Ontology Modeling for Goal Driven E-Government

Peter Salhofer
FH JOANNEUM
University of Applied Sciences, Graz
peter.salhofer@fh-joanneum.at

Bernd Stadlhofer
FH JOANNEUM
University of Applied Sciences, Graz
bernd.stadlhofer@fh-joanneum.at

Abstract

This paper presents an approach to model ontologies in the e-Government domain so that they can be easily used to assist a citizen in formally expressing a goal that can in turn be used for service discovery. This process is also known as “goal discovery” and can become relatively complex depending on the regulations that have to be applied. By introducing some guidelines and constraints on the model, this process can be simplified and complexity can be hidden from the user. We found that the concept tree, used to guide the user in uniquely identifying the goal, is a good basis to start with, since it is easy to use and also easy to maintain in comparison to other approaches. Our next steps will be to automatically generate forms based on ontologies.

1. Introduction

The basic idea of the approach presented in this paper is based on comprehensive experience in some e-Government implementation projects with a clear focus on G2C (Government to citizen). While these solutions are becoming increasingly mature and fully transactional[1], the actual interaction between system and citizens, which in most cases is a form that has to be filled in, is becoming the central point of the application. Users expect “intelligent” forms that are able to perform interactive plausibility checks on the information just filled in as well as auto-completion of fields. Thus more and more logic is integrated in the form itself which undermines well established software system design patterns like the Model View Controller pattern (MVC)[2]. This widely accepted pattern postulates the separation of every software application into the layers model (the data the application works with), view (elements that are used to display model elements) and controller (controls how model elements are manipulated). One benefit of this clear separation is enhanced maintainability of the entire system.

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[3][4] by adding and utilizing semantic annotations to web services.

Semantic methodologies can be used to add a layer of logic to any (text based) resource (see Figure 1). Thus the idea was born to use semantic methodologies in a more forward-engineering approach to create a semantic model first and to use this model e.g. as the basis for the automatic creation of “intelligent” web forms.

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 form creation. Discovering the citizen's actual goal is one of these use-cases and is actually the first step that has to be taken. Even if all frameworks for modeling semantic web services contain the notion of goals that is used for service discovery, the process of expressing goals can become rather complex.

2. GEA-PA – an e-government meta-model

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 choose a model proposed in the Government Enterprise Architecture – Public Administration (GEA-PA)[6] as shown in Figure 2.

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.

3. Goal-oriented e-government

One big advantage of semantic web services is the goal oriented approach they are based on. There are numerous services available on the web. 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[7]:

- **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.

This paper focuses on the goal discovery phase and the identification of a public service (and not a web service implementing it) that is needed to achieve the goal. 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 formulating the goal is needed.

In our prototype implementation of goal 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.
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.

4. 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[8] and WSMO[9]. We implemented evaluation prototypes based on both approaches and eventually chose WSMO which is based on WSML[10] over OWL-S which is layered on top of OWL[11]. While a detailed discussion of the differences between these two approaches can be found in [12] and [13], here we will simply state the reasons for our decision.

While OWL is based on XML, WSML can be seen as a domain specific language based on Meta Object Facility (MOF)[15]. As a consequence there is no XML overhead in WSML. Listing 1 and Listing 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[16]. 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.

5. 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[17][18] already exist, we conducted this step manually, identifying needed concepts and their interdependencies by carefully analyzing the text.

While OWL is based on XML, WSML can be seen as a domain specific language based on Meta Object Facility (MOF)[15]. As a consequence there is no XML overhead in WSML. Listing 1 and Listing 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[16]. 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.
5.1. 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[9]. 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[6]. 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 number and type 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.

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. 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 6. .

5.2. Specialization and classification

An ontology is defined as [19]:

\[ O_T := (C,T, \preceq_C, \preceq_T, R, s_R, s_T, \preceq_R, \preceq_A) \]

Consisting of a set of concepts \( C \) aligned in a hierarchy \( \preceq_C \), a set of Relations \( R \) with \( \preceq_R \), the signature \( s_R : R \rightarrow C \times C \), a set of datatypes \( T \) with \( \preceq_T \), a set of attributes \( A \) with \( \preceq_A \), the signature \( s_A : A \rightarrow C \times T \).

A similar definition of an ontology not including datatypes can be found in [20].

