Ontology translation approaches for interoperability: a case study with Protégé-2000 and WebODE

Oscar Corcho, Asunción Gómez-Pérez

Ontological Engineering Group. Departamento de Inteligencia Artificial.
Facultad de Informática. Universidad Politécnica de Madrid.
Campus de Montegancedo, s/n. 28660 Boadilla del Monte. Madrid (Spain)
{ocorcho,asun}@fi.upm.es

Abstract. We describe four ontology translation approaches that can be used to exchange ontologies between ontology tools and/or ontology languages. These approaches are analysed with regard to two main features: how they preserve the ontology semantics after the translation process (aka semantic or consequence preservation) and how they allow final users and ontology-based applications to understand the resulting ontology in the target format (aka pragmatic preservation). These approaches are illustrated with practical examples that show how they can be applied to achieve interoperability between the ontology tools Protégé-2000 and WebODE.

1 Introduction

In Computer Science, the term 'interoperability' is defined as the ability to transmit data and exchange information between systems whilst allowing each system to process information independently [18]. If we refer to ontology tools and languages, the term 'interoperability' can be defined as their ability to exchange ontologies without losing knowledge, in such a way that users of the target format (be them human users or applications) can understand correctly the ontology transformed. This complex transformation process is usually known as ontology translation [14].

The ontology translation problem appears when we decide to reuse an ontology (or part of an ontology) using a tool or language that is different from those ones in which the ontology is available. If we force each ontology developer or integrator, individually, to commit to the task of translating and incorporating ontologies to their systems, they will need both a lot of effort and a lot of time to achieve their objectives. Some of the reasons for the high effort needed are:

- The ontology developer may not know the language or tool where the ontology is available.
- The ontology developer may not know in depth the knowledge representation paradigm underlying the language or tool.
- The ontology developer may not be able to translate the ontology to another language or tool without losing knowledge.

Ontology reuse will be highly boosted if we provide ontology translation services among languages and/or tools. In fact, ontology tools provide export and import

services to and from ontology and general-purpose languages and tools, and there are also translation systems between ontology languages, as shown in [12].

However, there are not deep studies about the quality of the translations performed by ontology translation services. Only a few works ([2], [23]) point out some of the problems that appear in ontology translations. Neither are there characterisations of the approaches followed by current translation technology and their impact in knowledge preservation. For instance, the results obtained in the interoperability experiment proposed at the ISWC'03 workshop on Evaluation of Ontology Tools[1] showed that the fact that two tools provide export and import services to and from a common language does not mean that they actually interoperate.

In this paper we propose a characterisation of ontology translation approaches, focusing on the following two aspects: (1) how these approaches can overcome the differences between the knowledge models of the source and target formats without losing semantics in the transformation, and (2) how these approaches allow end users and ontology-based applications to understand the ontology transformed, what can be considered as an important part of pragmatic or interpretation preservation[2].

To better show and compare these approaches, we have implemented several ontology translation systems between the ontology tools Protégé-2000 [20] and WebODE [1]. The objectives of having translators between both tools are manifold:

On the one hand, WebODE ontology developers and applications will benefit from the use of Protégé-2000 services not offered by the WebODE workbench: ontology merge with PROMPT, advanced instance edition with Protégé-2000 forms, ontology visualisation with Jambalaya, etc. On the other hand, Protégé-2000 ontology developers and applications will benefit from the use of WebODE services: ontology documentation using Methontology's intermediate representations, Semantic Web portal automatic generation with ODESeW, Semantic Web services development with ODESWS, etc.

Furthermore, new ontology services can be developed jointly in the future. For instance, one of the translators from WebODE to Protégé-2000 has been used to export the top level ontology of universals used by the WebODE ODEClean service [10], which supports class taxonomy evaluation with the OntoClean method [15], hence reducing the effort to create the OntoClean plug-in for Protégé-2000.

This paper is organised as follows: Section 2 presents a brief overview of the knowledge models of Protégé-2000 and WebODE, so that the examples shown in later sections can be better understood. Section 3 analyses four groups of ontology translation approaches, focusing on semantic and pragmatic preservation. For each group we show examples of ontology translation systems between Protégé-2000 and WebODE, with the travel ontology developed for the ontology modelling experiment proposed in the EKAW'02 workshop on Evaluation of Ontology Tools[3]. Finally, section 4 presents some conclusions and future work.

---

[1] EON2003 (http://km.aifb.uni-karlsruhe.de/ws/eon2003)

[2] We use the term "pragmatics" as proposed by [19], with regard to the relationship between signs and their interpreters.

[3] EON2002 (http://km.aifb.uni-karlsruhe.de/eon2002)

2 Protégé-2000 and WebODE knowledge models

We start describing the ***Protégé-2000 standard knowledge model*** [20], based on the frame-based knowledge model defined by the OKBC protocol [6], combined with first order logic constraints. It contains the following ontology components:

- **Classes** represent entities of the domain. They can be *concrete* or *abstract*, that is, they can have direct instances or not respectively. *Slot constraints* can be attached to the class, and determine constraints on the slot values of the class instances.
- The taxonomic relation **subclass-of** can be defined between classes, allowing multiple classifications. The relation **subslot-of** can be defined between slots.
- **Slots** represent interactions between domain objects or characteristics of class instances. They have *value types*, *minimum* and *maximum cardinalities*, *minimum* and *maximum values* in case of numeric slots, *default values*, *template values*, *constraints* and the *inverse slot*. Slots are defined independently of the classes to which they are attached, and can be attached in two different ways:
  - **Template slots**. They describe properties of the instances of the class to which they are attached or interactions between instances of the class and instances of other classes. They are inherited by the subclasses and instances of the class, where they can be constrained and/or take values.
  - **Own slots**. They describe properties of the class itself or interactions between the class and other classes, taking some values in the class. These values are not inherited by its subclasses nor by its instances. The template slots of a class become own slots in its instances.
- **Facets** define slot constraints. The features of slots defined before (value types, minimum and maximum cardinalities, etc.) are built-in facets in the Protégé-2000 knowledge model. However, new facets can be defined for them.
- **Instances** of classes define individuals in the domain. They contain own slots (which are the template slots of the classes from which they are instances) with their corresponding values. In Protégé-2000 instances belong only to one class.
- **Constraints** are first order logic sentences used to check constraints in the ontology. They are created with PAL (Protégé Axiom Language), which is a superset of first order logic.

