INTRODUCTION

The SemanticGov project financed by the European Union (EU) aims to enable pan-European E-government based on semantic web services (SWS). Semantic technology is used either for the description and the implementation of semantic web services. The extended description based on ontologies enables a semantics-based discovery of services, whereas other ontologies model the runtime implementation reflecting usually member-state specific laws and terminology. Mapping between ontologies from different member states of the EU helps to overcome the semantic inconsistencies.

In this article, we will briefly sketch the architecture adopted for the whole project and then focus on the way how CentraSite has been used for a distributed semantic registry and repository. For more details on the SemanticGov project please refer to the SemanticGov web site at http://www.semantic-gov.org.

1.1 Basic Architecture of SemanticGov

The SemanticGov project uses the Web Service Modeling Ontology (WSMO; see http://www.wsmo.org/). It provides the conceptual model for the description of Semantic Web Services. WSMO uses the Web Service Modeling Language (WSML; see http://www.wsmo.org/wsml/) for defining e.g. ontologies, semantic web services and goals. The
latter are used to describe the intended result of the execution of a semantic web service yet to be discovered.

There are two types of semantic web services:

- Basic **component** services are implemented as a semantic wrapper around existing conventional web services. These backend services including the WSDL describing their interface provide the so-called **grounding** of the SWS.
- A **composite** or **orchestrated** SWS is composed of the invocation of multiple component services (which may be basic or composite themselves) with optional intermediate **mediation** steps. The orchestration is described by a state automaton.

Within the SemanticGov project, additional **support ontologies** have been introduced for the implementation of **public administration** (PA) services. They provide standardized aspects for the description of an SWS, e.g. the provider, the domain and the type of service.

Mediation supports a very important feature of the SemanticGov architecture, i.e. the capability to create **pan-European** SWS spanning different member states using different terminologies. In a semantic system, these terminologies are represented by different ontologies. Mediation is used to translate between the ontologies of different member states or domains. The SemanticGov projects implements **mediation via mapping files** (see http://www.scharffe.fr/pub/aLanguageToSpecifyMappingsBetweenOntologies.pdf).

The technical infrastructure provided by the SemanticGov project consists of the following major components:

- The **Design Environment** is used to create all types of required artifacts – it is based on Eclipse 3.3. The artifacts are deployed to a distributed repository based on CentraSite 3.1.8.

- The **Runtime Environment** consists of a portal (based on Tomcat and Liferay) and a backend Semantic Execution Environment (SEE) named WSMX (see http://www.wsmx.org/). It enables access to the distributed repository for storage and retrieval of all artifacts.

- The **Distributed Repository** is used by both, the design and runtime environment. The distribution is based on the Java implementation of the JXTA peer-to-peer protocol (see https://jxta.dev.java.net/). Artifacts are not only stored in CentraSite’s repository but also parsed and registered including their dependencies in the registry when being deployed from the design environment. Hence, the interdependencies of the artifacts can be visualized using CentraSite’s graphical impact analysis either inside Eclipse or inside CentraSite Control.

The following figure shows the overall picture of the SemanticGov architecture:
Below, we will give a short description of the major components of the Design and Runtime Environment. In the second part of this article we will illustrate how CentraSite 3.1 has been used to support the design and execution of pan-European semantic web services for public administration.

1.2 The City of Turin Showcase

Within the SemanticGov project different showcases have been implemented. One of these user scenarios is the so-called City of Turin (CoT) showcase. It deals with a Belgium citizen who wants to change his residence to Turin. The respective \texttt{TurinBelgiumChangeResidenceService} is implemented by the orchestration of the following steps:

- call \texttt{BelgianRegistryGetService} to verify name and identity card data of Belgian citizen by checking the Belgian population registry,
- invoke \texttt{TurinPoliceVerificationService} to simulate verification of the new residence address in Turin by a policemen inspecting that location physically,
- invoke \texttt{BelgianRegistrySetService} in order to update the Belgian registry with the new residence address,
- finally update the Turin registry by calling the \texttt{TurinRegistryService}. It adds the Belgian citizen with his new residence address in Turin to the Turin population registry.

