Ontology Driven e-Government

Peter Salhofer, Bernd Stadlhofer and Gerald Tretter
University of Applied Sciences, Graz, Austria
peter.salhofer@fh-joanneum.at
bernd.stadlhofer@fh-joanneum.at
gerald.tretter@fh-joanneum.at

Abstract: This paper presents an approach to model ontologies for the e-Government domain as a basis for an integrated e-Government environment. Over the last couple of years the application of semantic methodologies and technologies in the e-Government domain has become an important field of research. A significant number of these approaches aim at automatic service discovery and service orchestration (Lu et al. 2004) (Crichton et al. 2007) by adding and utilizing semantic annotations to web services. In contrast to these approaches it was our idea to use semantic methodologies in a more forward-engineering manner – to create a semantic model first and to use this model e.g. for service selection but also as basis for the automatic generation of "intelligent" web forms. Thus the ontologies can be seen as a model that forms the basis of a Model Driven Architecture (Miller et al. 2001) approach to e-Government. That is why we call it Ontology Driven e-Government. The principle is rather straightforward. Every public service is semantically modeled and contains references to the required input elements. Any constraints on the service input element – also known as preconditions – can be expressed by semantic rules and evaluated by semantic reasoners. This allows for an automatic creation of (web) forms and interactive plausibility checks of data gathered from the user. Instead of scattering logic over numerous functions and procedures in all possible layers of an application, it is now consistently kept in the semantic model. Another key advantage of this approach is that the knowledge of public services becomes available in a machine processable form which allows for much more than just forms creation. Discovering the citizen's actual goal is one of these use-cases and is actually a very central and important step. When developing the idea of ontology driven e-Government it was one main idea to achieve a strong decoupling between the form solution and the backend. Such a decoupling can be achieved by transforming the input data into a common data interchange standard format, which was EDIAKT II (Freitter et al. 2006) – an XML Schema definition for the exchange of electronic documents between public authorities in Austria – in our case. Following this approach the input data can be consumed by any application supporting the data interchange standard EDIAKT II like the SOA-backend also proposed in this paper.

Keywords: e-government, ontology, WSML/WSMO, goal orientation, form generation

1. WSMO-PA – an e-Government meta-model

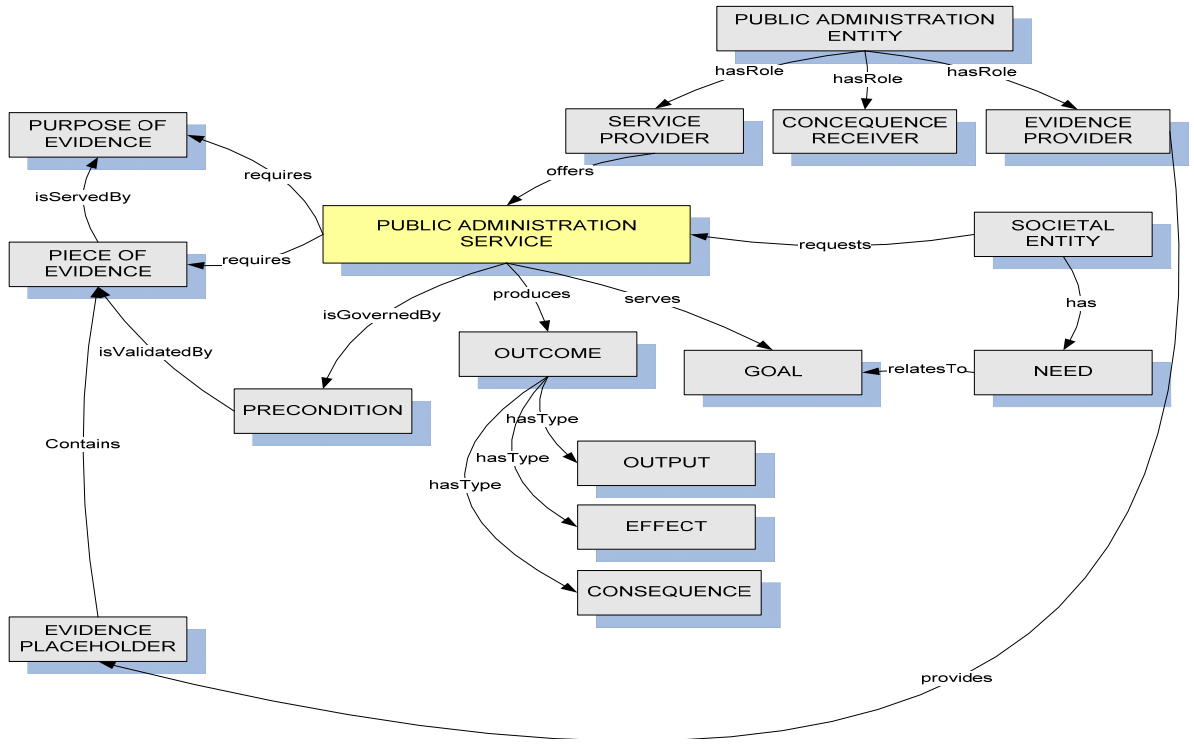

Figure 1: GEA-PA service model (Wang 2007)

