Effective Ontology Development using the UML and Enterprise Architect

All material (c) Sparx Systems 2013
http://www.sparxsystems.com
Table of Contents

INTRODUCTION........................................................................................................................................3
WHY USE ONTOLOGIES?..................................................................................................................3
THE ONTOLOGY DEFINITION METAMODEL...........................................................................3
MDG TECHNOLOGY FOR ODM........................................................................................................5
ONTOSTLOGY EXAMPLES...................................................................................................................5
  W3C METADATA...............................................................................................................................6
  Classes.............................................................................................................................................7
  Properties.........................................................................................................................................8
  Semantics..........................................................................................................................................9
  Facts................................................................................................................................................11
IMPORT/EXPORT................................................................................................................................12
  Diagram Tip....................................................................................................................................12
  Diagram Filtering..........................................................................................................................13
CONCLUSION.....................................................................................................................................15
REFERENCES.......................................................................................................................................15
Introduction

When modeling across a diverse group of organizations that are exchanging information you need to have consistent definitions of terms and the relationships between terms. Using an Ontology you can define what you know about your domain and refer to these definitions within your information models.

Originally, the Web Ontology Language (OWL) and the Resource Definition Framework (RDF) were defined purely to provide a machine-to-machine interchange of metadata and semantics. Now, OWL and RDF have been combined to form the Ontology Definition Metamodel (ODM), that supports a more human readable, abstract modeling of these languages.

Enterprise Architect provides a wide range of modeling capabilities for interacting with metadata in a variety of ways, for a variety of purposes. This paper introduces the MDG Technology for ODM, providing a definition of ontologies and showing how they can be integrated with other Enterprise Architect features to address the key challenges of developing large-scale ontologies within the fully-integrated modeling environment.

Why use Ontologies?

The real value of ontologies lies in their ability to formally define standardized knowledge in an industry domain. Creating such ontologies, however, can quickly become a daunting task both logistically and technically. Four key challenges are:

• Stakeholder Enablement: The ability to engage subject matter experts, analysts, managers, practitioners and implementers to contribute value to ontologies
• Maintainability: The ability to update, compare and revise ontologies against predecessor baselines
• Reusability: The ability to define ontology templates for repeated use across development projects
• Traceability: The ability to ‘drill down’ or ‘drill up’ within ontology elements to visualize their interrelationships.

Addressing such challenges within the ontology development environment very quickly goes beyond the scope of traditional, XML-based ontology tools. It requires the investment of additional applications that operate outside the core development activity. This is why Enterprise Architect supports the integration of ontology development within the visual modeling platform using ODM, to simplify modeling on a more abstract level whilst maintaining the necessary details defined in the OWL and RDF specifications, and providing connectivity to other modeling languages used in system development.

The Ontology Definition Metamodel

Ontologies define a common vocabulary for practitioners who share information in a domain. They provide a standardized, machine- interpretable definition of basic concepts within the domain, and of relationships between them.
**Benefits**

Using formally defined ontologies, stakeholders can:

- Share a common understanding of the structure of information between stakeholders or software agents
- Re-use domain knowledge
- Explicitly codify domain assumptions
- Separate domain knowledge from operational knowledge
- Analyze domain knowledge effectively

The Ontology Definition Metamodel (ODM) was formalized as a standard modeling notation for ontology definition by the Object Management Group (OMG). This extension of the UML defines an important set of capabilities used in Model Driven Architecture for a formalized representation of business semantics, taxonomies and knowledge.

To better understand how and why this custom modeling capability is achieved by extending the UML, you could visualize a modeling language ‘stack’ as shown in Figure 1:

![Figure 1: The Modeling Languages Stack](image-url)

Each of the various domain-specific modeling languages extends the UML by defining additional metadata, semantics and notation to represent a domain-specific entity. Using these extensions to the UML, domain practitioners are more focused on capturing and visualizing their designs as an industry accepted model. Similarly, models that depict other design aspects of a system can be defined using other language
extensions, with interrelationships between models defining how one part of a design relates to another.

