# Case Study: Using Protégé to Convert the Travel Ontology to UML and OWL

Holger Knublauch

Stanford Medical Informatics
Stanford University
MSOB x-215
251 Campus Drive
Stanford, CA 94305-5479
holger@smi.stanford.edu
WWW home page: http://www.knublauch.com

**Abstract.** Our goal was to evaluate the import/export capabilities of Protégé between various ontology file formats. As a starting point, we chose the Travel ontology used for the Protégé experiment from the previous EON workshop. We exported this into UML, from where we could import most of the ontology into the mainstream software development tool Poseidon. Furthermore, we exported the ontology into OWL. The resulting OWL file could be processed by the OWL Species Validator. All transformations maintained the structure of the ontology without problems but could not handle all of the model semantics correctly.

## 1 Introduction

Protégé (http://protege.stanford.edu) is one of the most widely used ontology editors with currently about 10,000 registered users. Its extensible open-source platform supports several ontology file formats including CLIPS (Protégé's native format), various XML dialects, databases, DAML+OIL and RDF(S). Very recently, storage plugins for the Unified Modeling Language (UML) and the Web Ontology Language (OWL) have been added. Both plugins are not complete yet and will evolve during the following months.

This document reports on a simple experiment with the UML and OWL Plugins. We wanted to test whether Protégé can convert a given ontology into these formats and to get an idea of which information are getting lost during conversion. Our starting point is the Travel Ontology developed by Natasha F. Noy as described in her contribution to the previous EON workshop. A screenshot of this ontology (displayed in Protégé) is shown in figure **??**.

The experiment was performed using the most recent alpha release of Protégé 2.0 (build 42). Older versions (starting with version 1.8) would expose the same behavior for the UML conversion. However, these versions do not support the OWL Plugin.

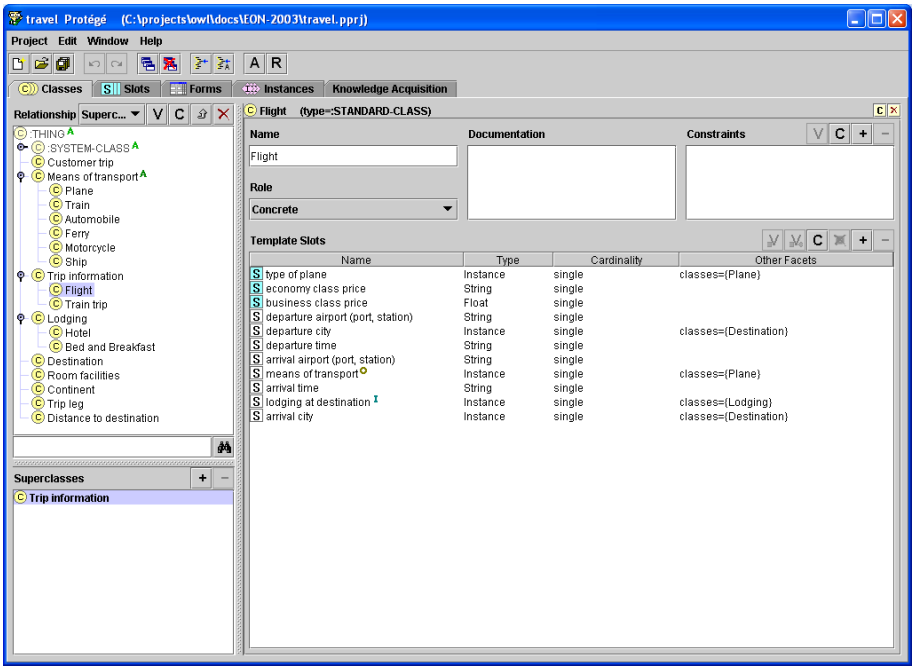


**Fig. 1.** The original ontology (CLIPS format) edited with Protégé.

## 2 UML Export and Import

UML is one of the best known modeling languages for real-world projects. There have been several attempts to exploit UML for ontology modeling so that mainstream tools can be used for knowledge modeling. The Object Management Group (OMG) has recently issued a call for proposals for a UML-based ontology language which will boost interest in ontology design among software developers. In order to provide some interoperability between Protégé and UML tools, the UML Plugin has been developed in February 2003. Since then, it has been adopted into routine use by many users.

The Protégé knowledge model (OKBC) and UML allow very similar constructs. Most obviously, the following conversions exist:

- UML classes can be compared to OKBC classes
- UML objects are similar to OKBC instances
- UML attributes and relationships are comparable to OKBC slots

However, there is a significant area of language elements that are incompatible. Most notably, Protégé supports a native constraint language called PAL, whereas UML uses its Object Constraint Language (OCL). Both have a similar structure but a converter does not exist yet.

Another difference is that Protégé supports generic facet overloading, which means that you can redefine slot properties (such as value type, cardinality and default values) for certain classes. This reflects a major difference between UML and OKBC, namely that in OKBC, slots are first-class elements and can exist without being assigned to a class, whereas UML attributes and relationships must be assigned to classes. Protégé's UML Plugin is able to handle this difference. For example, it creates multiple copies of an attribute if a slot is attached to more than one class. It fails however with complex facet overloads, because a comparable concept does not exist in UML.