To ensure that all public services are modeled in a consistent way, a meta-model that acts as a modeling guideline is needed. Therefore we chose a model proposed in the Government Enterprise Architecture – Public Administration (GEA-PA) respectively its WSMO implementation WSMO-PA (Wang 2007) as shown in Figure 1.GEA-PA is technology neutral. Thus it does not make any assumption about the semantic framework or the nature of the actual implementations of public services, hence most often they will be implemented as web services. It also models goals and needs to link citizens (societal entities) to public administration services. This allows for goal-oriented discovery of public services that do not necessarily have to be implemented as (semantic) web services.

2. Goal-oriented e-government

One big advantage of semantic web services is their inherent goal orientation. They contain a semantic description of what they do or achieve. Before a user can make use of one (or more) of these services, the following phases have to be passed through (Leitner 2003):

- Goal discovery phase: In this phase the actual goal of the user has to be correctly formulated using semantic notations.
- Semantic web service discovery phase: A set of semantic web services that might fulfill the goal is retrieved.
- Service selection phase: The web service that will actually be executed is selected from the set of retrieved services

Formulating the goal using any of the semantic methodologies can become relatively difficult. This is due to the fact that the problem domain itself can be relatively complex and, since this process involves user interaction, a simple and easy to use interface for expressing the goal is needed. In our prototype implementation of ontology driven e-Government we have limited the problem domain to the construction approval process. According to the construction law that has to be applied in this example there are three different categories any construction might fall into:

- Building development requiring official approval: In this case you have to apply for approval which will trigger a fairly complex process
- Notifiable building development: In this case you have to notify the responsible public agency providing detailed information about the project. The agency can prohibit the project within six weeks. Otherwise approval is granted
- Building development not requiring official approval: In this case you just have to inform the responsible public agency about when construction work will start and provide some basic information about the project.

Which of these services is needed for a given project depends on the type but also probably the size or extent of the structure. The correct answer to this question requires some in-depth knowledge of the construction law. To offer these services via e-Government to citizens you also have to provide some easy to use means of identifying the required service. This is done by semantic goal and service discovery.

3. Selecting the semantic modeling framework

There are currently several competing frameworks for modeling semantic web services submitted to the W3C. Among them is OWL-S (Martin et al. 2005) and WSMO (Fensel et al. 2007). We implemented evaluation prototypes based on both approaches and eventually chose WSMO which is based on WSML (Bruijn et al. 2006) over OWL-S which is layered on top of OWL (W3C 2004). While a detailed discussion of the differences between these two approaches can be found in (Bruijn et al. 2005) and (Polleres et al. 2004), here we will simply state the reasons for our decision.

One basic idea of our approach is its forward engineering nature. This means, that we want to start new e-Government projects by modeling the ontology first and then using this ontology as a domain model to form the basis of the generation of an application or service. In contrast to this, semantic technologies are usually used to add annotations to already existing services. In our context, the ontology can be compared to a platform independent model (PIM) as it is used in model driven architecture (Miller et al. 2001). Since the ontology is the key element, we had to look for a modeling approach that is simple and yet expressive and powerful enough to cover all possible aspects that might be needed for creating a runtime environment that is based on this model.

Table 1: A simple OWL example

```
<owl:Class rdf:ID="Person">
  <rdfs:label>Person</rdfs:label>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasParent">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasName">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string""/>
</owl:ObjectProperty>

<owl:Restriction>
   <owl:onProperty rdf:resource="#hasParent"/>
   <owl:maxCardinality rdf:datatype="xsd:nonNegativeInteger">2
   </owl:maxCardinality>
   <owl:minCardinality rdf:datatype="xsd:nonNegativeInteger">2
   </owl:minCardinality>
</owl:Restriction>
```

Table 2: A simple WSML example

```
concept Person
  hasName ofType (0 1) _string
  hasParent ofType (2 2) Person
```

While OWL is based on XML, WSML can be seen as a domain specific language based on Meta Object Facility (MOF) (OMG 2002). As a consequence there is no XML overhead in WSML. Table 1 and Table 2 represent the same facts. Every person has a name and parents. It is obvious that the WSML version is easier to read, which is definitely an advantage in this case since models can be easily created and reviewed by authors even without the use of tools.

Besides the expressive notation of WSML there exists a language variant called WSML-Flight, which supports logic programming based on F-Logic (Kifer et al. 1995). In contrast to OWL, this approach favors the closed world assumption, which makes the formulation of logical constraints much easier. As you can see, another minor difference is the terminology. OWL uses the term class whereas WSMO uses term concept for the abstraction of a thing. Both terms can be used synonymously.