MDG Technology for ODM

In the MDG Technology for ODM, the Web Ontology Language (OWL) extends the Resource Definition Framework (RDF) to define model entities that represent OWL taxonomies and semantics, or RDF web resources. Those models can then be referenced by other Architecture models, such as the Services-Oriented Architecture defined using SoaML, to describe the underlying metadata used between services. Alternatively, they could be used to define semantics between different System Blocks using the Systems Modeling Language (SysML).

OWL and RDF extensions to the UML provide practitioners with the scope to not only define their Ontologies or Resource Definitions, but also to link their designs with others that describe different parts of a broader architecture or system.

The MDG Technology for ODM provides:

• The UML Profile for the Resource Description Framework (RDF), used to represent data, properties and formal semantics of information in the web
• The UML Profile for Web Ontology Language (OWL), used to represent terms in vocabularies and their relationships; OWL extends the RDF to define both the domain information and the relevant domain meaning
• Customized diagram types and toolboxes for convenient access to elements and relationships to model ontologies and resources effectively
• Model templates to quickly get started with ontology modeling
• Model scripts to import and export RDF and OWL files

Ontology Examples

The Ontology Definition Metamodel provides a way to express the meaning and understanding of your operating domain, by defining an ontology that contains:

• Classes
• Properties
• Semantics
• Facts

An Owl Ontology consists of an Owl package containing Owl Classes with specific properties and restrictions, along with facts related to these.

The main Owl package defines the Namespaces used in the ontology. See the example in Figure 2.
This namespace declaration for the Ontology package can consist of a set of namespaces that are defined in the Tagged Values window (Ctrl+Shift+6).

Figure 2: An Example Namespace
For details on each of these namespaces see WC3 namespaces.

**W3C Metadata**

ODM, being based on W3C OWL specifications, relies on metadata that is stored with the elements and relationships. These ODM properties are stored as Tagged Values and are accessible from the Tagged Values window (Ctrl+Shift+6).

---

**Classes**

Ontology is very useful in its ability to make inferences about an individual based on what group it belongs to - its Classification or 'Class'. To work with this you use a method of grouping objects based on common properties. Any Class can be a member collection under a higher Class – a parent Class.

Where a Class is a 'subclass', its properties can be inherited from its parent Class(es).
So, as with the UML, a Class defines a Classification, but there can also be **instances** of that Classification (an Object representing an artifact); for example 'Jane' might be an instance of the Classification 'Woman'.

With a child Class, any properties inherited from its parents can be viewed in the Tagged Values window.

**Properties**

With Properties, you can define characteristics of Classes. The Properties can be either:

- Datatype properties, or
- Object properties

Based on the RDF, the ODM uses element triples equivalent to entity relationships; rather than Entity-Attribute-Value it uses Subject-Predicate-Object. The Subject and Object are defined with **rdfsClass** and the Predicate is defined with an **rdfProperty**. For example, to represent the statement 'The Sky is Blue' the triple consists of a Subject *Sky*, a Predicate *Has Color* and an Object *Blue*, as shown in Figure 6.

![Properties defined in the Tagged Values window showing inheritance from parent elements](image-url)

**Figure 5**: Properties defined in the Tagged Values window showing inheritance from parent elements
Semantics

With the ODM, the OWL semantics are defined using relationships between Classes (elements). These relationships are accessible from the Diagram Toolbox, as shown in Figure 7.

As an example of using semantic relationships, consider the English statement: 'A Vegetarian pizza can consist of a union of a number of non meat-based pizza toppings'. The primary set of possible toppings are defined as subclasses of PizzaToppings, which includes vegetarian, meat and fish toppings, as shown in Figure 8.
In the OWL, the vegetarian toppings are stated as `owl:unionOf`. Figure 9 shows this in ODM form.