The is_a relationship between to classes is defined as follows:

\[ c_1 \preceq_C c_2 \mid c_1, c_2 \in C \quad c_1 \text{ is subconcept of } c_2 \]

The is_a relationship is transitive:

\[ c_1 \preceq_C c_2 \land c_2 \preceq_C c_3 \quad c_1 \preceq_C c_3 \]

Defining a function attr(c)

\[ \text{attr}(c) : C \rightarrow A \]

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

\[ c_1 \preceq_C c_2 \mid c_1, c_2 \in C \quad \text{attr}(c_2) \subseteq \text{attr}(c_1) \]

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:

![Figure 3: Part of the concept hierarchy](image-url)
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:

\[
\begin{align*}
    c_1 \leq_c c_2 \land c_1 \leq_c c_3 \quad | \quad \{c_1, c_2, c_3 \in C, \ \ c_2 \neq c_3 \land \neg (c_2 \leq_c c_3 \lor c_3 \leq c_2) \} \\
    (\text{attr}(c_2) \cup \text{attr}(c_3)) \subseteq \text{attr}(c_1)
\end{align*}
\]

6. The goal discovery algorithm

Initially based on the GEA-PA model, we have refined and adapted parts of this model as shown in Figure 4.

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

As shown in Figure 5, 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:

\[
\text{attr}(b') \equiv \text{attr}(b)
\]
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:

\[
\text{findService}(g, c): G[xC] \rightarrow C
\]

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 6. The question always follows the same pattern:

“You want to {build} knock down \| \ldots\} 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.

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 7). The sub-concept is identified by the semantic reasoner.

7. Walkthrough example

To point out the functionality of the presented approach we will provide a short walkthrough example in this section. Let’s assume we want to build a so-called ‘adjoining building’. This type of building is usually relatively small, close to an existing building and not used for living in. They are typically used as stables, storage space, private workshops or similar. Assuming that we have found the public agency’s goal definition web page we can go along to identify the goal that best reflects what we want. Thus, we will start with

“I want to build a structure”

We select this goal and are now asked the following question:

“You want to build a structure. Please further specify the type of structure”

From the list of available options we select “building”. Now almost the same question as before shows up. This time we are asked to further specify the type of building that we want to develop. From the list of available options we select “adjoining building”. So far the system has used specialization to navigate through the tree of concepts. Now, however, the system recognizes that there are some axioms that can be used to infer the concrete class of adjoining building. Among these axioms is the following:

\[
\text{axiom AxAxSmallAdjoiningBuildingForAgricultureAndForestry definedBy}
\]

\[
?x \text{ memberOf SmallAdjoiningBuildingForAgricultureAndForestry impliedBy}
\]

\[
?x[totalArea hasValue ?totalArea, numberFloors hasValue ?numberFloors, habitable hasValue ?habitable, floorHeight hasValue ?floorHeight, forAgriculturalUse hasValue ?forAgriculturalUse] memberOf AdjoiningBuilding and ?numberFloors < 2
\]

\[
\text{and wsml#equal(_boolean("false"), ?habitable)}
\]

\[
\text{and ?floorHeight <= 3}
\]

\[
\text{and ?totalArea <= 40}
\]

\[
\text{and wsml#equal(_boolean("true"), ?forAgriculturalUse).}
\]
The meaning of this axiom is almost self-explanatory. There is some \( x \) that pretends to be an \textit{AdjoiningBuilding} and therefore has to have some attributes. If the values of these attributes meet the constraints of the axiom than this \( x \) is a \textit{SmallAdjoiningBuildingForAgricultureAndForestry}.

The system analyses all available axioms that can be applied to adjoining buildings and presents a form containing all attributes that are used in these axioms. In this case we have to fill in the total area, the number of floors, room height, usage (whether the building is used for agriculture or forestry) and have to define whether the building will be habitable. We fill in all values according to our plan. Since we also want to have a guest room in the new building, we declare it as habitable. The applicable law, however, defines that an adjoining building cannot be habitable! Thus our adjoining building is not recognized by any of the axioms. Since an adjoining building is a sub-concept of \textit{buildings}, all axioms that can be applied to buildings, can and will be used in this case as well. One of the available axioms it this one:

\[
\text{axiom AxBuildingsForAgricultureAndForestry}
\]

\[
\text{definedBy}
\]

\[
\text{memberOf SmallResidentialHouse}
\]

\[
\text{impliedBy}
\]

\[
\begin{eqnarray*}
x & \text{totalArea} & \text{hasValue} & \text{totalArea, numberFloors} & \text{hasValue} & x & \text{numberFloors, habitable} & \text{hasValue} & x & \text{habitable} & \text{memberOf Buildings} \\
\text{and wsml\#equal\_boolean("true"), ?habitable) & x & \text{totalArea} & < 600 & \text{and } x & ?\text{numberFloors} & \leq 3.
\end{eqnarray*}
\]