4. Modeling the ontology

As mentioned above, the first services that should be supported by this new approach are construction approval services. These services are governed by a local construction law. Thus this law is an important source for modeling the required ontology since it contains all needed concepts and the logical rules and requirements that form the basis of the public agency's actions. Even though some attempts to automatically extract semantic information from laws (Biagioli et al. 2005) (Schweighofer et al. 2001) already exist, we conducted this step manually, identifying needed concepts and their interdependencies by carefully analyzing the text.

Other important sources, especially for modeling the attributes of identified concepts, are existing (paper based) forms. Every input field on a form is considered to be a property of one of the concepts involved (e.g. “family name” is a property of the person concept acting as builder in this context).

While modeling the ontology for the construction approval domain, it became clear that it is relatively hard for non experts to even find out which one of the existing services (see Section 2) is the appropriate one or whether any service is needed at all. This is why we have introduced goals/desires to guide the user while selecting the appropriate service.

4.1 Goal templates

Goals are elements in all frameworks that are used to describe semantic web services. Since they are used to describe the capability of these services they typically exist at a relatively low and technical layer (Fensel et al. 2007). In the context of this paper we use the term goal as a synonym for the terms desire or need as they are suggested by GEA-PA (Wang et al. 2007). An abstract, yet typical goal in the given context would be:

"I want to {build|knock down|rebuild} a {structure}!"

This goal is a typical semantic triple consisting of subject ("I", the citizen), predicate ("build" or "knock down" or "rebuild") and object ("structure"). Our top-level goals reflect what citizens might want to do or want to achieve (e.g "to build something"). This should make it easy to identify appropriate goals that fit the needs of a particular life-situation. At this top-level, a goal typically cannot be uniquely mapped to one single service. Therefore every goal has to be refined so that it becomes a concrete one like

"I want to build a garage!"

However, this still does not unambiguously identify the required service since in this example the required type of service depends on the type and number of motor vehicles that will be parked in the garage. Thus, we would need a more specific goal like

"I want to build a garage for three cars!"

Which type of approval or service is actually needed is clearly defined in the underlying regulations. Assuming that the applicable law is consistent – which is typically the case – this approach does not lead to any goal conflicts since every possible case falls into exactly one category and is therefore assigned to one particular service.

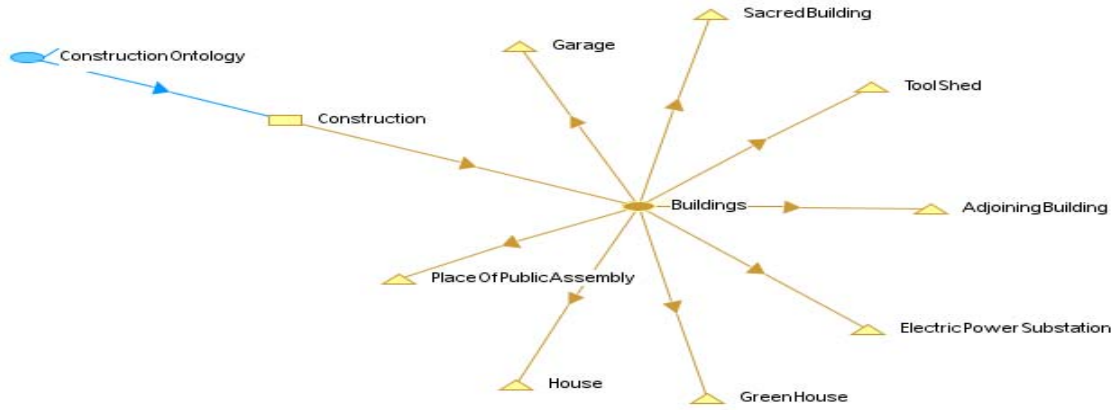

Figure 2: Part of the concept hierarchy

After analyzing the construction law, a model containing all concepts/classes was created. Since the construction law sometimes referred to more abstract concepts (e.g. "building") and sometimes to more specific ones (e.g. "detached family house"), the resulting concepts formed a hierarchy (see part of hierarchy in Figure 2). The top-level concept that could be used in the goal template is "construction". All other identified concepts representing more specific types of a construction are modeled as sub-concepts resulting in a tree of concepts.

The basic idea of the goal discovery process was to start with a goal template containing the most abstract concept (e.g. "construction" or "structure") and assist the user in refining these concepts by specialization until the administrative service that is needed to fulfill this goal can be unambiguously identified. The goal discovery algorithm is explained in Section 5.

