Evaluation experiment for the editor of the WebODE ontology workbench

Óscar Corcho, Mariano Fernández-López, Asunción Gómez-Pérez

Facultad de Informática . Universidad Politécnica de Madrid
Campus de Montegancedo, s/n. 28660 Boadilla del Monte. Madrid. Spain
{ocorcho, mfernandez, asun}@fi.upm.es

Abstract. We summarize our design decisions on the conceptualization of a travelling ontology, when building it with the ontology editor of the WebODE ontology engineering workbench. This ontology editor is composed of a set of HTML forms, a graphical taxonomy editor called OntoDesigner and an axiom editor called WAB (WebODE Axiom Builder).

1 Introduction

This paper presents the results of using the ontology editor of the WebODE ontology engineering workbench to conceptualize an ontology in the domain of travelling and lodging. This is the first experiment for the evaluation of ontology tools' editors, performed in the context of the Special Interest Group (SIG) on Enterprise-Standard Ontology Tools of the European IST OntoWeb thematic network (IST 2000-29243). This experiment is described in section 2.5 of the OntoWeb Deliverable 1.3 [6].

In section 2 we will briefly describe the WebODE ontology environment, its ontology editor and its knowledge model. Section 3 will present the design decisions that we have made to model this ontology in WebODE, focusing on those pieces of knowledge that we have been able to model and on those pieces of knowledge that we have not been able to model with it. Section 4 briefly comments on the formats used to deliver the ontology: the XML representation of WebODE and RDF(S). Finally, section 5 will present some conclusions that can be derived from this experiment.

2 The WebODE ontology engineering workbench

WebODE [4] is an ontology engineering workbench developed by the Ontology Group at the Technical University of Madrid (UPM). It is the successor of the ontology design environment ODE [2].

WebODE is easily extensible and scalable, supported by an application server. The core of WebODE is its ontology editor. Ontologies are browsed and edited either with HTML forms (which allow editing ontology components and which provide "copy&paste" functionalities) or with a graphical user interface, *OntoDesigner* (which

allows managing different views, where we can edit concept taxonomies with subclass-of relationships, disjoint and exhaustive subclass partitions and part-of relationships, and ad-hoc binary relations, and where we can either show or hide the different kinds of relationships in the ontology to highlight parts of it). The ontology editor also provides constraint checking capabilities, axiom and rule creation and parsing (with the WAB editor [4]), documentation in HTML, ontology merge, and ontology exportation and importation in different formats (XML, RDF(S), OIL, DAML+OIL, CARIN, Flogic, Java and Jess). Finally, its built-in inference service uses Prolog and a subset of the OKBC protocol [3].

2.1 WebODE's knowledge model

The WebODE's knowledge model [1] is based on the intermediate representations proposed in Methontology [6]. It allows modelling **concepts** and their **attributes** (both class and instance attributes), **concept taxonomies**, **disjoint** and **exhaustive** class partitions, **ad-hoc binary relations** between concepts, **properties of relations**, **constants**, **axioms** and **instances** of concepts and relations.

Bibliographic references can be attached to any of the aforementioned ontology components. Besides, it is possible to import terms from other ontologies. **Imported terms** are referred to by means of URLs.

Finally, the WebODE's knowledge model supports **views** and **instance sets**. Views highlight specific parts of the ontology in OntoDesigner. Instance sets make possible to populate a conceptual model for different applications or scenarios, maintaining different, independent instantiations of the same conceptual model in WebODE.

3 Conceptualization of the travelling ontology in WebODE

The ontology that we present has been conceptualized using Methontology. Methontology proposes to conceptualize the ontology using a set of tabular and graphical intermediate representations (IRs), and recommends the following order to assure the consistency and completeness of the knowledge already represented. First, we must identify the main concepts in the ontology and build the concept classification tree. Second, we create the ad-hoc binary relations between concepts in the same taxonomy or in different taxonomies. Then, we add the class attributes and instance attributes to the concepts, and finally we create axioms and rules. This is just a recommendation: this process is not necessarily sequential.