Hence, the \texttt{TurinBelgiumChangeResidenceService} is a real pan-European service as it spans different member states. Of course, each member state has its own laws and terminology, and thus
its own ontologies. A common base ontology (GEA – see e.g. [http://www.semantic-gov.org/index.php?name=WSMO](http://www.semantic-gov.org/index.php?name=WSMO)) for E-government public administration services as well as other basic ontologies (e.g. PROTON) have been created or adopted by the SemanticGov project.

The CoT showcase will be reflected in several of the subsequent figures.

### 1.3 SemanticGov Design Environment

The design environment contains the following major plug-ins for Eclipse (see also Figure 1):

- **WSMOSStudio** provides a set of editors and views for ontologies and semantic web services including a repository browser which supports also deployment of artifacts to a WSMO repository via drag and drop.

- The **PA Service Editor** supports creation of PA services including the *support ontology*. Both, the semantic web service and the ontology can be published to the distributed repository.

- The **Mapping Editor** obtained from the WSMT project supports easy creation of mapping files for data mapping.

- The **Semantic Web Services Composition Engine (SWSCE)** is an Eclipse plug-in allowing to generate an orchestration of selected web services based on the choreography of the composite SWS.

- **Goal-Tree Editor**: A goal tree is a special type of ontology describing the dialog of the citizen interacting with the SemanticGov portal. Traversal of the goal tree is used to determine the PA service to be invoked. The goal tree editor supports creation of a goal tree and its deployment to the portal.

- The **SemanticGov WSMX Repository** plug-in provides access to the distributed registry-repository based on multiple instances of CentraSite. It is an extended implementation of a WSMO repository supporting in addition the deployment of mapping files to the underlying WSMX resource manager.

- The Eclipse plug-ins delivered with **CentraSite** can be used to browse the artifacts stored in the CentraSite repository (and registry) nodes. Also, these plug-ins allow designing BIRT reports visualizing important information about artifacts stored in a CentraSite registry-repository.
1.4 Semantic Execution Environment

The semantic execution environment based on WSMX provides a prototype-level environment for execution of simple as well as orchestrated semantic web services. It has its own component architecture based on its own class loading vaguely similar to OSGI or a servlet container. Major components are (see Figure 1):

- The **Kernel** is responsible for loading and activation of components as well as the communication between components.
- The **Communication Manager** enables external communication via the following channels:
  - (incoming) Methods for access to WSMX components like discovery, resource manager, etc
    - Browser-based access via the WSMX Management Console
    - Web service interface
  - (outgoing) Invocation of backend services used for grounding
- The **Choreography** component supports execution of simple semantic web services
- The **Orchestration** enables execution of orchestrated semantic web services.
- Whenever a semantic web service or an ontology needs to be loaded during the execution of a simple or a composite service, the **Resource Manager** (RM) will be called. It can also be called to store WSMO entities like semantic web services, ontologies, or mapping files.

There are different implementations for the Resource Manager available – the **Distributed**
**Resource Manager (DRM)** used as the distributed repository will be described in more detail below.

- Execution of an orchestrated web service may also include the need for **data mediation** to overcome semantic inconsistencies. This may happen e.g. when instances of an ontology from one member state need to be transformed to instances of another member state’s ontology. The mapping file for the respective source and target ontologies is also loaded via the **Resource Manager**.

### 2. The CentraSite-Based Distributed Registry-Repository

The distributed registry-repository is based on installations of CentraSite 3.1 on multiple nodes, for example the Semantic Gateway or nodes owned by member states and domains in there (e.g. **Community and Social Services**). The communication among the repository nodes is accomplished via the JXTA protocol, a peer-to-peer protocol with reference implementations in Java and C.

Figure 3 illustrates the architecture of the distributed registry-repository used in the SemanticGov project. It shows that there are two types of peer groups used for communication:

- The global pan-European peer group contains all super peers (one per member state) as well as the Semantic Gateway not belonging to any single member state.

- For each member state there is an internal peer group used for the communication between the super peer and all the domain-specific repository peers within that member state.