4.2 Specialization and classification

An ontology is defined as (Ehrig et al. 2004):

A similar definition of an ontology not including datatypes can be found in (Bloehdorn et al. 2005). The is_a relationship between two classes is defined as follows:

$$c_1 \leq_C c_2 \mid c_1, c_2 \in C \Rightarrow c_1 \text{ is subconcept of } c_2$$

The is_a relationship is transitive:

$$c_1 \leq_C c_2 \wedge c_2 \leq_C c_3 \Rightarrow c_1 \leq_C c_3$$

Defining a function attr(c)

$$attr(c): C \rightarrow A$$

that returns the set of attributes belonging to a given concept c the following statements also hold true:

$$c_1 \leq_C c_2 \mid c_1, c_2 \in C \Rightarrow attr(c_2) \subseteq attr(c_1)$$

This means that all sub-concepts contain all the attributes of all their super-concepts but might also posses additional attributes. Since a concept might also have several direct super-concepts, the following statement has to be true:

$$c_1 \leq_C c_2 \wedge c_1 \leq_C c_3 \mid c_1, c_2, c_3 \in C, \\ c_2 \neq c_3 \wedge \neg(c_2 \leq c_3 \vee c_3 \leq c_2) \\ \Rightarrow (attr(c_2) \cup attr(c_3)) \subseteq attr(c_1)$$

Sub-concepts are also called specializations of their super-concepts since they are more specific. In our ontology model, however, we use two different types of sub-concepts. One is classical specialization. For example garage is a sub-concept of building. The second type of a sub-concept is no real specialization but more a kind of classification. As mentioned above, which type of administrative service needs to be used when building a garage depends on the size of the garage, which in turn is defined by law referring to the type and number of vehicles that will be parked there. Therefore we have defined three sub-concepts called small-, medium- and big-garage. These sub-concepts represent different classes of garages rather than more specific types of garages. Consequently one characteristic of classification compared to specialization is that attributes of sub- and super-concepts will be identical:

$$c_1 \leq_C c_2 \Rightarrow attr(c_2) \equiv attr(c_1) \\ \text{...in the case of classification}$$

The difference between sub- and super-concepts in the case of classification lies in the domains (i.e. values) of some of their attributes. In WSML these differences can be modeled by axioms which are used by the reasoner. Thus, having given any instance of the concept garage, the reasoner can infer the correct sub-concept it belongs to by analyzing number and type of vehicles parked there.

Table 3: Axiom defining a small garage

```
axiom SmallGarageDefinition1
definedBy
?x memberOf SmallGarage
impliedBy
?x[forVehicleType hasValue ?vehicleType, vehicleCapacity hasValue ?capacity] memberOf Garage
and (?vehicleType memberOf vehicle#Car
and ?capacity < 3
or
?vehicleType memberOf vehicle#Motorcycle
and ?capacity < 6).
```

The meaning of the axiom in Table 3 is almost self-explanatory. There is some x that pretends to be a Garage and therefore has to have some attributes. If the values of these attributes meet the constraints of the axiom than this x becomes a SmallGarage.

5. The goal discovery algorithm

Initially based on the GEA-PA service model, we have refined and adapted parts of this model as shown in Figure 3.

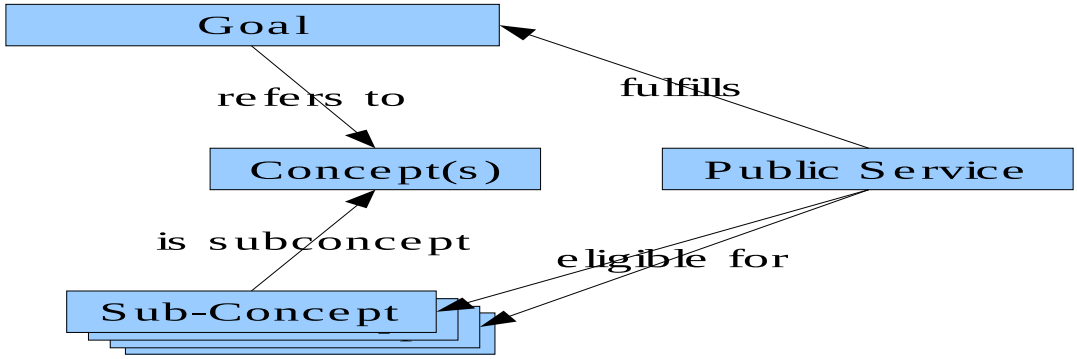

Figure 3: Refined goal service relationship

A goal might refer to one or more concepts (e.g. “I want to build a {construction}!”). Services are mapped to goals. There might be several services that fulfill a goal (e.g. in our construction approval example there are three different services), however, the combination of goal and a concrete (sub-) concept uniquely identifies the service needed. This assumption is based on the following constraints:

- Every concept that has sub-concepts is considered to be abstract.
- Every concept that has no sub-concepts (a leaf in the concept graph) is considered to be non-abstract.
- Public services only accept instances of concrete, non-abstract concepts.

