ontology is OWL [15], even if some ontological characteristics cannot be represented with this language’s syntax. Many tools are now available which can import OWL files, and some libraries of classes [18] allow to parse OWL files, thus, it becomes natural to use this formalism. However, we have developed our own ontological simplified language which allows us to produce OWL files containing some interesting features. One of these features is the automatic creation of inverse relations. We extend OWL descriptions with SWRL...
Rule-Based Knowledge

Knowledge in the system was first described by a classic ontology, i.e. with concepts, relations and individuals. But some knowledge elements were impossible to represent with this limited model. It was necessary to use rules for representing some elements. These rules are used to define virtual concepts, i.e. concepts for which it is not useful to search all the instances. It is only necessary, when needed, to verify if an individual is belonging to a specific virtual concept.

Rules are useful for defining some virtual concepts of an ontology:
- based on the composition of two relations
- using relation and Boolean operators

Rules can also be used to model preconditions.

Semantic Service Description

Web Services are the key bricks of the systems. However, as many researches tend to demonstrate, this technology is not sufficient for a complete integration of legacy systems. Several efforts of the W3C [27] in the domain of Web Services [26] tend to propose standards for formally describe Web Services capabilities: OWL-S [16], formerly DAML-S, and WSMO [17] (see [6] [7] [8] for WSMO-based Web Services environments). We have decided to adopt OWL-S, even tough this standard is still evolving, because some useful technical resources, such as Java libraries, are already available and will be regularly updated.

OWL-S is a language based on RDF/S [28] used for describing Web Services. In OWL-S a Service presents a profile, is described by a model and supports a grounding. The profile is mainly used for service discovery. It is related with the knowledge domain of the Service. The model (or process) is used to define the inputs, outputs, preconditions and effects of a Service. We also used the model for Service discovery.

The grounding describes the implementation of a Service. It is linked to WSDL [29] Service descriptions. The following excerpt of a semantic description shows the links with the domain ontology. The concepts whose labels are HealthCard and HealthCardId are defined in this ontology.

```
<process:hasProcess>
  <process:AtomicProcess rdf:ID="Process">
    <process:hasOutput>
      <process:Output rdf:ID="out0">
        <process:parameterType rdf:resource="http://.owl#HealthCard"/>
        <rdfs:label>HealthCard</rdfs:label>
      </process:Output>
    </process:hasOutput>
    <process:hasInput>
      <process:Input rdf:ID="in0">
        <process:parameterType rdf:resource="http://.owl#HealthCardId"/>
        <rdfs:label>HealthCardId</rdfs:label>
      </process:Input>
    </process:hasInput>
  </process:AtomicProcess>
</process:hasProcess>
```

Semantic Service Registry

Common Web Services descriptions are currently published in UDDI [24] registries, but these contain insufficient data for our purpose [3]. Thus, it is necessary to build new registries, containing enough semantic information for Service discovery.

This new type of registries allows the publication of semantic Service descriptions [9], and provides some Service discovery mechanisms based on service features appearing in the descriptions.

We mainly use the data types of input and output of a service for requesting a specific service to the semantic registry. In some cases, the research algorithm allows extensions to super-classes or subclasses of the requested types.

Figure 3 shows the architecture of the semantic registry.
Building Service Description

Service descriptions written using OWL-S are very verbose. Writing by hand such files is a painful task so, several initiatives to simplify this work can be highlighted.

The conversion between WSDL and OWL-S using the WSDL2OWL-S converter [20] produces OWL-S files containing the process and the grounding parts of the semantic descriptions. However, this technique has the main disadvantage of referencing the concepts with identifiers created automatically. In this case the coherence with concept identifiers of a shared ontology is lost.

Automated Semantic Service Annotation with Machine learning [30] is a WSDL annotator with semi-automated machine learning assistance. ASSAM [10] is designed primarily for the task of annotating several similar Web Services with a common ontology. ASSAM can also be used to annotate a single Web Service and ensure that it is compatible with existing Services descriptions. Ontologies can be imported and used as training data for annotation. The program consists of two main parts: A WSDL annotator application and a data aggregation algorithm called OATS. Once the ontology is completed it can be exported in OWL-S format.

Following the philosophy of Protégé1 extensions, the OWL-S editor is created as a plug-in which provides an easy-to-use editor tool for creating, editing and managing OWL-S Service descriptions through a new tab, which is shown when opening the Protégé with OWL-S format files. It is a project developed by SRI International which can be freely accessed. Code for the OWL-S editor is distributed under the Mozilla Public License version 1.1.

At the time of this writing, the Protégé OWL-S Editor plug-in [22] is currently in planning stage. It requires Protégé version 3.1, OWL plugin [21] and the GraphViz [23] editor for graphically building composite processes.

This tool provides edit modes for the four main parts of an OWL-S description: service, profile, process and grounding. The relationships between the different parts are shown in an intuitive way in the GUI and can also be shown as a graph.