One of the main features of the Protégé-2000 knowledge model is that it allows defining **metaclasses**, classes that act as templates for creating other classes. In fact, metaclasses are used to define the own slots of classes in an ontology. Own slots are defined as template slots in the metaclass, and later the classes that are instances of the metaclass inherit the slots as own slots.

The ***WebODE knowledge model*** [7] is also based on a combination of frames and first order logic axioms, and contains the following ontology components:

- **Concepts** represent entities of the domain. They can have *synonyms* and *abbreviations*, and may contain *attributes*. Attributes are similar to slots in Protégé-2000, though in WebODE they cannot be defined independently from a concept but always attached to it. There are two types of attributes:

    - **Class attributes** specify characteristics of the concept. They have a *value type* or *range*, which can be a basic data type (String, Integer, Float, etc.) or any XML Schema datatype [3]; *minimum* and *maximum cardinality*, which constrain the number of values that the attribute may have; and *value(s)*. Numeric class attributes can have a *measurement unit* and *precision*.
    - **Instance attributes** describe properties of the instances of the concept to which they are attached. Hence their value may be different for each concept instance. They have the same facets than class attributes and two additional ones, *minimum value* and *maximum value*, used in numeric attributes. Values inserted for instance attributes are interpreted as explicit values for them.
- **Concept groups** are sets of disjoint concepts, that is, concepts that cannot share subconcepts nor instances. They are used to create disjoint and exhaustive concept partitions. A concept can belong to several concept groups.
- **Built-in relations** are predefined relations in the WebODE's knowledge model. They are divided into three groups: taxonomy relations between concepts (*subclass-of*, *not-subclass-of*), taxonomy relations between groups and concepts (*disjoint-decomposition*, *exhaustive-decomposition*, and *partition*), and mereology relations between concepts (*transitive-part-of*, *intransitive-part-of*).
- **Binary ad-hoc relations** between concepts are characterized by their *name*, the *origin* (source) and *destination* (target) concepts, and their *cardinality*, which establishes the number of destination terms of each origin term through the relation. Cardinality can be restricted to 1 (only one destination term) or N (any number of destination terms). Optionally, we can provide their *algebraic properties*. Ad-hoc relations are similar to Protégé-2000 slots whose value type is a class instance.
- **Constants** are components that have always the same value and can be used in any expression. They have a *value type*, *value* and *measurement unit*.
- **Axioms** and **rules** define *formal expressions* in first order logic.
- **Properties** are used to describe algebraic properties of ad-hoc relations. They are divided in two groups: *built-in properties* (reflexive, irreflexive, symmetric, asymmetric, antisymmetric and transitive), and *ad-hoc user-defined properties*.
- **Imported terms** are components from other ontologies that are included in another ontology. They are described by their *URI*.

Besides, the WebODE's knowledge model supports **instance sets**, which make possible to populate a conceptual model for different applications or scenarios, maintaining independent instantiations of the same conceptual model.

In summary, the knowledge models of both tools are based on a combination of frames and first order logic. In both of them we can represent classes, organized in class taxonomies where multiple classifications are allowed, ad hoc relations between classes, instances and first order logic axioms. However, there are also important differences between them, related to the types of slots that can be defined for classes, the types of relations that can be used to define class taxonomies, etc.

Figure 1 shows an informal comparison of both knowledge models. The components in the intersection of the Venn diagrams are those that can be expressed in both tools (although they usually differ in the term used to refer to them, for

instance, the term "class" in Protégé is equivalent to the term "concept" in WebODE). The components outside the intersection are those with no direct representation in the other knowledge model, although they may be represented using other modelling primitives (e.g., the "subslot-of" relationship in Protégé-2000 can be represented as a first order logic axiom in WebODE).

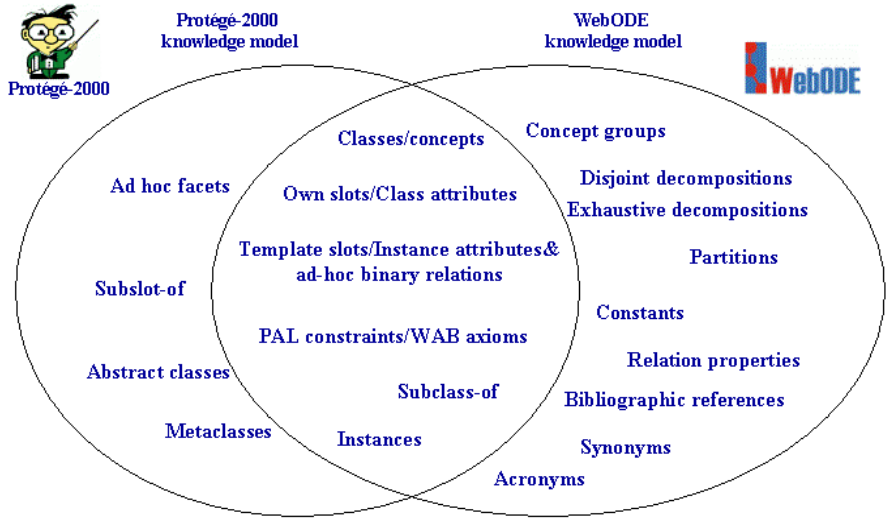

Fig. 1. Informal comparison of the knowledge models of Protégé-2000 and WebODE.

