Crafting Ontologies

From physical freight to machine readable data

Semantic Web data formats were designed to provide a way to describe and define data by using more data. The Web of Linked Data allow publishers to describe data models and data concepts in such a way that they can be linked, described, and queried as if they were part of a single database.

Different tools can be used in order to model physical objects into an RDF compliant vocabulary. This Tech Insight showcases the methodology used by the ONE Record data model and technical experts for creating the ONE Record ontology.

How to easily create ontologies?

Let's say that you understand Semantic Web, RDF and OWL and you feel that these concepts are the best thing since you have discovered how to use hashtags. Even if you follow technical examples and tutorials, you realize that you would never be able to manually create and maintain RDF (Resource Description Framework) nor OWL (Web Ontology Language) in your favourite text editor, by copying and pasting model description from a spreadsheet. Such a process would be long and error prone, and for big vocabularies, it would take hours and hours of copy-pasting.

Fortunately, there are different "non-manual editing" ways to create RDF data and OWL ontology models.

Protégé Tool

In order to create the ONE Record ontology, Protégé tool was selected.

Protégé tool, developed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine, is one of the oldest and most widely deployed ontology modelling tools. It was originally conceived as frame-based modelling tool for rich ontologies in accordance with the Open Knowledge Base Connectivity protocol. Later iterations of Protégé have expanded to include a plug-in that is now widely used for OWL and RDF modelling.

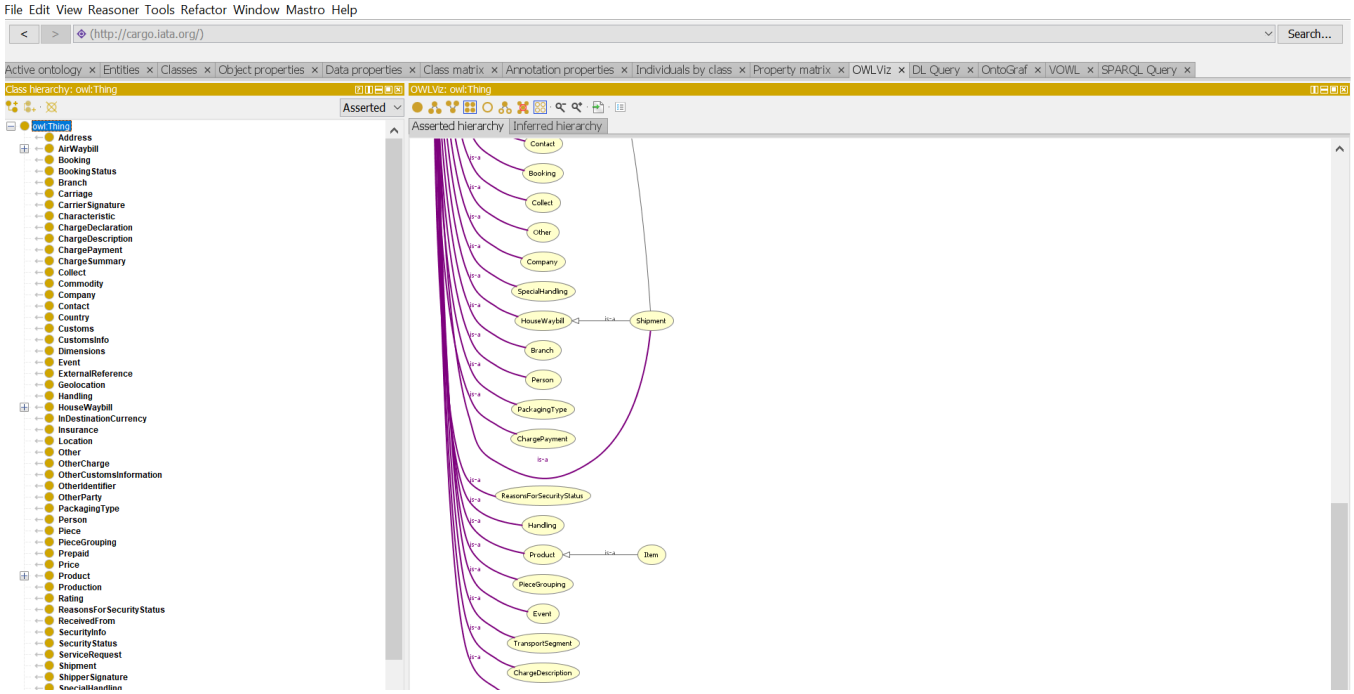
Although Protégé is most widely used in the academic community, its fully featured support for OWL and RDF is making the tool very popular among commercial enterprises as well. Because it is free and open-source, Protégé could be considered a leading ontology editor.