This allows for a very simple algorithm:

- Start with a goal template
- For each concept the goal template refers to, go down the concept hierarchy till a leaf is reached.
- Lookup the matching service.

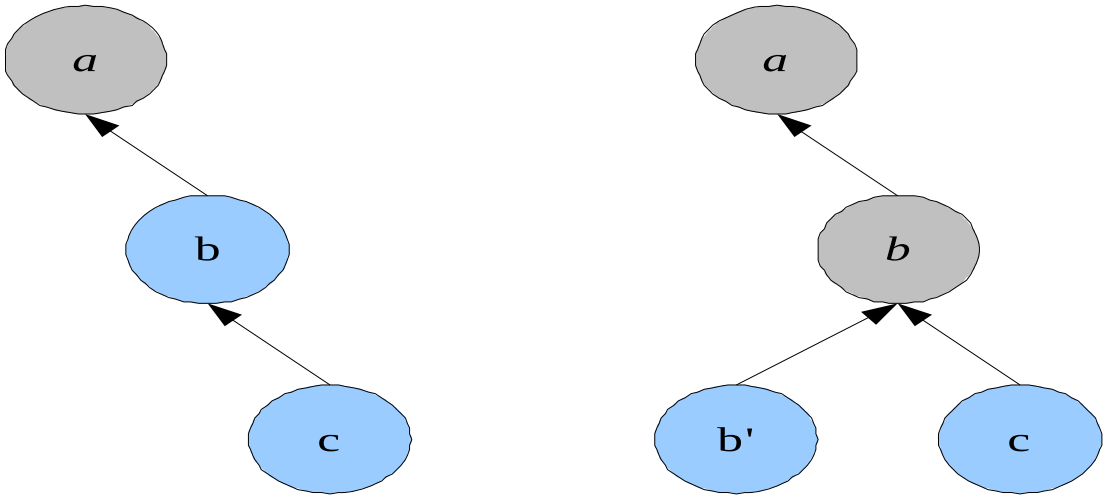

Figure 4: Non-abstract concept b is replaced by b', b is made abstract

As shown in Figure 4, these constraints do not impose any limits on our approach. Assuming that there is a concept b, that has sub-concepts and is not abstract (i.e. instances of this type are valid input to services), the concept hierarchy can by remodeled by declaring b abstract and introducing a non abstract concept b' with the following behavior:

$$attr(b') \equiv attr(b)$$

In this case an instance of the formerly non-abstract concept b is needed, an instance of concept b' is used.

The entire algorithm represents the following function:

$findService(g,c): G[xC]^n \rightarrow C$
where g is a goal and c is a possibly empty set of concepts;

The central part of this algorithm is traversing the tree of concepts. There are two different methods to identify the required sub-concept. In the case of real specialization, the user has to select the appropriate sub-concept from a list as shown in Figure 5. The question always follows the same pattern:

> *"You want to {build| knock down | ...} a {currentConcept}. Please further specify the type of {currentConcept}:"*

Possible answers are the sub-concepts of currentConcept. In use-cases where there are many different concepts organized in multiple hierarchies, this approach might be perceived as being a little onerous. Therefore, we are already considering about integrating some search facility as well.