3 A classification of ontology translation approaches

In this section we classify four approaches for ontology translation, according to their features with respect to their semantic and pragmatic preservation properties:

- Does the approach overcome the differences between the knowledge models of the source and target formats without losing knowledge in the transformation? This feature is specially important in cyclic transformations, where we should have exactly the same ontology when transformed back into the original tool.
- Are end users and ontology-based applications able to understand the ontology transformed, as if it was developed with the target ontology tool or language?

We will identify which of the current ontology translation systems have been developed according to each of these approaches, and we will present examples about ontology exchange between the ontology tools WebODE and Protégé-2000.

3.1 Indirect translation, by means of a common interchange language

In the beginning of the 1990s, KIF (Knowledge Interchange Format) [11] was created for exchanging knowledge between heterogeneous knowledge representation systems. The objective of this standardization effort was to reduce the number of

translators needed to achieve interoperability among a heterogeneous group of formats (from $O(n^2)$ to $O(n)$ translators for a number n of formats). The translation process only consists in exporting an ontology from the source format to the exchange language and importing the code obtained to the target format.

Recently, the World Wide Web Consortium (W3C) has proposed three language recommendations, namely RDF [17] and RDF Schema [5], whose combination is usually known as RDF(S), and OWL [9]. These languages have been created with the purpose of becoming widely used for representing knowledge in the Semantic Web. Consequently, they can be also considered as potential common exchange languages, although this is not their main objective.

Ontology tools like the Ontolingua Server and OntoSaurus are able to import from and export to KIF, and WebOnto is able to export to Ontolingua (which in its turn can be transformed directly to KIF by the Ontolingua Server). With regard to the W3C languages, [12] show that most of the existing ontology tools are able to import from and export to RDF(S) and OWL. However, this does not mean that they can really interoperate. We have already mentioned some of the results obtained in the EON2003 workshop, where this type of interoperability was experimented. The main reasons for the lack of interoperability are:

- Common interchange formats normally allow representing the same knowledge in many different ways. Hence, translations to and from interlinguas are usually written with regard to a specific format, making knowledge exchange difficult.
- Interoperability with interlinguas has been only proved with the same origin and target formats, but not between different source and target formats.
- In the case of RDF(S) and OWL, their standard knowledge models are not expressive enough to represent most of the knowledge that can be represented with other ontology languages and tools. However, their knowledge models can be extended to represent other pieces of knowledge, by means of metaclasses.

Therefore, **knowledge is not usually preserved** in the transformation and the resulting ontologies in the target format **are not always easy to understand**, since they incorporate expressions related to the source format. For further examples of these interoperability problems, we recommend [22], [2], [16], and [8].

Example: achieving interoperability between WebODE and Protégé-2000 using RDF(S) and OWL as common interchange formats. Protégé-2000 and WebODE are able to export to and import from different ontology languages, such as RDF(S), and OWL, and to and from other formats that are not specific for ontology implementation, such as XML [4] and UML. For the last two ones, these tools are not able to exchange ontologies, since they use different tag sets (the case of XML) and different types of application-specific UML. Hence we will consider only the indirect translation with RDF(S) and OWL.

Figure 2 shows an example of the result of transforming an ontology from WebODE into Protégé-2000, using RDF(S) as a common interlingua. The classes and their slots (be them originally WebODE instance attributes or ad-hoc relations), and the class taxonomy are preserved. However, some knowledge is lost in this transformation. For instance, the standard knowledge model of RDF(S) does not allow specifying the cardinalities of attributes (called properties in this language).

Therefore, if we transform an ontology from WebODE into RDF(S), and from RDF(S) into Protégé-2000, the information regarding cardinality restrictions is lost (the default cardinality restrictions in Protégé-2000 are 0 for minimum cardinality and multiple for maximum cardinality).

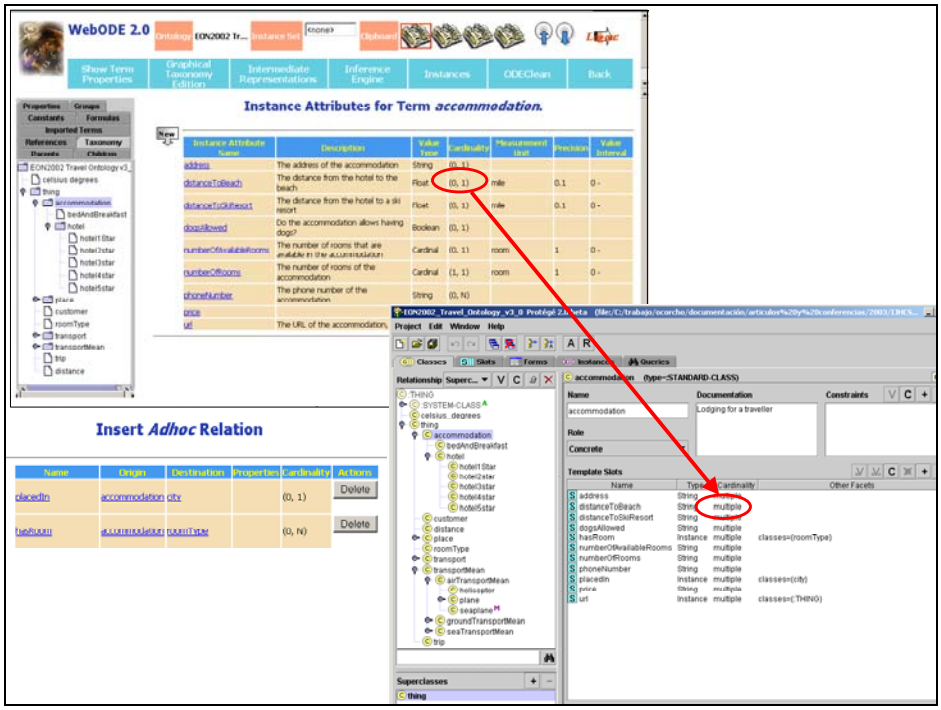

Fig. 2. Result of the transformation of the EON2003 travel ontology from WebODE to Protégé-2000, with RDF(S) as a common interchange format.

Other knowledge that cannot be expressed in the standard knowledge model of RDF(S) is the disjoint and exhaustive knowledge in class taxonomies, formal axioms, bibliographic references to ontology components, etc.

The previous problem appears because the WebODE export service to RDF(S) does not export the knowledge that cannot be represented in the standard knowledge model of RDF(S). In order to avoid these knowledge losses in the export process, the metaclass facilities of RDF(S) could have been used to represent the knowledge model of WebODE. In fact, this is exactly what the Protégé-2000 RDF Storage backend does [21] when storing Protégé-2000 ontologies in RDF(S): it creates metaclasses for some of the components of the Protégé knowledge model that have not a direct translation into the standard knowledge model of RDF(S).

However, all this extra knowledge added to the standard knowledge model of RDF(S) cannot be usually imported into other RDF-aware tools different than Protégé-2000. They will only "understand" and import those parts of the ontology that are implemented with the standard RDF(S) knowledge model.

3.2 Direct transformation restricted to the standard knowledge model of the target format

Transformations in this group consist in identifying the mappings between the standard knowledge models of the source and target formats, and executing these mappings. The knowledge components of the source format that have not a direct correspondance in the target format are not transformed, even in the case that the target format does provide means for extending its standard knowledge model.

This type of transformation is mainly driven by the objective of making **the transformed ontology easily understood by users and applications** aware of the target format. This is the reason why the standard knowledge model of the target format is not extended. As a consequence of this decision, **some knowledge is usually lost in the transformation**. This occurs when the standard knowledge model of the target format does not include the source one, that is, when the target format does not express all the knowledge that can be represented in the source format.

In this group of transformations we can include most of the ontology translation systems currently available in ontology tools: the WebODE import and export services from and to RDF(S), DAML+OIL, OWL, CARIN, FLogic, and UML; the OntoEdit import and export services from and to RDF(S), DAML+OIL, and FLogic. KAON import and export services to RDF(S), etc.

Example: achieving interoperability between WebODE and Protégé-2000 by direct translation restricted to the target format standard knowledge model. With this transformation approach, we transform all the components from the WebODE domain ontology that have a direct representation in Protégé-2000. For example, WebODE concepts can be transformed into Protégé-2000 classes, WebODE instance attributes and ad hoc relations into Protégé-2000 slots, WebODE class attributes into Protégé-2000 own slots (with the use of some metaclasses), WebODE axioms to PAL constraints, etc. Not only this one-to-one transformations will be done, but also more complex ones, such as transforming WebODE partitions as Protégé-2000 class taxonomies creating *subclass-of* relationships between the classes in the partition and the superclass, declaring the superclass of the partition as abstract, and creating PAL constraints that ensure the disjointness between classes.

Some knowledge is lost in the transformation, such as algebraic properties of relations, synonyms and acronyms, bibliographic references, etc.

Figure 3 shows the result of transforming the travel ontology to Protégé-2000 following this proposal. The figure shows that the concepts *hotel1Star*, *hotel2Star*, *hotel3Star*, *hotel4Star* and *hotel5Star*, which form a partition of the class *hotel* in WebODE, are transformed into Protégé classes that are subclasses of the class *hotel*, and have constraints to ensure that they are disjoint with the other ones in the partition. Besides, the class *hotel* is defined as an abstract class in Protégé-2000, that is, it cannot have direct instances. It also shows that the metaclass *hotelTypes* has been created. This metaclass has the template slot *numberOfStars* attached, which is converted to an own slot in the classes *hotel1Star*, *hotel2Star*, etc.

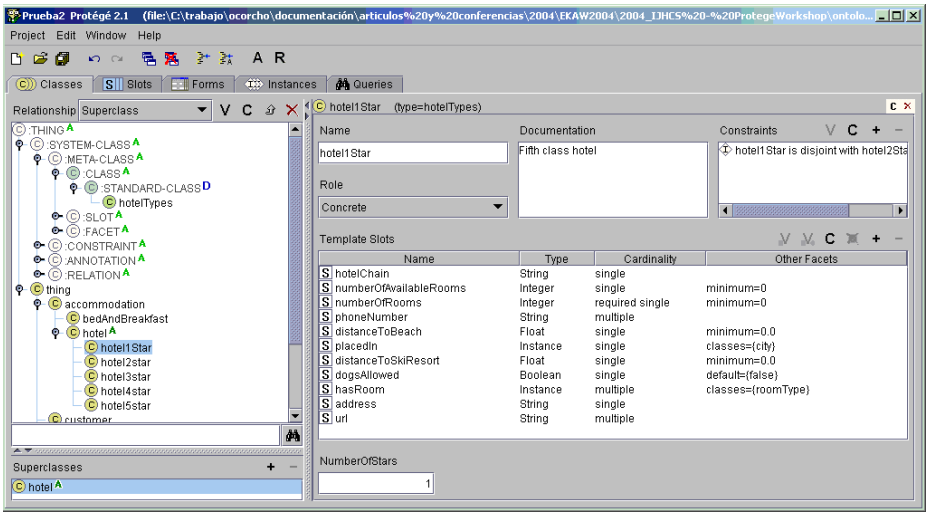

Fig. 3. Result of the transformation of the EON2003 travel ontology from WebODE to Protégé-2000, using only the standard knowledge model of Protégé-2000.

The inverse transformation, that is, from Protégé-2000 to WebODE, is similar. It consists in determining the direct correspondances between the knowledge models of Protégé-2000 and WebODE (already identified previously) and performing the transformations of only those components that have such a direct correspondance.