Other differences between UML and OKBC include the handling of meta-classes (which is much more flexible in Protégé) and support for instances. Although UML officially has the concept of Object Diagrams, few tools support it properly, and so the UML Plugin does not export instances. There is however no reason why this should not be supported in future versions.

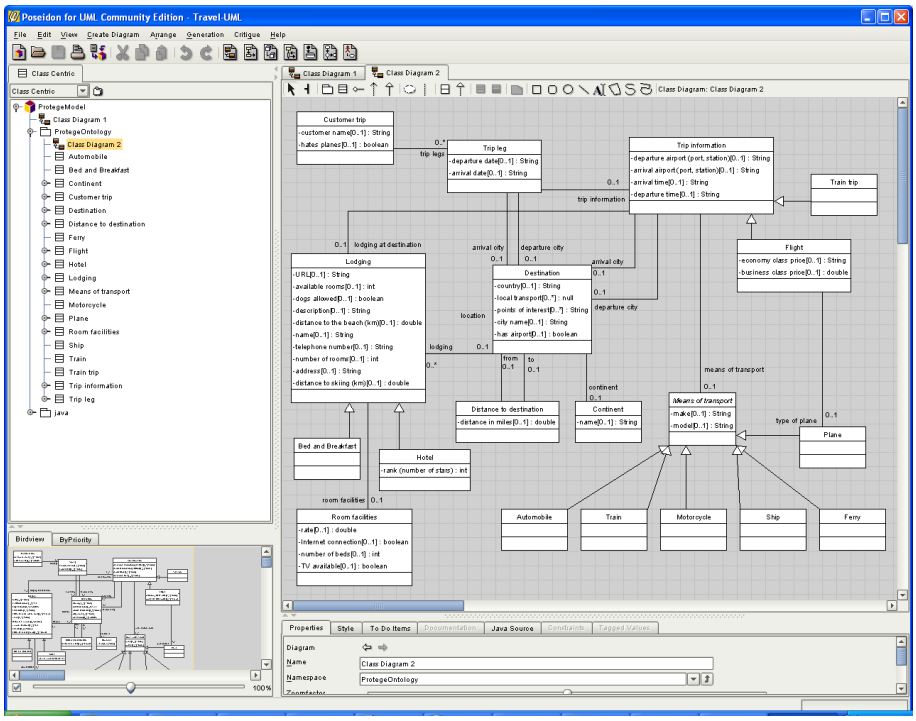


**Fig. 2.** The ontology exported with Protégé in UML format opened with Poseidon for UML.

For the given travel ontology, most of the structural information from the ontology could be preserved. As shown in figure **??**, the resulting UML file (in

XMI format) could be loaded with the well-known UML modeling tool Poseidon. Since not all CASE tools support the XMI standard equally well, it might not be possible to load UML files generated with Protégé into all tools. This shortcoming is however due to different interpretations of UML/XMI standards by third-party tools, while Protégé supports the official UML specification.

Note that Protégé can also re-import UML files that have been changed with an external tool. In this step it will also combine multiple namesake attributes into a single slot, etc.

The following information got lost during the translation:

- PAL Constraints
- Facet overloads (there were 4 of them in the original ontology)

The allowed values of symbol slots are exported correctly in the XMI file, but not displayed by the UML tool so that the datatype of some attributes is "null". This is a bug in Poseidon.

While UML and OKBC each provide different modeling elements, they are both extensible and thus allow for a complete round-trip mapping. Protégé's generic metamodeling architecture can be used to define new metaclasses which capture UML-specific items such as methods and OCL expressions. This has been partially implemented so that Protégé can also be used to define class methods. UML has a number of extension mechanisms, such as stereotypes and tagged values, which can be used to store Protégé-specific data for round-tripping.

The rather awful problem with the current UML specification (before 2.0) is that there is no standard exchange format for diagrams. This means that users need to re-layout their class diagrams each time when it has been changed.

## 3 OWL Export and Import

Work on the OWL Plugin for Protégé started in April 2003 and is not finished yet. Therefore the following results are preliminary (and might have changed at the workshop time). Protégé relies on the Jena API, a leading Java-based API for OWL and RDF. Since this software is also still in alpha state, not all features are implemented yet.

As shown in figure **??**, Protégé and OWL each support constructs that are not available in the other. A major difference is that OWL supports arbitrary class descriptions, whereas Protégé only knows primitive named classes. We have extended Protégé's metamodel to express these additional language elements. More details on this mapping can be found on the OWL Plugin web site (http://protege.stanford.edu/plugins/owl).