Artifacts due to the infrastructure or created during the design phase are deployed corresponding to the ownership as follows:

- Common base ontologies (e.g. GEA, PROTON, handling of SOAP faults) are not specific to any member state and thus are stored in all member state repositories

- All other WSMO entities (ontologies, semantic web services) which are owned by a member state are stored
  
  - in the CentraSite used by a domain-specific repository peer if the entity belongs to a specific domain, or
  
  - in all repository peers of that member state if the entity does not belong to a specific domain

- Mapping files are typically used for the cross-member state data mediation. They are stored on the Semantic Gateway node.

- Backend web services used for the grounding of semantic web services are registered in the repository peer belonging to the domain providing the backend web service.

At execution time of an orchestrated semantic web service, all referenced entities (component web services, ontologies) are loaded from the distributed repository.
2.1 Registry and Impact Analysis

All artifacts used within the SemanticGov infrastructure are either stored as serialized documents in the CentraSite’s WebDAV repository and registered in its JAXR registry. Taking advantage of the extensibility of CentraSite’s JAXR data model, corresponding user-defined object types are defined in CentraSite, i.e.

- WSMO Web Service
- WSMO Ontology
- WSMO Goal
- WSMO Mediator
- Mapping File
- Backend “WSDL” web service (predefined by CentraSite)

Whenever an artifact is stored in a repository peer or in the Semantic Gateway node, its dependencies on other artifacts are extracted and registered as associations between the instances of these object types. Figure 4 illustrates the important object types and their registered interdependencies obtained by parsing the WSML documents.
During the lifecycle of artifacts used throughout the infrastructure developers or system engineers need to know the dependencies on artifacts in order to identify the impact of changing it as part of the maintenance. However, due to the architecture of the distributed registry-repository, each member state or domain specific peer has only a local and thus incomplete view of all interdependencies. In order to overcome this limitation, another feature of CentraSite has been used in the SemanticGov project, i.e. Federation.

2.2 CentraSite Federation: The SemanticGov Mediator

CentraSite federation allows replicating registry data from any type of slave (source) registry into a CentraSite JAXR-based registry. Out of the box, CentraSite federation supports two types of slave registries, i.e. a CentraSite JAXR registry or a UDDI registry. Each type of source registry is supported via a so called Mediator component which can be registered with the Federation Servlet. The standard configuration of CentraSite provides two types of mediators, a CentraSite mediator and a UDDI mediator.

Figure 5 below illustrates the usage of CentraSite federation in the distributed SemanticGov registry-repository. The federation servlet periodically polls for new registry objects in the repository peers. New objects will then be transferred to the master registry residing on the Semantic Gateway node. A technical user (system engineer) can then connect to that central master registry in order to perform an impact analysis including all relevant artifacts from all member states.
In the SemanticGov scenario, the source registries are all CentraSite JAXR registries. However, the standard CentraSite mediator cannot be used as is for the following reasons:

- WSMO objects have their own identifier, so called IRIs (international resource identifiers), whereas in a JAXR registry the artifacts are identified by system-generated registry keys. References between mapping files and different types of WSMO entities are based on IRIs.
- The same WSMO entities and dependencies may be registered independent of each other in different member state and domain specific registries or in the Semantic Gateway registry. For example, a mapping file depending on its source and target ontology may be registered in the Semantic Gateway registry before the source or target ontologies including their dependencies have been propagated from the respective member-state registries to the Semantic Gateway registry. Hence, the dependency trees in the Semantic Gateway and the member-state registries must be merged. A mapping between IRIs and registry keys needs to be applied during the replication process. This is accomplished by the **SemanticGov Federation Mediator** which is used for replication of all types of WSMO entities. Replication of other types of objects ("WSDL" backend services; organizations) is achieved by delegation to the standard CentraSite mediator.