3.3 Direct transformation by instantiation of the source format KR ontology in the target format

Transformations in this group are performed following two steps. The first step consists in creating the KR ontology of the source format in the target one, with the standard knowledge modelling components of the target format. Basically, it consists in developing that KR ontology as if it was a domain ontology of the target format. Once that this KR ontology has been created, the second step of the translation process consists in instantiating it according to the contents of the ontology to be transformed. For instance, a concept in the source format is transformed into an instance of the class *Concept* in the target format; an instance in the source format is transformed into an instance of the class *Instance* in the target format, and is connected to one or several instances of the class *Concept* by means of the relation *isInstanceOf*; etc.

In summary, the only translation decisions taken into account in this transformation are related to how to transform the source ontology components to instances of the KR ontology created in the target format. That is, no real translation decisions are performed and the transformation process is extremely simple. The main advantage of this transformation proposal is that it is fairly easy to implement, since we do not have to use different ontology components in the target tool but only change the underlying format of the original ontology.

With regard to the two aspects that we are analysing for each approach, we must take into account that if the source KR ontology is correctly modelled in the target format, **the knowledge represented in the original ontology is preserved**. However, end users and applications will **not be able to understand correctly the ontology transformed**, since the components that represent domain knowledge (basically instances of concepts and of relations) are mixed with the components that represent the KR ontology of the source format.

In this group we can include some import and export formats of ontology tools, such as the WebODE import and export services from and to Prolog, XML and Java, the import and export services of OntoEdit from and to OXML, the ones between Protégé-2000 and XML, and XML Schema, etc. These languages are used mainly as a syntax for exchanging ontologies, but not as knowledge representation languages with their characteristics.

Example: achieving interoperability between WebODE and Protégé-2000 by instantiating the WebODE knowledge model in Protégé-2000. As described above, with this approach we follow a two-step process: (1) create the WebODE KR ontology in Protégé-2000; (2) transform all the components of the WebODE domain ontology according to that KR ontology, that is, instantiate the WebODE KR ontology with the knowledge from the source domain ontology.

The WebODE KR ontology contains 14 classes, which represent the ontology components of the WebODE knowledge model (*:WebODEComponent*, *:Concept*, *:Attribute*, *:InstanceAttribute*, etc.), and 36 slots, which represent the relationships between the previous components: *:name*, *:documentation*, *:hasAttribute*, *:hasDomain*, *:hasRange*, *:datatype*, *:maxCardinality*, etc. The ontology also contains 32 PAL constraints that model restrictions of the WebODE knowledge model.

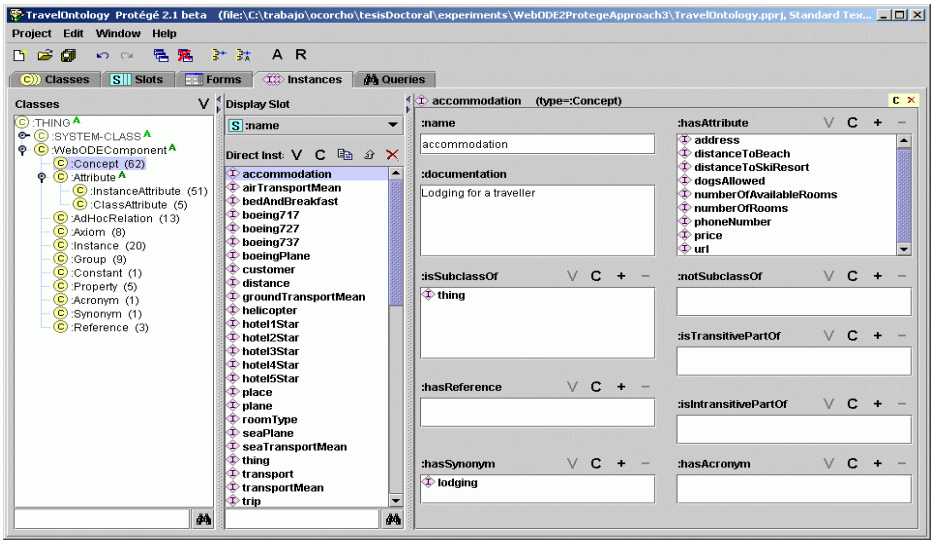

Fig. 4. Result of the transformation of the EON2003 Travel ontology from WebODE to Protégé-2000 as an instantiation of the WebODE KR ontology.

Figure 4 shows a screenshot of the class taxonomy of the WebODE KR ontology modelled in Protégé-2000. It also shows the details of the It shows also the concept *accommodation*, with its documentation, the classes of which it is a subclass, the synonyms, etc. This concept is transformed into Protégé-2000 as an instance of the class *:Concept* in Protégé-2000.

To transform the ontology back to WebODE, the translation decisions are also easy to implement. They only consist in converting the instances of each class in this ontology to the corresponding component in WebODE.

The inverse case (translating from Protégé-2000 to WebODE, and then back to Protégé-2000) is similar to this one: we create the Protégé-2000 KR ontology in WebODE and transform all the components of the Protégé-2000 ontology according to it. To achieve complete interoperability between WebODE and Protégé-2000 in both directions we need four translation systems: two for the cycle WebODE-Protégé-2000-WebODE, and another two for Protégé-2000-WebODE-Protégé-2000.

3.4 Direct transformation by an extension of the standard knowledge model of the target format

This group of transformations propose to transform as much knowledge as possible using the components of the target format's standard knowledge model (as described in section 3.2), so that **the knowledge transformed can be easily understood** and dealt with by human users and applications. Besides, it proposes **to preserve the knowledge that has not been transformed**, so that they can be recovered in case that the ontology is transformed back to the original format (as described in section 3.3). This proposal avoids mixing the domain ontology components transformed with the knowledge modelling components used for knowledge preservation.