Therefore, our first task consisted of extracting concept taxonomies and their ad-hoc relations from the ontology description. We created five different **views**:

?? `Trip` view. A `customer` makes one or more `trips`, which use some kind of `transport` and `accommodation`. Here we understand by trip a combination of one or several transports and (possibly) an accommodation. That is, in the example, John will make three different trips: the one from Madrid to NY, the one from NY to Washington DC, and the one from Washington DC to Madrid.

?? `Means of transport` view, which contains all the concepts relevant to means of transport, classified by `air`, `ground` and `sea transportation`.
?? `Plane` view, which presents the concept taxonomy under the concept `plane`, which is contained in the previous view. This is done since the concept plane is the only one being more specialized in the ontology.
?? `Location` view, which contains the concepts `city`, `airport` and `importantPlace`, and their ad-hoc relations: a city may have several `nearest airports`, and several `important places` worth to visit.
?? `Accommodation` view, which contains the concepts related to accommodation. The most general concept is `accomodation`, which specializes in `hotels` and `bed and breakfasts`, as proposed in the NL description of the example.

In these views we modelled, if possible, disjoint and exhaustive partitions instead of simple subclass-of **concept taxonomies**. For instance, in the accommodation view we created an exhaustive partition of the concept `hotel` in the different hotel categories (from 1 star to 5 stars). And in the same view, we created a disjoint partition of the concept `accommodation` in `hotel` and `bed and breakfast`, since there are other types of accommodation that have not been included in the ontology.

Another important design decision related to the concept taxonomies was that of transport. We defined two different concept taxonomies for transports, considered as services (`flight`, `city bus`, `taxi`, `rental car`, etc.) and means of transport (`plane`, `car`, `bus`, `underground`, etc.). Both taxonomies are connected with the ad-hoc relation `usesTransportMean`, which is defined between the most general concepts in both taxonomies (from the concept `transport` to the concept `transportMean`) and specialized in some of the more specific concepts. For instance, between the concepts `taxi` and `car`, between `flight` and `plane`, etc.

Besides, another important issue is how to define flights from one city to another. We classified flights according to the air company in charge of them (this does not prevent a specific flight being subclass of several air companies' flights, in case of joint flights), and created a class for each flight code, that is, `aa0415`, `us1453`, etc. Instances of these concepts will be the specific flights in a specific date.

Once that we defined concept taxonomies and **ad-hoc relations** between concepts, we deepened in the description of concepts by defining concept attributes. We have selected the attributes that we considered most relevant for each concept, according to the description provided in the example. For each attribute, we provided its NL description, its value type, its minimum and maximum cardinalities, and its measurement unit, precision, minimum and maximum values for numerical attributes.

Class attributes define properties that describe the concept. For instance, the `number of stars` of each kind of hotel, the `economy`, `first` and `business class standard prices` of each kind of flight, the `air companies` of a flight or the `typical cruise speed` of a kind of airplane. These attributes have also their corresponding values.

Instance attributes define properties that will take their values in instances of the concept. For instance, in the concept `accommodation` we defined as attributes the `address`, `URL`, `phone number`, `number of rooms`, `number of available rooms`, `dogs allowed`, `distance to the beach` and `distance to a ski resort`. In the concept `flight`, we defined the `air company`, and the `departure` and `arrival dates`. In the concept `place`, we defined `longitude` and `latitude`, which can be used later to compute distances.

Later, we moved to the **logical axioms**, which are defined in first order logic, according to the WAB syntax. We created the following seven axioms:

- ?? Two axioms stating that the business class standard price of a flight is always more expensive than the first class standard price of the same flight, and for stating that the first class standard price of a flight is always more expensive than the economy class standard price of the same flight.
- ?? One axiom to obtain a flight's arrival city from the flight's arrival airport
- ?? Two axioms to establish that the only kind of transport that can be used for going from America to Europe, and vice versa, is a flight.
- ?? One axiom to state that in every kind of city transport the arrival city and departure city are the same.
- ?? One axiom to obtain the preferences of a customer for a trip, depending on the distance between two cities, as presented in the NL description of the example.

Many other constraints could have been inferred from the NL description of the problem. However, we have tried to restrict to the most relevant ones and those defined explicitly in the text.