Link between Ontology and Description

The Profile section of an OWL-S description must use a service category. Using OWL-S, it is necessary to create in the reference ontology, some subclasses of the class ServiceCategory. The following excerpt shows in the reference ontology the declaration of the class HealthCareService as a subclass of the OWL-S class ServiceCategory:

```
<owl:Class rdf:ID="HealthCareService">
  <rdfs:subClassOf rdf:resource = "http://www.daml.org/services/owl-s/1.1/Profile.owl#ServiceCategory"/>
</owl:Class>
```

The relation serviceCategory allows to associate an instance of a service profile and an instance of a category, in the OWL-S profile section:

```
<profile:Profile rdf:ID="anagraficaProfile">
  <profile:serviceCategory rdf:resource="#healthCareIdentification"/>
</profile:Profile>
```

Service Composition

Complex tasks

---

1 protege site : http://protege.stanford.edu/
Enabling heterogeneous enterprise application integration requires the composition of several Web Services into a workflow, for implementing complex tasks. Current technologies address the Web Service composition issue through Web Services orchestration, the major current industry initiative being BPEL4WS [25].

BPEL4WS aims at specifying business process behavior based on Web Services; it distinguishes between executable processes (modeling actual behavior of a participant in a business interaction) and abstract processes (describing the technological interfaces of business processes). BPEL4WS focus on representing static compositions, where both the flow of the process and its building blocks (i.e. the Web Services) are known in advance.

In order to achieve the dynamic composition of Web Services, it is necessary to extend such traditional workflow engines, with modules allowing for the dynamic discovery of Web Services, based on semantic information describing them.

Accordingly, abstract and executable processes have to be updated in such a way that dynamic processes based on qualified patterns could be implemented at run-time. The algorithms these modules implement use the ability to discover available Web Services at run-time. Such orchestrated processes (named eProcedures, in TerreGov) can be combined together by several algorithms for computing complex tasks. The “Instantiator” component of the eProcedures module serves as a bridge towards BPEL execution [19]).

Related Works

There are several ongoing efforts done in the fields of Web Services dynamic discovery and composition. Our approach is similar to the one described in [1], but we have augmented the capabilities of standard BPEL4WS execution engine, with automated reasoning on OWL/S descriptions. Both approaches try to build Web Services sequences that consume a set of (formally described) inputs, and produce a set of expected (formally described) outputs.

Mandell and McIlraith proposed a “Semantic Discovery Service” (SDS) which sits between the BPEL4WS engine and the service partners. This SDS accomplishes both dynamic discovery and dynamic composition, and act as a proxy between the discovered partners and the BPEL4WS engine.

In our approach we separate tasks: the semantic UDDI registry is concerned with dynamic discovery; the eProcedure module is concerned with the dynamic composition; the BPEL4WS engine is concerned with the execution (the “Instantiator” component of the eProcedure module serving as a bridge towards BPEL execution).

Another approach to Web Services composition is described in [4]. A semi-automatic approach at dynamic composition is proposed: an operator is assisted in the composition of a Web Services sequences by a program, which presents at each step of the composition the possible matching services. In our case, the composition must be dynamic and implemented at run-time.

There are also several examples of research and work done in the field of matchmaking algorithms for dynamic discovery. We just mentioned some. A prominent example is shown in [3] [2]: the algorithm tests if the request can provide all required inputs of a service and if the offered output satisfies the requester's demands.

The test can have several degrees of accuracy (for example exact, or subclass/superclass matching). A similar work has also been done within the matcher for the LARKS developed by Sycara et al. [11]. Finally, some interesting insight on the matchmaking problem is given in [5], where the experience report shows a development of a semantic web-based matchmaking prototype.

Acknowledgements

The TerreGov project is an integrated project cofunded by the European Commission

---

2 The content of this paper is the sole responsibility of the authors and in no way represents the views of the European Commission or its services.
under the IST (Information Society Technologies) Programme, eGovernment unit, under the reference IST-2002-507749. The TerreGov Consortium includes: AIRIAL Conseil (F), Université de Technologie de Compiègne (F), GFI Informatica (E), Hewlett Packard Italiana (I), Ratio Consulta (I), Oxford Computer Consultants (UK), Technion – Israel Institute of Technology (IL), HEC School of Management (F), Associazione Impresa Politecnico (I), Business Flow Consulting (F), European Regional Information Society Association (B), Stowarzyszenie Miasta w Internecie (PL), Aquitaine Europe Communication (F), Regione del Veneto (I), R&S Info (I), Epektasis (G).

Conclusion

The objective of this paper was to present the minimal semantic requirements necessary to conceive an integrated platform for eGovernment interoperable services. The solution adopted here starts from the building of a single multilingual ontology having several goals and in particular the indexing of documents in a knowledge base and the indexing of services in semantic registries. Some issues are still remaining. Building OWL-S service descriptions from WSDL files may lead to incoherences with the ontology. A solution could be to insert a semantic adaptor service between actual Web Services and the semantic registry.

References

[22] project home page: http://owlseditor.semwebcentral.org/


[29] Christensen E., Curbera F., Meredith G., Weerawarana S. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl