Using XSLT for Interoperability: DOE and The Travelling Domain Experiment

Antoine Isaac[1,2], Raphaël Troncy[1,3], and Véronique Malaisé[1,4]

[1] Institut National de l'Audiovisuel, Direction de la Recherche, Équipe DCA
4, Av. de l'Europe - 94366 Bry-sur-Marne
{aisaac,rtroncy,vmalaise}@ina.fr
http://www.ina.fr/
[2] Université de Paris-Sorbonne, LaLICC, http://www.lalic.paris4.sorbonne.fr
[3] INRIA Rhône-Alpes, Équipe EXMO, http://www.inrialpes.fr/exmo
[4] AP-HP, Équipe STIM, http://www.biomath.jussieu.fr

1 Introduction

This paper presents the results of the use of the *Differential Ontology Editor*[5] (**DOE**) during the second experiment on the evaluation of ontology-related technologies that was initiated by the OntoWeb thematic network[6].

The first experiment aimed at evaluating the modeling of an ontology in various environments through a shared description of the travelling domain written in natural language. The workshop (organized at the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002) in Spain) showed that the results were very heterogeneous [1]. This second experiment proposes to analyze how an ontology can be exchanged (exported and/or imported) between different tools.

We have used the following protocol to perform this experiment:

1. Reuse the ontology modeled with the DOE tool for the EON 2002 Workshop;
2. Perform a circular transformation with RDFS as exchange language, that is:
   - Export the DOE ontology in RDFS [10], import it in each of the four following environments: **Protégé2000** [5], **OilEd** [4], **OntoEdit** [11] and **WebODE** [2] and assess the quality of the transformation;
   - Modify the ontology in these environments and export it again in RDFS; Import the result in DOE and assess the quality of the transformation;
   - Compare the original ontology and the final ontology that results from this circular transformation.
3. Perform another circular transformation with the same tools, but with OWL [6] as exchange language.

[5] The tool is available for free at http://opales.ina.fr/public/. A repository of ontologies are available in various languages at http://opales.ina.fr/public/eon2003/.

[6] See the Special Interest Group homepage at http://delicias.dia.fi.upm.es/ontoweb/sig-tools/index.html.

The remainder of the paper is organized as follows. In section 2, we draw some conclusions about the last experiment and the very heterogeneous ontologies developed. Section 3 presents the DOE methodology, the DOE tool and our design choices for modeling the travelling domain ontology. We also introduce the way we use XSLT stylesheets to exchange the ontology model, since all languages have an XML serialization. The experiment begins in section 4, where we follow the protocol with RDFS as exchange language, and continues in section 5, with OWL as exchange language. Finally, we give in section 6 the conclusions that can be derived from these experiments.

2 Lessons Learned From the EON'2002 Workshop

The first EON workshop led to the creation of ten ontologies that were very heterogeneous with respect to their conceptualization. First of all, the ontologies can be classified in two categories: the ones with top-level concepts and relations, and the ones without. The first category is clearly more interested in the taxonomy structure (with design decisions inspired by the philosophy) whereas the second one is rather focused on the concepts that the user needs. Regarding the description of the domain, all ontologies have, more or less, modeled the same concepts. However, the taxonomies produced are quite different. The main differences appear in the branches "means of transport" and "reservation".

According to us, these differences are mainly due to the lack of particular use cases that could guide the ontology design. The application using this ontology was not specified and each ontologist has freely interpreted how to define and classify the concepts. The knowledge representation paradigm supporting the modeling and the tool itself contribute to the variability of the ontologies built. For instance, tools based on the frame paradigm support concept attributes (binary relations) whereas DOE can model n-ary relations between concepts.

3 DOE: The Differential Ontology Editor

Many approaches (for a complete survey, the reader can refer to the OntoWeb Technical RoadMap[7]) have been reported to build ontologies, but only few of them detail the steps needed to obtain and structure the taxonomies. This observation has previously led us to propose a methodology entailing a semantic commitment to normalize the meaning of the concepts [3]. We briefly present this methodology in section 3.1. In section 3.2, we describe how the DOE editor implements this methodology, and particularly how its exchange facilities, via import and export procedures, are performed by XSLT transformations. Finally, we present the design decisions that we have made to model the travelling domain ontology (section 3.3).

[7] `http://babage.dia.fi.upm.es/ontoweb/wp1/OntoRoadMap/index.html`

3.1 General DOE Methodology

As shown in [3], none of the methodologies reported to build ontologies force the ontologist to explicit the real meanings of the concepts. The terms used to refer to the concepts are still liable to multiple interpretations. This results in possible misunderstandings and consequently bad modeling and use of the ontology. As a solution, we have already suggested a three-steps methodology that consists in:

- a *semantic normalization*: aims at reaching a semantic agreement about the meaning of the labels used for naming the concepts;
- a *formalization*: aims at formalizing the ontology, that is, define the concepts (in the Description Logics sense), constrain the relations, add axioms (e.g. algebra properties), add instances, etc.
- an *operationalization*: allows to equip the concepts with the possible computational operations available in a particular knowledge representation language.

The first step is based on *differential semantics* [8]. Practically, the ontologist has to express, in natural language, the similarities and differences of each notion (concept or relation) with respect to its neighbors: its parent-notion and its siblings-notions. The result is a taxonomy of notions, where the meaning of a node is given by the gathering of all similarities and differences attached to the notions found on the way from the root notion (the more generic) to this node. We follow four principles to explicit this information:

- The *similarity with parent* principle (or **SWP**): explicits why the notion inherits properties of the one that subsumes it;
- The *similarity with siblings* principle (or **SWS**): gives a semantic axis, a property – assuming exclusive values – allowing to compare the notion with its siblings.
- The *difference with siblings* principle (or **DWS**): precises here the property value allowing to distinguish the notion from its siblings;
- The *difference with parent* principle (or **DWP**): explicits the difference allowing to distinguish the notion from its parent;

For example, Figure 1 gives the differential principles bound to the notion `MeansOfTransport`.

In the second step, the notions that come from this conceptualization are added with a formal meaning. The ontologist can define some concepts, precise the domains of the relations, add axioms and rules or give instances for some concepts. Tools make also some consistency checking. For instance, they have to check the propagation of the arity along the hierarchy of relations – if specified – and the inheritance of the domains. The third and last step of the methodology allows to export the ontology modeled into a knowledge representation language.

3.2 The DOE Editor

DOE is a simple prototype that supports partially the three steps of the methodology detailed above. It is not intended to bring a direct competition with other existing environments. Rather, its purpose is to demonstrate by experimentation how taxonomy structuring can benefit from our proposed methodology.

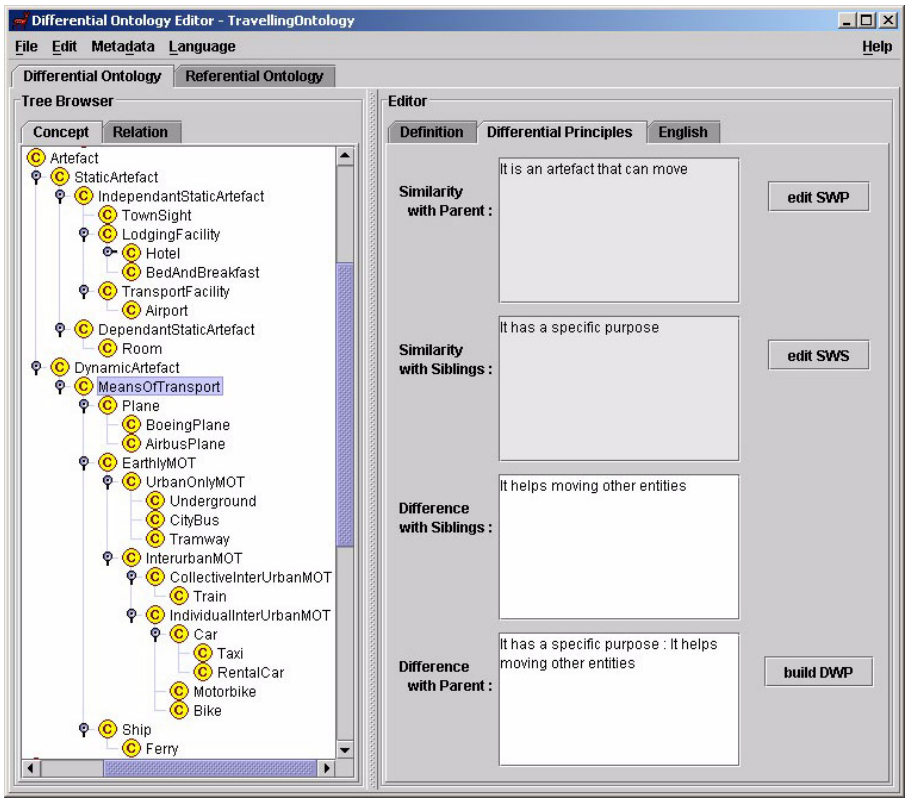

Fig. 1. The differential principles bound to the notion *MeansOfTransport* in the *DOE* tool

During the first step, the ontologist can enter the definition of the notions according to our principles. The tool automatizes partly this task. The Figure 1 shows the interface with the concept `MeansOfTransport` highlighted, and its differential principles fillers. For the second step, the taxonomies built previously are shown and the editor allows the ontologist to specialize existing concepts and relations (without the differential information), as well as to specify the arity and domains of the relations. The last step is implemented by exporting the referential ontology into commonly-used knowledge representation languages

(RDFS, DAML+OIL or OWL for instance) that can be used by specific application environments. This export mechanism also allows to refine the ontologies built, using other editors and the features they support.

All the exchange facilities (import and export from various languages[8]) are performed, in an original way, with XSLT [12] transformations. Actually, all proposed languages for representing ontologies on the Web have an XML serialization and the ontology editors themselves have usually their model described in XML. Therefore, XSLT, which is a language for transforming XML documents into other XML documents, seems adapted to perform this task. The ontologist can also use its own stylesheet, dynamically from the file menu, in order to import (resp. export) ontologies. In this case, the user has to specify the URI of the stylesheet and the input (resp. output) source. This feature provides a flexible way to accomplish the interoperability between DOE and others ontology builder tools.

3.3 The Travelling Domain Ontology

After comparing with the other ontologies presented at the previous workshop, we made some minor changes in our ontology (add some concepts and relations). Our ontology contains a top level to be consistent with our methodology. The first distinction that we make concerns the possibility for entities to be spatio-temporally located or not (`ConcreteEntity` and `AbstractEntity`).

The `ConcreteEntity` is then considered mostly in regard of its spatial or temporal location. `TemporalEntity` includes the `Reservation` types. One difference with other ontologies is the treatment of *flight*. Here, this concept is seen as a special reservation. For any kind of reservation, the relation `motUsed` can be established with a particular means of transport (e.g. `Plane` for the `FlightReservation`). There are three kinds of `SpatialEntity`: `Biological-Object` (the travel agent or the customer), `GeographicObject` (continent, country, city or resort) and `Artefact`. This last branch is composed of the different means of transport (by air, sea or earth) and of all types of building (town sight, hotel or transport facility).

The taxonomy of relations is not very deep. They are grouped according to their domain, like attributes (or slots) in other knowledge representation paradigms. The DOE editor, because of its Conceptual Graphs model, supports n-ary relations. We found this possibility particularly useful to model the relation `distance` that involves two spatial objects and the distance metrics.

Finally, it is not possible to write axioms in DOE because its purpose is mainly to guide the ontologist during the very first steps of the ontology conceptualization. Therefore, we could not specify that a travel between America and Europe could only be done with an airplane or a ship, or that the one to five star hotels were the only possible hotel categories.

[8] Technically, our editor can import ontologies written in RDFS and OWL, and supports export in RDFS, OWL, DAML+OIL and CGXML (a language for Conceptual Graphs specification). For adequacy concerns, we have limitted our paper to the languages currently focused on by the Semantic Web community.

4 RDFS as Exchange Language

The four following environments (Protégé2000 v2.0 beta, OilEd v3.5, OntoEdit v2.6 free release and DOE v1.5) can import and export RDFS ontologies. Consequently, we can do the export/import loop from DOE to each of them and come back. The RDFS import functionality seems to be unavailable online for the WebODE v2.0 tool and hence, cannot be tested. However, the export feature is available and we tested it after having imported our ontology via OWL.

4.1 From DOE To Other Environments

As seen in section 3.1, the main contribution of the DOE tool is to force the ontologist to assign a clear meaning to concepts through the use of differential principles. Our experiment has shown that we can keep a trace of this *semantic commitment* in produced RDFS ontologies by exporting all the related information into the `rdfs:comment` element.

Regarding formal expressiveness, our model is very limited. In fact it is very close to RDFS: DOE allows concept/relation specialization, domain and range assignment for relations, and concept instanciation. Therefore, it is not surprising that our editor easily manages to translate this information.

We then tried to open the produced RDFS file in other environments (the results are summarized in Table 1). OilEd was able to read it properly, like Protégé2000, which nevertheless encountered some problems dealing with the accents on letters found in the ontology. However, these two editors were not able to import the metadata associated to the ontology even if they were written according to the Dublin Core recommendation. OntoEdit managed to import the ontology model, but transformed the Dublin Core container by adding 11 DC elements to the relation list. It also added a mysterious instance that did not appear anywhere in the display and, after a glance at the exported RDFS file, proved to be a generated instance that has the DC attributes entered in the container. Finally, its pure frame-oriented interface did not show properties that had no domain defined, whereas they still existed in the model, which is quite disturbing.

4.2 From Other Environments To DOE

More problems occurred when trying to import back the RDFS ontologies exported by other environments. Firstly, we could not properly import the file exported by Protégé2000. We must mention that neither OilEd nor OntoEdit succeeded in this ordeal: it is due to RDFS errors, such as the use of `rdfs:label` as an attribute (instead of a sub-element) of `rdfs:Class`.

Secondly, both OntoEdit and WebODE do not use the RDF abbreviated syntax to encode the ontology. All class and property definitions are serialized as `rdf:Description` instead of `rdfs:Class` and `rdf:Property`. Consequently,

	DOE	Protégé	OilEd	OntoEdit	WebODE
Number of Concepts	79	79[a]	79	79	
Number of Relations	48	48[b]	48	59	
Number of Instances	22	22	22	23	*NOT*
Multiple Inheritance	exported	preserved	preserved	preserved	*available*
Domains assignment	exported	preserved	preserved	preserved	*online*
Ontology metadata	exported	omitted	omitted	transformed	
Differential Definition	exported	displayed	displayed	displayed	

[a] Protégé adds 15 system classes.
[b] Protégé adds 34 system slots.

Table 1. Statistics of the travelling ontology modeled in DOE, exported in RDFS and imported in several environments

we built a more intricate XSLT stylesheet[9] in order to deal with every possible serialization. We also have to specify, by hand, an XML encoding information adapted for the letter with accents. With this minor change, we are able to properly import OntoEdit and WebODE outputs. The only thing we do not get back is the relation hierarchy, which is not exported by WebODE: during the OWL import, this information is translated into logical axioms that are not serialized into `rdfs:subPropertyOf` subelements during the export. Both tools export our differential definitions, but we cannot import them properly: since they are stored in unstructured `rdfs:comment` elements, it would require string parsing to get the original structure. Consequently, we store them in a text element that is usually used in our model to store unstructured definition elements.

Things were more simple with OilEd, which uses "pure" RDFS serialization. We got our two taxonomies back, as well as the domain and range assignments for the relations. However, instances are not dealt with by OilEd's simple-RDFS export. Our differential information was forgotten too: whereas OilEd successfully gets and displays `rdfs:comment` content, it does not export it. Therefore, we lost the most valuable piece of information issued by our editor.

4.3 Conclusion

Roughly speaking, there is no loss of information when exporting our ontologies in other environments with RDFS. One may ironically insist on the fact that it is because we have little formal information to lose. However, it is very important for us that such a step be a success, since we advocate using our tool as a precondition for using other tools to further formalize ontologies.

The problems with RDFS import (in fact, OntoEdit and WebODE RDFS) is strongly linked to the fuzziness of this norm syntax. We have dealt with every

[9] It also has to deal with other alternatives, such as using `rdf:about` or `rdf:ID` to specify the name of an entity.

possible encoding for a class statement, but one may wonder whether the best solution is to question the variability of RDFS encoding.

We also have to improve the theoretical validity (with respect to our own approach) of our translations: for example, when importing a concept inheriting from multiple parents in the hierarchy corresponding to our first methodological step, we choose the "differential image" of the first parent encountered to be the parent of the "differential image" examined concept[10].

5 OWL as Exchange Language

Among the five environments we study, three of them are able to import and export OWL ontologies: Protégé2000 v2.0 beta, WebODE v2.0 and DOE v1.5. However, OilEd has the ability to export models in OWL format: to test it, we have imported an RDFS ontology in OilEd and then exported it in OWL.

5.1 From DOE To Other Environments

Our experiment with Protégé2000 has been quite successful (see Table 2). It managed to import our OWL file, getting back the two taxonomies with multiple inheritance, differential definitions, domain and range assignments, as well as instance definitions. However, some instances whose name did not follow XML specification were collapsed after their first digits being truncated. For example, reading `717` and `777`, two instances of the `BoeingPlane` concept, resulted in creating one instance whose name was an empty string.

Furthermore, all instances that are not in the Protégé namespace[11] were not imported. Consequently, the OWL export of DOE puts all instances in this namespace, which is a strange hack to allow the interoperability between these two tools.

Concerning WebODE, concept and relation hierarchies were properly imported, as were the differential definitions and domain and range assignments. However, it failed in getting back the instances, whatever namespace they are linked to.

5.2 Other Environments To DOE

When importing Protégé2000 OWL file, DOE got very limited information. It is partly due to the restrictions of our model, which is limited compared to OWL expressiveness, and partly due to the choice of XSLT. For instance, we could not import individuals, since Protégé2000 declares them using the RDFS elements `<namespace:ClassName rdf:ID="instanceID"/>` which are difficult to catch

[10] Our differential taxonomy is a tree. However, our referential one is not, which implies that there is no real loss of formal inheritance information when importing a concept with multiple parents.

[11] `http://owl.protege.stanford.edu`

	DOE	Protégé	WebODE
Number of Concepts	79	79[a]	80[b]
Number of Relations	48	48[c]	48
Number of Instances	22	20	0
Multiple Inheritance	yes	yes[d]	yes
Domains assignment	exported	preserved	preserved[e]
Ontology metadata	exported	missing	missing
Differential Definitions	exported	displayed	displayed

[a] Protégé adds 15 system classes and 19 OWL-related classes.
[b] WebODE adds the `owl:Thing` concept.
[c] Protégé adds 34 system slots and 21 OWL-related slots.
[d] Slot inheritance is not displayed in the interface, but preserved in the model.
[e] WebODE explicitly assigns `owl:Thing` to relation domains and ranges that are not defined.

Table 2. Statistics of the travelling ontology modeled in DOE, exported in OWL and imported in Protégé and WebODE

and to transform with XSLT features. We could use string-parsing features, but the result would be quite hazardous.

The problem of instance import does not appear any more when importing OilEd OWL ontologies. Since instances are serialized using the element `owl:Individual` from the OWL Presentation Syntax [7], DOE could easily get them back using an XSLT stylesheet dedicated to the translation of OWL-Presentation Syntax ontologies. But OilEd has a weird strategy in encoding the ontologies, mixing the OWL RDF-XML exchange syntax for the terminological part and the OWL Presentation Syntax for the assertional part of the model.

OWL individuals are not exported by WebODE, as well as the subsumption relation between relations (during the import, it is translated into logical axioms which are not translated back into `rdfs:subPropertyOf` subelements). However, we got back the hierarchy of concepts, together with the `rdfs:comment` including our differential definitions. We also imported the list of relations with their domain definition, but this one is incomplete if it uses the `owl:Thing` concept explicitly introduced by WebODE, an error due to an incomplete namespace declaration when using `owl:Thing` in the export file.