An example of using a different OWL semantic, `owl:complementOf`, is shown in Figure 10.
The `complementOf` relationship shows, in Figure 10, that meat or fish based toppings are not vegetarian toppings.

**Restrictions**

An OWL restriction is used to define a restricted set similar to an SQL `WHERE` condition. It defines a rule based on `onProperty`, and a Value Constraint such as `allValuesFrom` or `someValuesFrom`. This is typically used in conjunction with one of the other semantic relations, such as `subClassOf`.

For example, continuing the theme of pizza toppings, for the `Artichoke Topping` Class:

```
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasSpiciness"/>
    <owl:someValuesFrom rdf:resource="#Mild"/>
  </owl:Restriction>
</rdfs:subClassOf>
```

Figure 11 shows how this is represented in an ODM diagram.
This reads as: 'an artichoke topping is a vegetable topping with the `hasSpiciness` property mild'.

**Facts**

The **Fact model** is used to define specific facts about certain objects that are instances of Classes.

Figure 12 shows facts about *Paulene*, an instance of *Woman*, concerning her relationship to her parents. Specifically, it shows the fact that she has a male parent *Tim*. 
Import/Export

The MDG Technology for ODM supports the import and export of .owl and .rdf files. These facilities are accessible from the Enterprise Architect Main menu: Extensions | ODM.

Once imported to the model, these files can be interchanged with other UML models using the XMI Import/Export facilities, which include diagram specific details.

As .owl and .rdf files do not include graphical detail there is no initial diagram generated for the imported data.

Diagram Tip

Two useful features for setting up diagrams are:

- For a small ontology you can drag, from the Project Browser, the complete set of elements onto a diagram and format the diagram (main menu: Diagram | Layout Diagram)

- For a large ontology it is better to maintain a set of smaller, readable diagrams based around the key Classes; you can do this by dragging a specific element onto a new diagram and then using the Add Related Elements option to create the diagram around your key element

  - From the Project Browser, drag onto the diagram a Class that is likely to be a key object (like Publication, as opposed to Newspaper)
Series: Ontologies

Enterprise Architect
Visual Modeling Platform
http://www.sparxsystems.com.au

- Right-click on the Class in the diagram
- Select **Add Related Elements** from the context menu
- Set the **Levels** to import and tick the **Layout** option if preferred

![Diagram Filtering Example]

**Figure 13: Creating a Diagram using the Insert Related Elements feature**

**Diagram Filtering**

A simple OWL file can involve a complex set of relationships that you might find difficult to interpret. Using Diagram Filters you can simplify your diagrams, setting a specific connector type to be displayed. Figure 14 is an example of an unfiltered diagram with many **disjointWith** connectors.
Figure 15 shows a cleaner view of the same diagram, using a diagram filter set to only display the connectors of the `rdfsSubClassOf` connector type.

For more information on using this filter option, see the Diagram Filters Help topic.
Conclusion

The use of ontologies in model driven development provides a means of expressing a more formal definition of terms in the conceptual model. With each term having the ability to have specific relationships to other terms, ontologies provide a more concise and in-depth representation of the terminology that the model uses to reflect real world concepts.

Using Enterprise Architect's MDG Technology for ODM, you can now integrate ontologies into the visual modeling environment. This provides a standard, well-defined process for modeling the ontology, as well as allowing for interoperability with models based on other languages such as UML, SysML and UPDM.

References

For access to the ODM specification see:

- http://www.omg.org/spec/ODM/1.0/

For more information on the W3C OWL ontology see:

- http://www.w3.org/TR/owl-ref/

For a comprehensive list of ontologies accessible from the web see:

- http://owl.man.ac.uk/library/ontologies.txt

Sources for example Owl Ontologies used in this document:

- Pizza:
  http://www.co-ode.org/ontologies/pizza/2005/10/18/pizza.owl

- Mad Cows ontology
  http://www.cs.man.ac.uk/~horrocks/OWL/Ontologies/mad_cows.owl