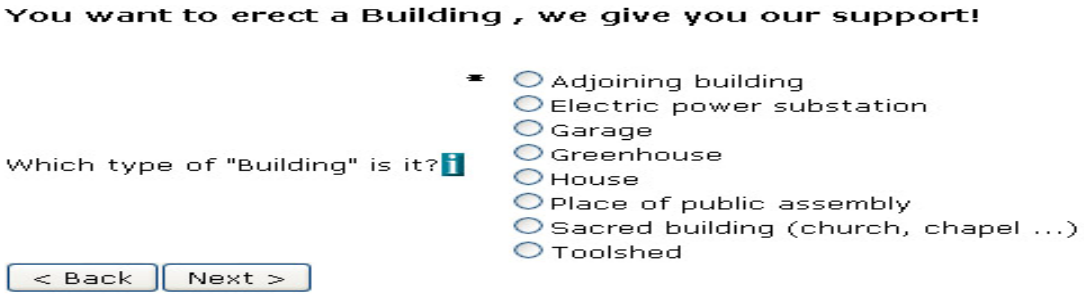

Figure 5: Prototypic user dialog for selecting a sub-concept

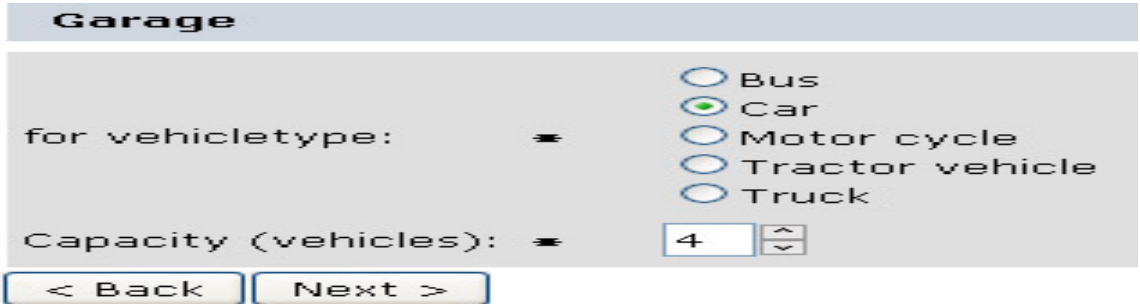

Figure 6: Prototypic user interface to specify attribute values that are needed for semantic reasoning

In the case of classification, the prototype system analyses the axioms and asks for the attributes that are evaluated to infer the correct sub-concept (see Figure 6). The sub-concept is identified by the semantic reasoner.

6. Form generation

After identifying the appropriate service for the citizens' goal the next step that has to be taken is to identify the relevant input for the selected service and to transform the semantic representation of the required information into a corresponding web form. In our prototype application the java library wsmo4j (details see http://wsmo4j.sourceforge.net) was used to access the created ontologies. We have used Java Server Faces (JSF) as our web technology.

Table 4: Concept representing inputs to building permit service

concept BuildingPermitApplicationRequest nonFunctionalProperties dc#description hasValue "concept representing input to building permit service" endNonFunctionalProperties applicant ofType (1 *) personData#Person construction ofType (1 1) Construction delegate ofType (0 1) personData#PhysicalPerson sitePlan ofType (1 1) segofUtil#File floorPlan ofType (1 1) segofUtil#File constructionViews ofType (1 1) segofUtil#File buildingSiteEligibility ofType (0 1) segofUtil#File

In table 4 a WSML concept representing relevant input to the building permit service is shown. This concept is characterized by several properties, like applicant, construction, delegate etc, which in turn represent WSML concepts. These properties – so called object-properties since they do not hold simple data values but references to other concepts instead – have to be further explored by a citizen within the generated form (see Figure 7). For each object-property in the generated form a button is added to further specify the values of the object-property.

By clicking such a button, another form with all properties of the previously selected concept is generated. As shown in Listing 4 an "applicant" is of type "Person". A person is characterized by personal data and address data. The form representation of the concept "personal data" is shown in Figure 8. This concept doesn't consist of object-properties but consists of so called datatype-properties that hold simple data values like string, integer or date values.