The current version of the OWL plugin allows to load arbitrary OWL (DL) files into Protégé. Some elements of OWL Full, especially metaclasses, can be represented. Protégé maintains a copy of the OWL model using the Jena API, and changes in the Protégé model are synchronized with the OWL objects. This technology ensures that all language elements that Protégé does not support in

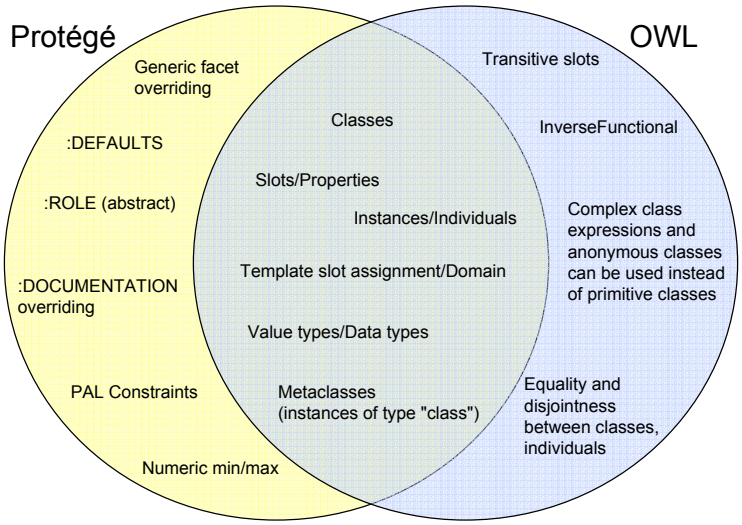


**Fig. 3.** The language elements of Protégé and OWL in comparison.

its own metaclass hierarchy at least remain untouched when saved back to a file. Editing OWL files with Protégé is therefore lossless.

The example travel ontology could be converted into Protégé without problems. As shown in figure **??**, facet overloads are automatically converted into OWL restrictions (here: An allValuesFrom restriction). The only information that currently gets lost is Protégé-specific elements such as PAL constraints.

The OWL files created by Protégé obey the recent OWL standard specification and can be loaded by external OWL tools such as the OWL Species Validator. However, due to the lack of other ontology tools with OWL support, we could not seriously test advanced issues such as round-tripping between tools.

## 4 Discussion and Future Work

The simple case studies show that Protégé is a suitable platform for interchanging models in standard languages such as UML and OWL. Both languages play a central role in two huge communities that are traditionally not counted as ontology builders: Mainstream Software Engineering and the Semantic Web, respectively. The wide adoption of Protégé's support for these languages has

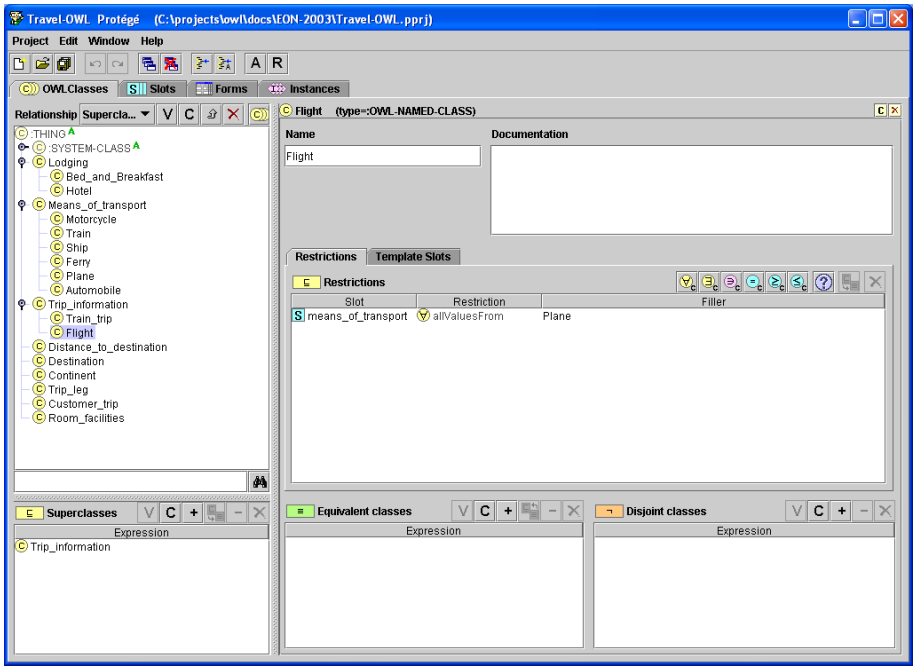


**Fig. 4.** The ontology in OWL format edited with Protégé.

shown us how important they are and that ontology construction could play a much more important role in these communities.

Both examples also demonstrate the flexibility of the OKBC knowledge model. OKBC provides a very flexible metamodeling architecture that can be easily extended to capture other languages than those natively supported by Protégé. With an extended metamodel in place, one only needs to adapt the user interface to get a custom-tailored modeling tool for almost any language. Several specific editor components have been implemented for OWL.

In support of true round-trip engineering – which is crucial for real world projects – the tools should make sure that one tool's language specific data is not lost when opened with another tool. We have not fully implemented these capabilities due to lack of time. Currently, Protégé-specific information that does not have a direct counterpart in OWL or UML is getting lost. There are however no reasons why this should not be possible in the future.