To achieve this twofold objective, this approach proposes to implement part of the source format's KR ontology in the target format, by means of meta-knowledge (which is usually expressed with metaclasses, annotation properties, etc.). The KR ontology implemented contains only the ontology components that cannot be represented directly in the standard knowledge model of the target format, so that they are only used in case that some knowledge of the original ontology cannot be directly transformed to the target format. In summary, following this approach the translation of an ontology consists in performing the following steps:

- Create part of the KR ontology of the source format in the target format, by means of meta-knowledge, so as to represent the knowledge modelling components of the source format that have not a direct correspondence with the knowledge modelling components of the target format. Meta-knowledge can be represented by means of metaclasses, annotations, structured natural language documentations, etc.
- Whenever there is a direct correspondence between the source format's ontology components and the target format's ones, transform the component of the original ontology to its corresponding components in the target format.
- Transform all the knowledge that was not transformed previously, according to the partial KR ontology created in the first step.

To use this approach, the target format must allow representing fragments of KR ontologies in such a way that the domain ontology components transformed are not mixed with the knowledge modelling components used for knowledge preservation.

Inside this group we can include the Protégé-2000 backend for RDF(S), where specific pieces of knowledge of the Protégé-2000 knowledge model are exported to that language [21], the Protégé-2000 plug-in for OWL, which extends the Protégé-2000 standard knowledge model with OWL-specific features, and the current WebODE import and export services to Protégé-2000.

Example: achieving interoperability between WebODE and Protégé-2000 by an extension of the Protégé-2000 standard knowledge model. As described above, this approach consists first in creating part of the WebODE KR ontology in Protégé-2000 by extending the Protégé-2000 standard metamodel.

This KR ontology contains six metaclasses (*:WebODEConcept*, *:WebODESlot*, *:WebODEAttribute*, *:WebODEAdHocRelation*, *:WebODEPredefinedSlots*, and *:WebODEPredefinedFacet*), three subclasses of the class *:PAL-CONSTRAINT* (*:WebODEDisjointConstraint*, *:WebODEPropertyConstraint*, and *:WebODEAxiom*), and five subclasses of the class *:THING* (*:WebODEReference*, *:WebODESynonym*, *:WebODEAbbreviation*, *:WebODEConstant*, and *:WebODEProperty*).

Figure 5 shows the details of the metaclass *:WebODEConcept*, which extends the metaclass *:STANDARD-CLASS*, used by default for creating Protégé-2000 classes. The extension consists in adding new WebODE-specific slots to store information about classes such transitive and intransitive parts of, not subclass of, and bibliographic references, abbreviations, and synonyms.

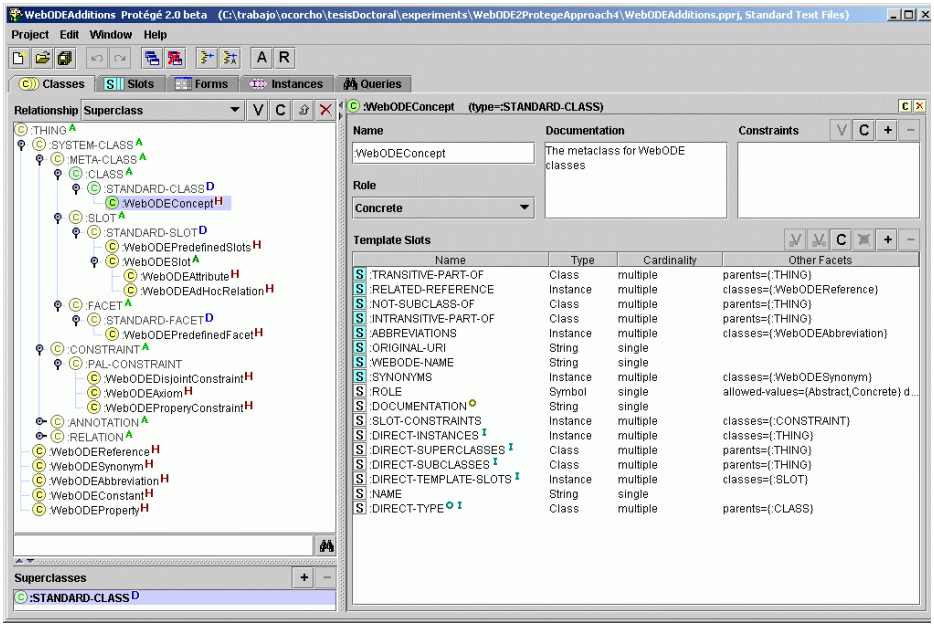

Fig. 5. WebODE KR ontology defined as an extension of the Protégé-2000 standard knowledge model.

Two other slots are attached to this metaclass, so as to recover the original ontology in cyclic transformations: the original URI and name of the WebODE concept, since the class name may have suffered transformations in the translation process. The rest reuses the standard slots for standard classes in Protégé-2000.

Once this ontology has been generated, all the WebODE ontology components that have a direct representation in the standard knowledge model of Protégé-2000 are transformed. Then, the rest of the WebODE ontology components that have not a direct representation in the previous knowledge model are transformed. This transformation consists in creating instances of the WebODE KR ontology described above, hence preserving the knowledge that could have been lost in the translation when the ontology is transformed back to WebODE.

Figure 6 shows a screenshot of the travel ontology translated using this approach. The metaclass *:WebODEConcept* has a subclass *hotelTypes*, used as the metaclass for the classes *hotel1Star*, *hotel2Star*, *hotel3Star*, *hotel4Star*, and *hotel5Star*, since they all have an own slot *numberOfStars* whose value is 1, 2, 3, 4, and 5, respectively. The rest of classes in this ontology are instances of the metaclass *:WebODEConcept*.

In contrast with the solution presented in the previous section, the classes and class taxonomy of the ontology can be easily seen and understood. The figure shows the details of the class *hotel1Star*, with its template slots and the own slot *numberOfStars*, which come from the instance attribute and class attribute definition of WebODE. There are also other own slots that do not usually appear in Protégé-2000 ontologies: *:ORIGINAL-URI*, *:WEBODE-NAME*, *:SYNONYMS*, *:ABBREVIATIONS*, etc. These slots are used to include the extra knowledge from WebODE, as discussed above.

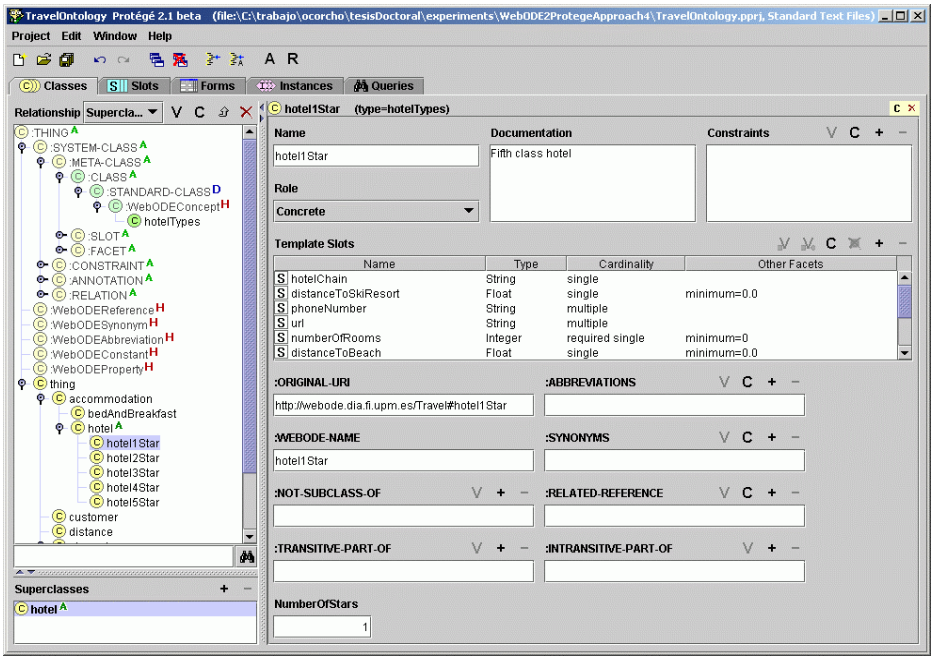

Fig. 6. Result of transforming the EON2003 Travel ontology from WebODE to Protégé-2000 with the Protégé-2000 standard knowledge model and part of the WebODE KR ontology.

The main drawback of this approach is that an end-user who is not acquainted with the knowledge model of WebODE will not be able to understand what the terms *:WebODEReference*, *:WebODESynonym*, etc., mean, since this solution is merging the modeling of the domain ontology with the modeling of the KR ontology. We need a last step in this process, to ensure that the new components related to the WebODE KR ontology are not shown to Protégé-2000 users.

If we use the Protégé-2000 hiding-class functionality and remove some components from the forms showed to users, end users do not need to know anything about how to model ontologies with WebODE and can use the ontology using the common modeling conventions provided by Protégé-2000. In the transformation back to WebODE, this hidden knowledge is transformed without problems.

This proposal meets the two objectives of semantic and pragmatic preservation: it maintains all the knowledge in the transformation and allows end users and applications to understand easily the resulting ontology.

A similar approach can be applied to transform ontologies from Protégé-2000 into WebODE: the new components are not shown to WebODE users since they are instances of the WebODE KR ontology and are not accessed directly from the resulting ontology. However, they are also available for performing the transformation back to Protégé-2000.

3.5 Conclusion and future work

Figure 7 compares the four approaches presented in this paper according to the two criteria used in our description. The horizontal axis represents the amount of knowledge preserved in the transformation (semantic preservation), while the vertical axis represents the legibility preserved in the transformation (pragmatic preservation). We do not aim at providing quantitative measures about these amounts, but we just provide approximate qualitative measures for them. Besides, not all the source and target format pairs behave in the same way with respect to this preservation.

As shown in the figure, the second and third approaches have extremely different behaviours. The former is devoted to preserve pragmatics, with no guarantees about the semantics of the knowledge transformed; the latter is devoted to preserve semantics, with even less guarantee about the preservation of the pragmatics of the ontology transformed. The most interesting approach seems to be the fourth one, which can maintain most of both properties. Indirect translations through common interchange languages are less precise and depend on the quality and standardization of the transformations to the interlingua.

With regard to the implementation effort required to carry out each approach, the results are different. The use of common interchange formats does not usually require much implementation effort, since in most of the cases there are already ontology translation systems implemented for such tasks. With respect to the second and third approaches, they require more or less the same amount of effort: the effort required in the second approach is mainly related to the identification of mappings between the knowledge components of the source and target formats and the implementation of such mappings, while in the third approach the effort is mainly related to the creation of the source KR ontology in the target format (the transformation is usually easy).

Finally, the fourth approach requires more effort, since it combines the two previous ones. However, it provides better preservation properties than the other ones.

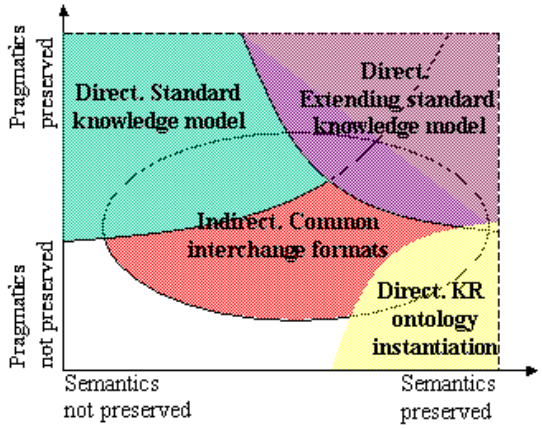

Fig. 7. Comparison of transformation approaches according to their preservation properties.