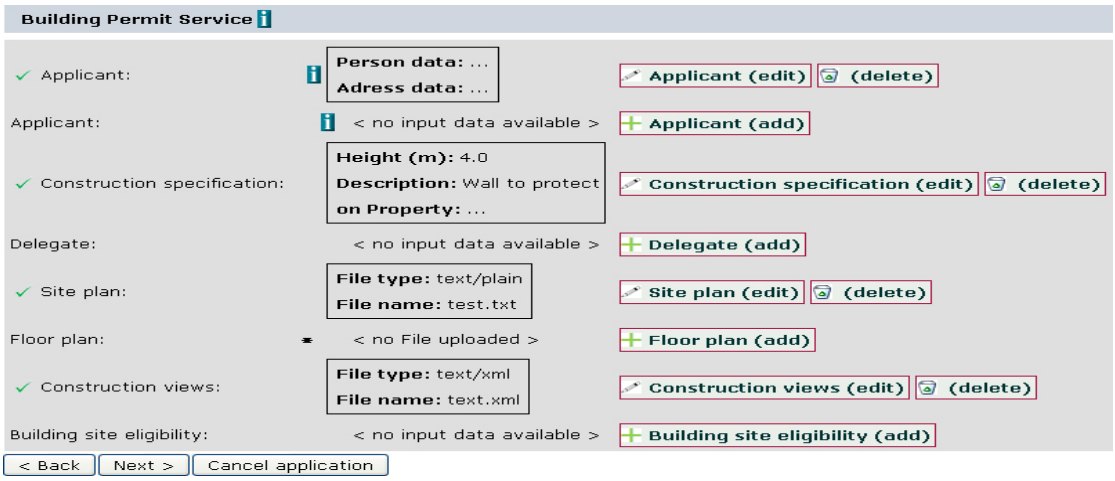

Figure 7: Form representation, object-properties

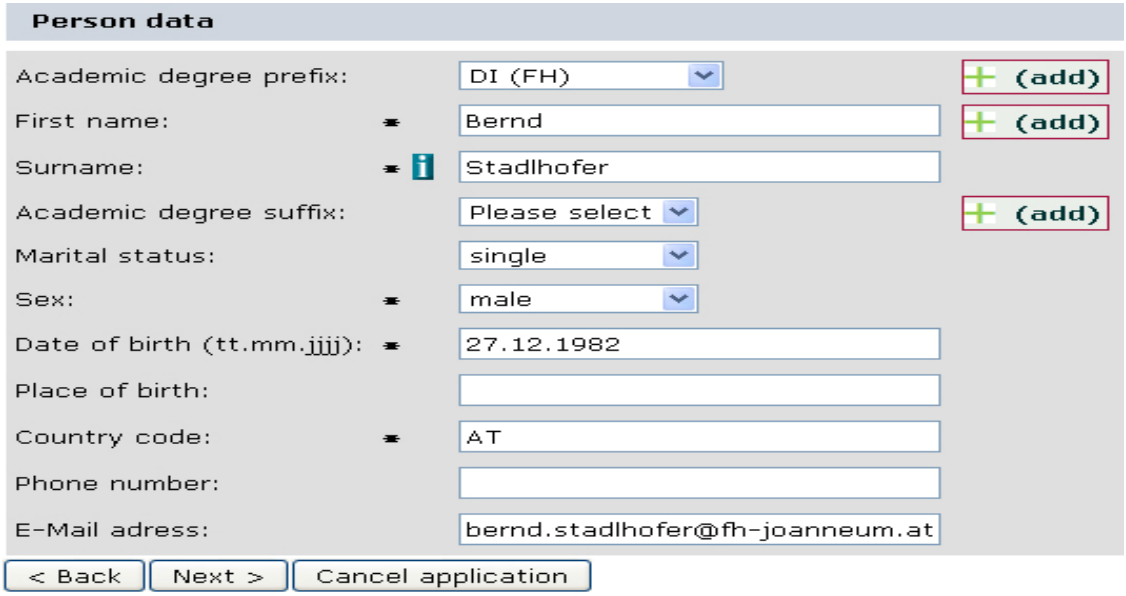

Figure 8: Form representation, datatype-properties

After the user has filled in all required data an instance of the corresponding WSML concept (see Table 4) is generated within the ontology and existing axioms are used to validate its correct state.

When developing the idea of ontology driven e-Government it was one main idea to achieve a strong decoupling between the form solution and the backend. Such a decoupling can be achieved by transforming the input data into a common data interchange standard format, which was EDIAKT II (Freitter et al. 2006) – an XML Schema definition for the exchange of electronic documents between public authorities in Austria – in our case. Following this approach the input data can be consumed by

any application supporting the data interchange standard EDIAKT II like the SOA-backend described in the next section.

7. Accessing SOA backend

In this section we propose a SOA (Service Oriented Architecture) backend for ontology driven e-Government. Thereby it is our intention to implement a so-called grounding for each semantic web service to a concrete web service, which is typically represented by a WSDL file. The WSMX (Web Service Modeling eXecution environment) (Fensel et al. 2008) developed within the WSMO Project (Fensel et al. 2007) represents one promising approach to establish groundings from a semantic web service to a concrete web service. As both, WSMX as well as the semantic web services and ontologies from our prototype implementation are based on WSML (Bruijn et al. 2006), the decision to use WSMX as execution framework is obvious.

As already mentioned the concrete implementation of the web service is represented and described by a WSDL file. Behind this service description in our case is a BPEL (Business Process Execution Language) process – implemented with "openESB" (open Enterprise Service Bus). Since every concept can be automatically transformed into XML-schema and each of its instances into XML based on this schema, all data gathered from the user can be seamlessly used within BPEL processes. This allows for very easy process composition, utilizing other already existing services like a central register of residents to make plausibility checks of the citizens' address data. Not only web services but almost every single service offered by public agencies could be involved in such a BPEL process as long as it is attached to the service bus. Thus entire – or at least major parts of – back office business processes within the public agency involved are supported electronically from the beginning – which is the request of the citizen – to the end – which typically is the success/failure notification about citizens' applications to the citizens.

8. Conclusions

The approach to using semantic technologies in e-Government presented in this paper is a first step towards an ontology based process to implement e-Government services, where modeling the ontology should be one of the first steps. Goal and service discovery is one important part of this type of e-Government solutions. Using semantic reasoners to identify the concepts involved is very powerful and can hide much of the complexity of underlying regulations from citizens. The concept tree used in this example to guide the user in identifying the concept actually involved in her goal is a good basis to start with.

Intelligent electronic forms – as described in section 6 – that know the current context they are running in, greatly simplify interaction with public agencies and back office business processes. Orchestrating all participating back office services with a tool like "openESB" – as proposed in Section 7 – would also greatly simplify the work of the particular public agency's staff as the transparency of the offered public service itself is tremendously increased.

