DMTF - CIM to OWL: A Case Study in Ontology Conversion

Dennis Heimbigner

Computer Science Department
University of Colorado
Boulder, CO 80309-0430 USA
dennis.heimbigner@colorado.edu

Abstract

The process and problems associated with translating a specific software engineering ontology into a different ontology language are described. The software engineering ontology is the Distributed Management Task Force (DMTF) Common Information Model (CIM). The target ontology language is the W3C OWL DL language. The specific translations are described for various CIM constructs. Difficulties in translation are characterized.

1. Introduction

As part of a larger project, we had a need to integrate a number of software engineering models. Our approach was to choose a common modeling language into which all of the other models would be translated. Once translated, the various models could then be integrated.

Each of the existing models used a different modeling language, which meant that we either had to choose one of them or choose a language different from all of the others. After some examination, it became clear that each of the existing modeling languages was closely tied to the model's content, and thus not suitable for representing the other models. Further, each of the modeling languages appeared to be ad-hoc in one or more parts.

In light of this, we chose to use a completely separate modeling language, namely OWL [1] which is the latest ontology language from the World Wide Web Consortium (W3C). OWL had well-defined semantics, and was intended to represent the content of a wide variety of web-based data.

Our initial goal was to translate the Common Information Model (CIM) developed by the Distributed Management Task Force (DMTF) [2]. This model is designed to represent "…an implementation-neutral schema for describing overall management information in a network/enterprise environment".

To be precise, CIM is a model of the domain of managed software systems. Thus it has notions such as services, clients, and software components. This domain is represented using a more-or-less classical object-oriented programming model. For clarity, the modeling language will be referred to as CIM-O, and the domain representation (specific uses of CIM-O) will be referred to as CIM-D. The term CIM will be used for the combination. Our goal, then, was to re-represent the CIM domain (CIM-D) in OWL. The approach taken was to translate the CIM-O constructs used in the CIM-D representation to corresponding OWL constructs.

To give some additional background, the goal of the overall project is to address the problem of detecting previously unseen intrusions (e.g., viruses and worms) using *structural anomaly* detection. Instead of checking the behavior of software systems looking for anomalous behavior, our approach validates the structure of a running system against a comprehensive model describing that structure. Deviations are used to signal possible intrusions. This project requires a relatively complete model of the structure of a software system at run-time. This in turn requires the integration of a number of existing models, none of which alone is sufficient.

2. CIM-O Overview

The CIM-O language is a typical, although somewhat ad-hoc, object-oriented language. It has notions of class, attributes, and methods. The attributes represent named values associated with instances of the classes.

2.1. Classes

Classes in CIM-O use a single-inheritance model, so a basic class has the following skeleton

class <classname> : <superclass> {…}

where the superclass specification is optional. The body of the class is enclosed in curly brackets and contains a sequence of attributes and methods (with arguments).

2.2. Attributes

Attributes take one of the following general forms:

<simple type> name;

or

<class name> REF name;

The first case is used for a fixed set of literal types such as string, uint16 (e.g., 16 bit unsigned integer), Boolean, and so on. The second case is used when the attribute value is a pointer ("REF") to an instance of some class.

Any attribute may be designated as having a vector of values. Thus one might say

string commands [];

to indicate that a command attribute is a vector of strings.

Alternatively, one may indicate that an attribute has an initial/default value using this syntax.

uint16 priority := 0;

2.3. Methods

A method (procedure) has the following general form

<return type> name (arg, arg, …);

where each argument has the form of an attribute.

3. OWL Overview

We assume familiarity with OWL [1], or at least RDF [3]. OWL is basically much like the semantic models that were popular in the early 80's in the AI and Database communities. It has notions of class and property. In fact, it is essentially constructed using only binary relations consisting of a relation name, a domain set and a range set (a triple in RDF terminology). Thus, subclass inheritance is a relation between the parent and subclass.

Note that OWL comes in three "flavors": lite, DL, and full. The term DL stands for "description logics" because this level of OWL is equivalent to description logics in expressive power. We adhere to OWL DL except where noted.

4. Translation Phase 1

The first translation phase addressed the most significant problems: how to translate the specific classes and simple attributes used in CIM-D.

4.1. Class Conversion

Converting a specific class skeleton in CIM-D to OWL is relatively straightforward. The following class

class Service : System {…}

would translate to the following.

```
<OWL:Class rdf:ID="Service">
  <rdfs:subClassOf rdf:resource="#System"/>
</OWL:Class>
```

OWL supports a variety of class formation mechanisms, including multiple inheritance, union, and intersection. But these are unneeded for translating CIM-D classes, although they are needed for translating attributes.

4.2. Attribute Conversion

At first glance, it might appear that CIM-O attributes can naturally be converted to OWL properties. Unfortunately, a number of conversion problems surfaced.