6 Conclusion

We have carried out the experiment proposed by the EON Workshop in order to prove that the interoperability between different ontology editors is feasible. We started from DOE, a tool implementing a methodology for building ontologies based on differential semantics. We then exported the travelling domain ontology in various environments and came back to DOE using XSLT transformations. The model of the DOE editor is very simple. It can be easily exported to other

environments, which confirms our primary intuition: DOE can interoperate with them.

	Export	Import
Global comments	OK, put aside some string encoding and namespace problems	OK, but difficult when multiple serializations are allowed for a single fact
Concept/relation lists	OK	OK, but new concept and relation roots were added
Instances	almost OK	difficult with the abbreviated syntax
Formal definitions	OK, but limited information was exported	OK, but limited information had to be imported
Differential definitions	OK, displayed in unstructured comments fields	OK, but imported in unstructured comments fields
Ontology metadata	Dublin Core not properly dealt with by other environments	Dublin Core not dealt with by other environments

Table 3. Summary of the experiment

Using XSLT transformations in a limited expressiveness context has shown both successes and limits: we are able to export ontologies to other ontological frameworks, in a satisfactory way, the information we are interested in (the results are summarized in table 3). However, when we import ontologies built in those frameworks, we face more problems: some are linked to theoretical considerations (the status of the imported information regarding our methodology), others are linked to practical implementation shortfalls (the limited expressiveness of our formal apparatus), and others are linked to the lack of maturity of Semantic Web standards. However, the most important information is quite successfully dealt with, which comfort us in thinking that a limited but satisfying interoperability can be easily achieved by simple, syntax-based means.

Acknowledgments

We would like to thank Oscar Corcho for its helpful comments and for reviewing early drafts of this paper.

References

1. J. Angele, and York Sure (eds.). *Evaluation of Ontology-based Tools (EON'02)*, Proceedings of the 1st International Workshop EON2002, Sigüenza, Spain, CEUR-WS Publication, Vol. 62. `http://CEUR-WS.org/Vol-62/`

2. J. Arpirez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez. WebODE : a Workbench for Ontological Engineering. In *Proc. of the 1st international Conference on Knowledge Capture (K-CAP'01)*, Victoria, Canada, 2001.
3. B. Bachimont, A. Isaac, and R. Troncy. Semantic Commitment for Designing Ontologies: A Proposal. In *Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02)*, Lecture Notes in Artificial Intelligence, Vol 2473, pages 114-121, Sigüenza, Spain, 2002.
4. S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a Reason-able Ontology Editor for the Semantic Web. In *Proc. of KI2001, Joint German/Austrian conference on Artificial Intelligence*, Lecture Notes in Artificial Intelligence, Vol 2174, pages 396-408, Vienna, Austria, 2001.
5. N.F. Noy, R.W. Fergerson, and M.A. Musen. The Knowledge Model of Protégé2000: Combining Interoperability and Flexibility. In *Proc. of the 12th International Conference on Knowledge Engineering and Knowledge Managment (EKAW'00)*, Juanles-Pins, France, 2000.
6. OWL, Web Ontology Language Reference. W3C Candidate Recommendation, 18 August 2003. `http://www.w3.org/TR/owl-ref/`
7. OWL Web Ontology Language XML Presentation Syntax. W3C Note, 11 June 2003. `http://www.w3.org/TR/owl-xmlsyntax/`
8. F. Rastier, M. Cavazza, and A. Abeillé. *Sémantique pour l'analyse*. Masson, Paris, France, 1994.
9. RDF, Ressource Description Framework Primer. W3C Working Draft, 05 September 2003. `http://www.w3.org/TR/rdf-primer/`
10. RDF Schema, RDF Vocabulary Description Language 1.0. W3C Working Draft, 05 September 2003. `http://www.w3.org/TR/rdf-schema/`
11. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer and D. Wenke. OntoEdit: Collaborative Ontology Engineering for the Semantic Web. In *Proc. of the 1st International Semantic Web Conference 2002 (ISWC 2002)*, Lecture Notes in Computer Science, Vol 2342, pages 221-235, Sardinia, Italia, 2002.
12. XSLT, XSL Transformations W3C Recommendation, 16 November 1999. `http://www.w3.org/TR/xslt`