Our future work will be devoted to exploring other possibilities of interoperability between formats that do not rely on translating all the components of the ontology each time that we need an ontology in a specific format. Ontologies can evolve in their source and target formats, due to reengineering processes [13], and in the case that we are reusing an ontology that has evolved in its original format and in its target format, it could be interesting to import only the ontology components that have changed, and to apply ontology merge techniques in order to detect which are the changes that affect the reengineered ontology.

Acknowledgements

This work is supported by the IST project Esperonto (IST-2001-34373).

References

1. Arpírez JC, Corcho O, Fernández-López M, Gómez-Pérez A (2003) *WebODE in a nutshell*. AI Magazine 24(3):37-48
2. Barley M, Clark P, Williamson K, Woods S (1997) *The neutral representation project*. In: Farquhar A, Grüninger M (eds) AAAI-97 Spring Symposium on Ontological Engineering, Stanford, California. AAAI Press
3. Biron PV, Malhotra A (2001) *XML Schema Part 2: Datatypes*. W3C Recommendation. http://www.w3.org/TR/xmlschema-2/
4. Bray T, Paoli J, Sperberg-McQueen CM, Maler E (2000) *Extensible Markup Language (XML) 1.0*. W3C Recommendation. http://www.w3.org/TR/REC-xml
5. Brickley D, Guha RV (2004) *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. http://www.w3.org/TR/PR-rdf-schema
6. Chaudhri VK, Farquhar A, Fikes R, Karp PD, Rice JP (1998) *Open Knowledge Base Connectivity 2.0.3*. Technical Report. http://www.ai.sri.com/~okbc/okbc-2-0-3.pdf

7. Corcho O, Fernández-López M, Gómez-Pérez A, Vicente O (2002) *WebODE: an Integrated Workbench for Ontology Representation, Reasoning and Exchange*. In: Benjamins R, Gómez-Pérez A (eds) 13th International Conference on Knowledge Representation and Knowledge Management (EKAW2002). Sigüenza, Spain.
8. Corcho O, Gómez-Pérez A, Guerrero-Rodríguez DJ, Pérez-Rey D, Ruiz-Cristina A, Sastre-Toral T, Suárez-Figueroa MC (2003c) *Evaluation experiment of ontology tools' interoperability with the WebODE ontology engineering workbench*. In: Sure Y, Corcho O, Angele J (eds) ISWC2003 Workshop on Evaluation of Ontology Tools (EON2003). Sanibel Island, Florida. CEUR Workshop Proceedings 87. (*http://CEUR-WS.org/Vol-87/*)
9. Dean M, Schreiber G (2004) *OWL Web Ontology Language Reference.* W3C Recommendation. *http://www.w3.org/TR/owl-ref/*
10. Fernández-López M, Gómez-Pérez A (2002) *The integration of OntoClean in WebODE*. In: Angele J, Sure Y (eds) EKAW'02 Workshop on Evaluation of Ontology-based Tools (EON2002), Sigüenza, Spain. CEUR Workshop Proceedings 62:38–52. Amsterdam, The Netherlands. *http://CEUR-WS.org/Vol-62/*
11. Genesereth MR, Fikes RE (1992) *Knowledge Interchange Format. Version 3.0. Reference Manual*. Technical Report Logic-92-1. Computer Science Department. Stanford University, California, *http://meta2.stanford.edu/kif/Hypertext/kif-manual.html*
12. Gómez-Pérez A, Fernández-López M, Corcho O (2003) *Ontological Engineering*. Springer-Verlag. London, UK
13. Gómez-Pérez A, Rojas MD (2000) *Ontological Reengineering and Reuse*. In: Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00). Juan-Les-Pins, France. Springer-Verlag, Lecture Notes in Artificial Intelligence (LNAI) 1937, Berlin, Germany, pp 139–156
14. Gruber TR (1993) *A translation approach to portable ontology specification*. Knowledge Acquisition 5(2):199–220
15. Guarino N, Welty C (2000) *A Formal Ontology of Properties*. In: Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00). Juan-Les-Pins, France. (Lecture Notes in Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp 97–112
16. Isaac A, Troncy R, Malaise V (2003) *Using XSLT for Interoperability: DOE and The Travelling Domain Experiment*. In: Sure Y, Corcho O, Angele J (eds) ISWC2003 Workshop on Evaluation of Ontology Tools (EON2003). Sanibel Island, Florida. CEUR Workshop Proceedings 87. (http://CEUR-WS.org/Vol-87/), pp 92-102
17. Lassila O, Swick R (1999) Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation. http://www.w3.org/TR/REC-rdf-syntax/
18. Merriam-Webster online. http://www.m-w.com/home.htm
19. Morris C (1938) *Foundations of the Theory of Signs*, in Carnap R et al (eds.) International Encyclopedia of Unified Science, 2:1, Chicago: The University of Chicago Press.
20. Noy NF, Fergerson RW, Musen MA (2000). *The knowledge model of Protege-2000: Combining interoperability and flexibility*. In: Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00). Juan-Les-Pins, France. Springer-Verlag, LNAI 1937, Berlin, Germany.
21. Noy NF, Sintek M, Decker S, Crubézy M, Fergerson RW, Musen MA (2001) *Creating Semantic Web Contents with Protégé-2000*. IEEE Intelligent Systems and their applications. March/April. pp:60-71.
22. Raschid L, Vidal ME (1996) *A KIF for Multiple F-Logic databases*, Technical report, Institute for Advanced Computer Studies University of Maryland
23. Valente A, Russ T, MacGregor R, Swartout W (1999) *Building and (Re)Using an Ontology of Air Campaign Planning*. IEEE Intelligent Systems & their applications 14(1):27–36