Using CentraSite 3.1, it is not straightforward to use an own mediator. Nevertheless, it is possible and thus allows fulfilling the special needs of SemanticGov. Support for custom mediators has been greatly enhanced within CentraSite 8.0.
The following screen snapshot illustrates the configuration of a federation site using the SemanticGov CentraSite Mediator:

![Figure 6: Creation of a federation to the Belgium CentraSite Registry using the SemanticGov Federation Mediator](image)

Once the federation site has been set up correctly, e.g. a Belgian system engineer could use the master CentraSite registry to find out about the usage of the Belgian population registry services by other pan-European semantic web services as illustrated in Figure 7 below.
In order to obtain this complete diagram, it is required to customize the configuration for the graphical impact analysis: application-specific association types like hasGrounding need to be activated. Then it is possible to find out e.g. which backend web services are used by an orchestrated semantic web service, even if it is spanning multiple member states.

2.3 Custom Reports

In the SemanticGov project, CentraSite’s feature of customizable reports has been used to provide two types of reports. The first report lists all WSMO objects created or changed within the last month (figure 8).
The second report gives a histogram with the number of instances being stored for each WSMO entity type (figure 9).

Figure 8: Custom report listing all WSMO objects created or changed recently

Figure 9: Custom report showing number of registered WSMO entities
3. Summary

Using CentraSite in the SemanticGov project has proven its great flexibility based on
- its extensible data model via user-defined object types and user-defined association types,
- its extensible federation mechanism taking advantage of a custom federation mediator,
  (Note that an improved implementation of federation as well as the newly documented
  feature of creating a custom federation mediator has been provided by CentraSite 8.0.)
- user-defined reports

Furthermore, different from the initial plans, CentraSite has also been used for storing artifacts in its repository in addition to their registration via JAXR.

For the implementation of the distributed registry-repository a license for a CentraSite 3.1 Enterprise Edition had been required in order to create user-defined object types. For setting up the showcases, the freely available Community Edition is sufficient.
## 4. Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoT</td>
<td>City of Turin – a user partner in the SemanticGov project</td>
</tr>
<tr>
<td>EU</td>
<td>European Union</td>
</tr>
<tr>
<td>Grounding</td>
<td></td>
</tr>
<tr>
<td>IRI</td>
<td>international resource identifier – used for WSMO entities as well as for mapping files</td>
</tr>
<tr>
<td>JAXR</td>
<td>Java API for XML registry – data model used for the CentraSite registry</td>
</tr>
<tr>
<td>JXTA</td>
<td>A peer to peer protocol – see <a href="https://jxta.dev.java.net/">https://jxta.dev.java.net/</a></td>
</tr>
<tr>
<td>Mediator</td>
<td>Please note that this article deals with two quite different types of mediators: (a) semantic mediators as introduced by WSML WSMO (b) federation mediators allowing to replicate registry objects from an slave (source) registry of an arbitrary type to a CentraSite JAXR master (target) registry</td>
</tr>
<tr>
<td>Member state</td>
<td>A member state of the EU</td>
</tr>
<tr>
<td>PA service</td>
<td>Public administration service</td>
</tr>
<tr>
<td>Registry</td>
<td>Please note that this article deals with two completely different types of registries: (a) CentraSite JAXR registries allowing to register arbitrary (registry) objects (b) population registries</td>
</tr>
<tr>
<td>SWS</td>
<td>Semantic Web Service</td>
</tr>
<tr>
<td>SWSCE</td>
<td>Semantic Web Service Composition Engine</td>
</tr>
<tr>
<td>WSML</td>
<td>Web Service Modeling Language, see <a href="http://www.wsmo.org/wsml/">http://www.wsmo.org/wsml/</a></td>
</tr>
<tr>
<td>WSMO</td>
<td>Web Service Modeling Ontology, see <a href="http://www.wsmo.org/">http://www.wsmo.org/</a></td>
</tr>
<tr>
<td>WSMT</td>
<td>Web Service Modeling Toolkit, see e.g. <a href="http://www.wsmo.org/TR/d9/d9.1/v0.1/20050127/">http://www.wsmo.org/TR/d9/d9.1/v0.1/20050127/</a></td>
</tr>
<tr>
<td>WSMX</td>
<td>Web Service Modeling eXecution environment, see <a href="http://www.wsmx.org/">http://www.wsmx.org/</a></td>
</tr>
</tbody>
</table>