If there is some \( x \) of type \textit{Buildings} with some attributes that meet the given constraints than this building is a \textit{SmallResidentialHouse}. Thus the project that we plan has to be submitted as a small residential house and not as an adjoining building. This has important consequences for selecting the appropriate approval. The system has selected the appropriate service based on our input.

This example shows the potential of semantic technologies. Even though we have already been in a wrong branch of our class hierarchy (that could also be seen as a decision tree), the system has brought us back to the correct concept by evaluating the available axioms.

8. Related work

A very interesting approach of goal formulation can be found in [21]. The tool described there supports ontology guided as well as natural language input. Systems that process natural language need a significant amount of training that covers almost all terms of the problem domain. The paper also mentions that people who start using the system prefer the guided approach since their first results with natural language are relatively poor, whereas more experienced users prefer the natural language approach. This leads to the conclusion that people first learn the terms that are understood by the system. In the case of e-Government, however, where the system is not used that frequently by an individual, this learning effect will most likely not happen. Thus the natural language approach does not seem to be very promising.

Another relatively similar approach is described in [22]. Instead of identifying a single service that meets a given goal, this approach is capable of identifying an entire workflow composed of several services potentially offered by different public authorities. Since they do not use any of the established semantic frameworks, they had to develop their own reasoning algorithms. The basic principles are very similar. Whereas in our approach a goal is further specified by selecting and specifying concepts that are related to the goal, in this approach a so called user profile is used. This user profile also contains information about concepts involved (e.g. whether a new business will have employees), but these attributes are detached from the actual concepts and are maintained in a separate attribute tree. They also mention, that the user profile information is gathered in an online interview session but do not present any further details about this step. The discovery algorithm used, takes the goal and the user profile (i.e. a set of attributes and their values) as input and calculates a directed graph of tasks based on rules and task dependencies. This leads to some redundancy since attributes have to be maintained in the rules as well as the tree that is used for gathering the user profile.

Another approach using almost the same technologies as we do, is presented in [23]. Whereas the system described in [22], as well as our method, uses goals in combination with additional information (user profile or related concepts respectively), this approach uses a tree of goals that is defined using WSMO concepts. Instead of composing a tree by creating a concept hierarchy (using the sub-concept construct) they have decided to create several \textit{Node} concepts (with attributes like “hasParentNode” and “hasChildNode”) to model a tree. Like in our approach, only leaves of the tree are non-abstract. Thus, they also have to traverse the tree. In contrast to our approach that uses specialization and classification.

Another approach using almost the same technologies as we do, is presented in [23]. Whereas the system described in [22], as well as our method, uses goals in combination with additional information (user profile or related concepts respectively), this approach uses a tree of goals that is defined using WSMO concepts. Instead of composing a tree by creating a concept hierarchy (using the sub-concept construct) they have decided to create several \textit{Node} concepts (with attributes like “hasParentNode” and “hasChildNode”) to model a tree. Like in our approach, only leaves of the tree are non-abstract. Thus, they also have to traverse the tree. In contrast to our approach that uses specialization and classification.
to navigate through the tree, they use questions that are also stored as text attributes in the nodes ("hasQuestion"). To allow meaningful reasoning, this approach uses an auxiliary citizen ontology that is specific to the goal tree of a given domain. When extending this approach to other domains, another specific user ontology has to be modeled.

9. 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 solution. In this project we have seen that expressing the citizen's goal in a way that can be used to unambiguously determine required services, probably might go beyond the capabilities of existing frameworks and needs some smart tool support. 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, since it is easy to use compared to other approaches like the one described in [21]. At any time, based on a set of attribute values, the reasoner is still able to identify a concept in a different branch of the tree as the correct one, thus bringing the user back to the correct path.

Paper forms that were used to identify additional concepts and their attributes contain fields for all possible cases and are therefore almost always potentially overloaded. Intelligent electronic forms, that know the current context they are running in, would greatly simplify interaction with public agencies. Thus one of our next steps will be to automatically generate forms based on ontologies.

10. References