The first problem, name scoping, is one that occurs repeatedly in any attempt to map CIM-O to OWL. In CIM-O the domain of an attribute is implicitly scoped to the class in which the attribute is defined. One would normally map an O-O attribute to an OWL property with the class being the domain and the value type of the attribute being mapped to the range of the OWL property. We would expect to map the specific CIM-D attribute

string Name;

to the following OWL format.

```
<OWL:DataTypeProperty rdf:ID="Name">
  <rdfs:domain rdf:resource="#Service"/>
  <rdfs:range rdf:resource="xsd:string"/>
</OWL:DataTypeProperty>
```

This is technically correct, but incomplete, because in OWL, normal properties have global scope, and hence are independent of classes. If, as is common, the same CIM-D property occurred in another class, then this would conflict with the existing definition of the Name property because we would be trying to define the property with two different domains.

A similar problem exists for the range specification. The default is that the range of a property is global to all uses of the property. Thus we cannot have an alternative definition of Name whose range is integer instead of string.

There are basically two ways out of these problems.

1. We can rename the property to include the domain class and thus make each property unique. Thus, the above example would be converted from Name to something like Service_Name. This also implicitly handles the range problem because each unique property can have whatever range it requires.
2. We can utilize the OWL *allValuesFrom* restriction on the property to essentially indicate that when the domain comes from a specific class, the range is the class specified by the *allValuesFrom* restriction.

We chose the second solution, but recognized that it complicated our translation process. The first solution was rejected because it would be difficult to decide the name of the property to use when constructing instances, and it apparently would complicate the handling of inheritance. In choosing solution 2, we hoped it would not be needed very often. In many cases, attributes were in fact unique to a specific class; hence, they could be defined using the simple approach.

In some cases, all definitions of attributes with the same name in fact had the same range. This is true of the Name attribute; its range is always a string. This would result in the very general definition such as the following.

```
<OWL:DataTypeProperty rdf:ID="Name">
  <rdfs:domain rdf:resource="OWL:Thing"/>
  <rdfs:range rdf:resource="xsd:string"/>
</OWL:DataTypeProperty>
```

This case can be extended because even if not all ranges are the same, most of them may be the same, in which case the more complex use of a restriction need only be used for those exceptional situations. If it turned out that Name had an integer range (sint32) for the class

Device, then we would add a specific restriction for that class.

4.3. Attribute Objectification

CIM-D contains one specific class whose purpose is to represent directed dependency relationships between two arbitrary objects. This class, named "CIM_Dependency" has two attributes: *Antecedent* and *Dependent*. The direction is from *Antecedent* to *Dependent*. Subclasses of this class are frequently created in CIM-D to represent specific dependency relationships. The following example defines a relationship between a service object (CIM_Service) and a managed element, which is any generic software object.

```
class CIM_ProvidesServiceToElement
      : CIM_Dependency {
  CIM_Service REF Antecedent;
  CIM_ManagedElement REF Dependent;
};
```

Such classes are often referred to as *objectified* attributes. Clearly any such relationship could be represented by an attribute attached to the antecedent class, but instead, it has been converted to a separate class of objects representing the relationship. Objectification is typically used when either the relationship does not naturally associate with the antecedent (or the dephendent, for that matter), or when the objectified relationship is actually an n-ary relationship for n > 2. CIM-D uses objectification for both reasons. Many of the CIM_Dependency subclasses are independent of the antecedent, and as well, some of the subclasses have additional attributes that convert them to ternary relationships.

We have followed the CIM-D model in this approach and have kept objectified relationships in our translation. Of course, for n-ary relationships, we have no choice since OWL does not support n-ary properties.

5. Secondary Translation Issues

The first phase of translation indicated that the process was generally feasible. The second phase involved examination of various secondary features of CIM-O used in CIM-D to see how they might be translated to OWL.

5.1. Vectors

CIM-O class attributes can specify that their range is actually a vector of values (Section 2.2). Mapping vectors into OWL is one of those topics for which no discussion could be found in the OWL documentation. Frankly we have no idea what the "proper" mapping should be, so we have developed our own representation using the rdf:List class which is defined as OWL's sole container type; this is at the technical cost giving up the random access behavior of the vector. OWL uses only lists because they contain a fixed number of elements: the *first* and *rest* properties as of the rdf:List class. This two-element definition apparently simplifies reasoning about lists.

Our solution defines for each class C, another class, vectorC, representing the vectors whose elements are of type C. The vectorC object is defined as follows.

```
<OWL:Class rdf:ID="vectorC">
  <rdfs:subClassOf rdf:resource="rdf:List"/>
</OWL:Class>
```

This definition is unsatisfying because it places no restrictions on the types of elements in the list. In an attempt to remedy this, we make use of the *allValuesFrom* restriction to force the values to be of the correct type and to force the list nodes to be of type vectorC.

A possible solution recently suggested was to model vectors as multi-valued properties and use the MaxCardinality restriction to handle non-vector attributes. This would, however, appear to give up the ordering property of a vector, which the list approach keeps.