Protégé allows the definition of classes, class hierarchy's variables, variable-value restrictions, and the relationships between classes and the properties of these relationships.

Simple methodology for ontology creation

There is no "correct" way for developing ontologies. We can state that ontology development is following an agile process as it is an incremental and iterative process: we have first defined the rough objects, then we have defined the relations between them and finally we have refined the relations by adding further details such as cardinality. At the moment, the ONE Record ontology does not contain any business process definitions. Business process will be defined in the future iterations of the ONE Record ontology development.

In the following example, we are presenting the high-level steps in creation of an ontology. Protégé website contains several step-by-step guides with screenshots of the user interface for each of these steps. For simplicity purposes, we are not providing these screenshots in the current document.

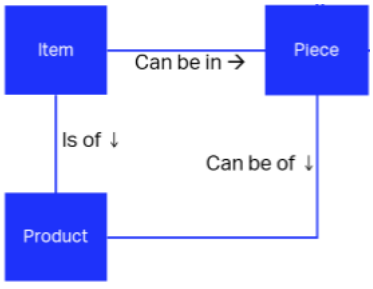


Protégé User Interface

Step 1 – Define the terms of the ontology and the relations between them

The concepts in the ONE Record ontology are digital twins of the physical entities in cargo industry. A digital twin is the "digital replica" of a physical entity and it can be applied to physical objects, processes, systems, etc. These are nouns (objects) or verbs (relationships) in sentences that describe the ONE Record domain.

In the following simple diagram, the objects are **Product**, **Item** and **Piece**, digital twins of the physical objects with the same name from freight ecosystem.



Fragment of ONE Record Data Model

Step 2 – Define the classes and the class hierarchy

The second phase of the ontology definition with Protégé is the creation of the classes. In our example, the classes would be **Item**, **Product** and **Piece**. As every Item is a Product, Item would be defined at a lower level than **Product**, meaning that **Item** is a subclass of **Product**.

Step 3 – Define the properties of classes

Properties define the internal structure of concepts, as classes alone cannot provide enough information about a domain.

In our example, we would create a **containedItem** property which would be of domain **Piece** and range **Item**, RDF translation of the fact that a **Piece** contains an **Item**.

All subclasses of a class inherit the properties of that class. For example, all the properties of the class **Product** will be inherited to **Item**, a subclass of **Product**.

Step 4 – Define further information about the properties

Properties can have different facets describing the value type, allowed values, the cardinality and other features of the values the properties can take.

The classes to which a property is attached or classes that the property describes are called the domain of the property. In our example, **containedItem** is of domain **Piece**.

Range define allowed classes for a property. In our example, **containedItem** can only be of range **Item**.

In our example, the property **containedItem** has also a cardinality of minimum 1, meaning that a **Piece** contains at least one **Item**.

Step 5 – Generate Turtle representation of the ontology

After all the classes and their properties are defined in the Protégé environment, a Turtle representation of the ontology can be exported.

Turtle is a syntax and file format (.ttl) for expressing RDF data model that is machine-readable. Software developers can use this file to implement applications consuming ONE Record compliant models.

Example of a simple generated Turtle representation for the description of **containedItem** property:

```
piece:containedItem rdf:type owl:ObjectProperty ;
                    rdfs:domain :Piece ;
                    rdfs:range :Item ;
                    rdfs:comment "Details of contained item(s) "@en ;
                    rdfs:label "piece:containedItem"@en .
```

Turtle syntax for **containedItem** property

The cardinality of **containedItem** is generated at the **Piece** level class description, as follows:

```
:Piece rdf:type owl:Class ;
       rdfs:subClassOf [ rdf:type owl:Restriction ;
                         owl:onProperty piece:containedItem ;
                         owl:minCardinality "1"^^xsd:nonNegativeInteger
                       ] ,
```

Turtle syntax for cardinality

Conclusion

In this document, we have described a short ontology-development methodology that is used by ONE Record data model and technical experts via Protégé tool. However, after following the suggestions, the most important things to remember are the following: there is no single correct ontology for a domain and there is no single correct way to create it.

More info at https://www.iata.org/one-record/.