A **bibliographic reference** was used to obtain many of the NL descriptions of concepts, attributes and relations: the `Merriam-Webster on-line`.

Finally, we created two **instance sets**: one for an agency in New York and another one for an agency in Madrid. We have created all the instances in the first one: an instance for `John`, three instances for John's trips, instances for the two hotels to be used, for the cities that he will visit, for the `Statue of Liberty` and for John's flights. We also created the instances of relations between these instances.

As a summary, we created in this ontology 58 concepts (organized in concept taxonomies with 23 subclass-of relations, 6 disjoint and 3 exhaustive partitions), 19 class attributes, 28 instance attributes, 21 relations, 0 constants, 1 reference and 7 axioms. We created 23 instances in the New York Agency instance set.

We found the following difficulties when modelling our ontology in WebODE:

Enumerated types cannot be represented in WebODE, in the sense of allowed values for an attribute in a class. For instance, they would be useful for representing the allowed values of the attribute hotel chain in the concept hotel, or for the continents in the attribute continent of the concept city.

We cannot represent attributes attached to ad-hoc relations. For instance, the number of rooms of each room type in a hotel. To represent this, we should create an

intermediate concept that represents the relation, and define the corresponding instance attributes in it. However, we lose clarity and legibility in the representation.

WebODE cannot compute distances between places. This calculation must be done by external systems (such as inference engines or traditional software systems), which would be in charge of creating the corresponding instances of the concept `distance`.

4 Formats in which the ontology has been delivered

We have this ontology in two formats, automatically generated from the WebODE ontology editor: (1) **WebODE's XML** (in which we have all the components of the ontology that we have presented in the previous section), and (2) **RDF(S)**. In the transformation to RDF(S), we lose much of the knowledge of the WebODE ontology, since the RDF(S) knowledge model is less expressive than WebODE's. Hence, in RDF(S) we do not represent partitions, some attribute's information (cardinalities, measurement units, precision, minimum and maximum values), axioms and views.

5 Conclusions

In this experiment we have developed an ontology from a short NL description of the problem to be solved. It is clear that the ontologies that will be presented in this workshop will be very different from each other, since the problem description left open many modelling issues, so as to allow exploiting each ontology tool features.

This experiment is the starting point of a set of experiments that can be conducted by the ontology community. The domain of the experiment can be enriched, and also this experiment can be used for multiple purposes. For instance, it can be used: (1) to evaluate the tools' knowledge models, so that we can determine which components can be represented in each tool and which components cannot be represented, (2) to evaluate the possibilities of integrating the output generated by these tools with other ontology techonology (parsers, inference engines, etc.); (3) to analyze the possibilities of interoperability among tools; (4) to evaluate the usability of each tool; etc.

Acknowledgements

This work has been partially funded by the OntoWeb thematic network (IST-2000-29243) and by a FPI (*Formación de Personal Investigador*) grant from UPM.

References

1. Arpírez JC, Corcho O, Fernández-López M, Gómez-Pérez A (2001) *WebODE: a scalable ontological engineering workbench*. First International Conference on Knowledge Capture (K-CAP 2001). Victoria, Canada.

2. Blázquez M, Fernández-López M, García-Pinar JM, Gómez-Pérez A (1998). *Building Ontologies at the Knowledge Level using the Ontology Design Environment*, Proceedings of the Eleventh Knowledge Acquisition Workshop, KAW98, Banff, 1998.
3. Chaudhri VK, Farquhar A, Fikes R, Karp PD, Rice JP (1997) *The Generic Frame Protocol 2.0*. Technical Report, Stanford University.
4. Corcho O, Fernández-López M, Gómez-Pérez A, Vicente O (2002) *WebODE: an integrated workbench for ontology representation, reasoning and exchange*. 13th International Conference on Knowledge Acquisition and Knowledge Management (EKAW'02). Sigüenza. Spain.
5. Fernández-López M, Gómez-Pérez A, Pazos J, Pazos A (1999) *Building a Chemical Ontology using methontology and the Ontology Design Environment*. IEEE Intelligent Systems and their applications 4(1):37-45.
6. Gómez-Pérez A (editor) (2002) *Deliverable 1.3: A survey on ontology tools*. OntoWeb deliverable.