5.2. Default Values

Another feature of CIM-O is the assignment of default values for attributes. Surprisingly, modeling default values turns out to be very difficult in OWL. The subject has been extensively discussed in the W3C. From that discussion we concluded that (1) defaulting cannot be included in the standard OWL DL model, and (2) there was no agreement about defaulting and users were encouraged to experiment. In effect, we were on our own. This was irritating, and almost made us abandon our choice of OWL as our common modeling language.

The solution we are currently pursuing is to treat defaulting as a form of inference. This idea came from experiments with the Jena ontology database system [5]. Figure 1 shows one view of the architecture of Jena. The idea is that one has a base graph representing some model — our converted CIM-D model, for example. This is accessed using the standard Jena graph API. Additional components can be layered over that graph with the proviso that each layer exports that same standard graph API. The figure shows an important case of layering where the layered component is a *reasoner* that supports inferences over the underlying graph and makes those inferred edges and nodes visible through its interface. In effect, the reasoner extends the base graph with any nodes and edges that can be inferred from it. This layering can be continued by adding additional reasoners.

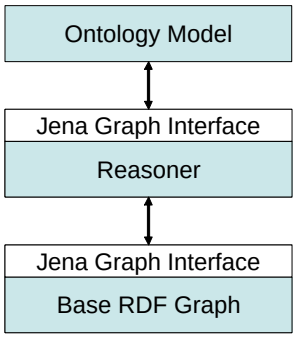

Figure 1. Jena Graph Layering.

An example is the transitive reasoner. OWL supports transitivity by explicitly attaching a "TransitiveProperty"

annotation to properties. The transitive reasoner adds virtual edges to the base graph based on the existence of transitive markers in that base graph.

Our approach to default values is based on this idea. The key is to treat default values as an inference problem. In particular, suppose there is a query for all triples of the form (A,priority,_). If a search of the base graph returns the empty set, then the default reasoner applies the relevant defaulting rules and returns, for example, the triple (A,priority,0). This assumes that the base graph has been annotated with defaulting markers.

5.3. Methods

Another problem in translating CIM-D to OWL involves methods, which are the signatures for executable procedures associated with a class. An ontology language like OWL has no built-in concept of procedure, so we were again forced to improvise.

There are a number of approaches for modeling methods, but many of them fail in the face of overloaded names and signatures. Our approach requires several ontological elements. There is no room for details here, but, briefly, we define a class for the method, and that class is used as the value for a property of the class containing the method. The arguments of the method become attributes of the method class.

The remaining issue is method inheritance. This is technically an execution time issue, which we consider to be out of scope for this current translation effort.

6. Related Work

The RDF primer [3] makes reference to a project to convert the DMTF CIM model to RDF, however, no reference is given, and a search of the web found no references to this project as of the time of writing of this paper. When it becomes available, a comparison with the work described here will undoubtedly be instructive.

Others have recognized the potential for using an ontology language to represent software engineering information [4]. However, these efforts appear to be in the early stages of development, and the DMTF CIM model does not appear to be one of their targets.

7. Discussion and Conclusions

Translating the DMTF CIM(-D) model into OWL turned out to be more difficult than originally anticipated. In retrospect, using OWL as the common model for our project may not have been a good choice.

OWL's limited notion of scope was a significant problem. The use of a flat, partitioned namespace makes it hard to easily represent a system like CIM where nested naming and implicit reference are inherent in its semantics. Scoping in CIM has associated semantics (basically inheritance), and that is not easily represented in OWL. The only solution may be to flatten the CIM namespace and then find an approach for handling the implicit semantics.

The lack of sophisticated containers in OWL is another major issue. In mapping CIM vectors to lists, for example, the random access nature of vectors is lost. For our purposes, this is not essential, but it is irritating and it may be important in other contexts.

Defaulting is a major problem but we are satisfied with our solution.

In summary, our experience in attempting a natural translation from CIM to OWL was disappointing. We recognize that OWL may not have been designed for this purpose, but that limits its utility outside of the semantic-web context.

Acknowledgements. This material is based in part upon work sponsored by the DARPA and AFRL Rome Labs, under Contract Number F30602-00-2-0608. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

References

[1] S. Bechjofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Pagel-Schneider, L.A. Stein. "OWL Web Ontology Language Reference W3C Recommendation 10", W3C Working Group , February 2004. http://www.w3.org/TR/2004/REC-OWL-ref-20040210/.

[2] Distributed Management Task Force, Inc. ,"Common Information Model (CIM) Specification Version 2.2", January 14, 1999. http://www.dmtf.org/standards/documents/DMI/DSP0005.pdf

[3] F. Manola, and E. Miller, "RDF Primer", W3C Working Group, Sept. 5, 2003. http://www.w3.org/TR/2003/WD-rdf-primer-20030905/.

[4] J. E. López de Vergara, V. A. Villagrá, J. I. Asensio, J. Berrocal, "Ontologies: Giving Semantics to Network Management Models," IEEE Network, special issue on Network Management 17(3) (May/June) 2003.

[5] B. McBride, "Jena: Implementing the RDF Model and Syntax Specification", Proc. of the 2nd Int'l Workshop on the Semantic Web, Hongkong, China, May 1, 2001.