One of our next goals is to further enhance the quality of the citizen's input data to public services on the one hand and to further increase the system's usability on the other hand. One major step thereby will be to shift services, which are currently located in the back office layer, to the application layer. Preferably, it should be possible to verify an applicant's address and personal data during form completion, instead of in the back office. Thus, a semantic representation and integration of typical validation services like a central register of residents has to be accomplished. Besides validation services another type of services, so called data provider services (e.g. address list, list of states) will be investigated to finally achieve a citizen friendly, barrier free and easy to use user interface to our "Ontology driven E-Government" environment.

References

Bloehdorn, S., Haase, P., Sure, Y. and Voelker, J. (2005) D.6.6.1 Report on the integration ofML, HLT and OM [online] http://www.sti-innsbruck.at/fileadmin/documents/deliverables/Sekt/sekt-d-6-6-1-Int._ML__HLT__OM.pdf

Biagioli, C., Francesconi, E., Passerini, A., Montemagni, S., and Soria (2005) Automatic semantics extraction in law documents, in Proceedings of the 10th international Conference on Artificial intelligence and Law, ACM, pp 133-140

Bruijn, J.d., Lausen, H., Polleres, A. and Fensel, D. (2006) The Web Service Modeling Language WSML: An Overview, in The Semantic Web: Research and Applications, Lecture Notes in Computer Science, vol. 4011, Springer, pp 590-604

Bruijn, J.d., Lara, R., Polleres, A. and Fensel, D. (2005) OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web, in WWW '05: Proceedings of the 14th international conference on World Wide Web, ACM

Crichton, C., Davies, J., Gibbons, J., Harris, S. and Shukla, A. (2007) Semantic frameworks for e-government, in ICEGOV '07: Proceedings of the 1st international conference on Theory and practice of electronic governance, ACM, pp 30-36

Ehrig, M., Haase, P., Stojanovic, N. (2004) Similarity for ontologies - a comprehensive framework, in Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, at PAKM 2004

Fensel, D., Lausen, H., Polleres, A., Bruijn, J.d., Stollberg, M., Roman, D. and Domingue, J (2007) Enabling Semantic Web Services: The Web Service Modeling Ontology, in Enabling Semantic Web Services: The Web Service Modeling Ontology, Springer, Berlin, Heidelberg, NJ

Fensel, D., Kerrigan, M. and Zaremba, M (2008) Implementing Semantic Web Services: The SESA Framework, in Implementing Semantic Web Services: The SESA Framework, Springer, Berlin, Heidelberg

Freitter, M., Gradwohl, N. and Denner, R. (2006) Empfehlung für das XML-Schema zu EDIAKT II, V. 1.2.0 (German), [online], Bundeskanzleramt Österreich http://www.ag.bka.gv.at/index.php/Portal:Ediakt

Kifer, M., Lausen, G. and Wu, J. (1995) Logical foundations of object-oriented and frame-based languages, in Journal of the ACM (JACM)

Leitner, C. (Ed.) (2003) eGovernment in Europe: The State of Affairs [online], www.eipa.eu/files/repository/product/20070214113429_egoveu.pdf

Lu, L., Zhu,G. and Chen, J. (2004) An Infrastructure for E-Government Based on Semantic Web Services, in SCC '04: Proceedings of the 2004 IEEE International Conference on Services Computing, IEEE Computer Society, pp 483-486

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N. and Sycara, K. (2005) Bringing Semantics to Web Services:The OWL-S Approach, in Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition, Springer, pp 22-42

Miller, C. (Ed.), Mukerji, J. (Ed.), Burt, C., Dsouza, D., Duddy, K., El Kaim, W., Frank, W., Frankel, D., Iyengar, S., Miller, J., Mischkinsky, J., Mukerji, J., Siegel, J., Soley, R., Tyndal-Biscoe, S., Uhl, A., Watson, A. and Wood, B. (2001) Model Driven Architecture (MDA), Document number ormsc/2001-07-01, Architecture Board ORMSC, OMG

OMG (2002) Meta Object Facility (MOF) Specification [online] http://www.omg.org/docs/formal/02-04-03.pdf

Polleres, A., Lara, R. and Roman, D. (2004) D4.2v01 Formal Comparison WSMO/OWL-S [online], DERI http://www.wsmo.org/2004/d4/d4.2/v0.1/20040315/

Schweighofer, E., Rauber, A. and Dittenbach, M. (2001) International Conference on Artificial Intelligence and Law, in Proceedings of the 8th international conference on Artificial intelligence and law, ACM, pp 78-87

Wang, X., Vitvar, T., Peristeras, V., Mocan, A., Goudos, S. and Tarabanis, K. (2007) WSMO-PA: Formal Specification of Public Administration Service Model on Semantic Web Service Ontology, in Proceedings of the 40th Hawaii International Conference on System Sciences, HICSS, pp 1-10

W3C (2004) OWL Web Ontology Language Overview [online], W3C http://www.w3.org/TR/owl-features/

