A JC3IEDM OWL-DL Ontology

Steven Wartik
Institute for Defense Analyses
Alexandria, VA
swartik@ida.org

One of the major unanswered questions about the semantic web is the degree to which real-world business rules can be expressed in OWL-DL. Ontologies that only contain class hierarchies are useful searching aides, but are unlikely to promote automated reasoning and classification. Only ontologies augmented with more complex concepts can effectively identify inconsistencies and classify individuals. However, OWL-DL is by design restrictive in what concepts it can represent. It will only prove truly useful if its restrictions mainly apply to uncommon situations.

In this paper we discuss using OWL to express business rules from the domain of Consultation, Command and Control (C3). We have created an OWL-DL representation of the Joint C3 Information Exchange Data Model (JC3IEDM). The JC3IEDM is a NATO standard agreement (STANAG 5525) for information exchange among NATO members in C3. The JC3IEDM has been under development for 11 years, and its antecedents date back to the 1980s. It contains over 4,500 business rules that govern valid relationships among its data elements. It can fairly be said to represent a nontrivial domain-specific body of knowledge derived from experience and real-world concerns.

Historically, the JC3IEDM has been used to achieve common understanding and a shared operational picture through information exchange via database messaging and replication, but in the past few years its governing body has considered how it can be used in a net-centric environment. It is our belief that the JC3IEDM is well suited to use in net-centric operations. It comprises many commonly used concepts (Figure 1) and draws on authoritative sources to define them. It can provide a rich foundational ontology for elements of C3 (and C2).

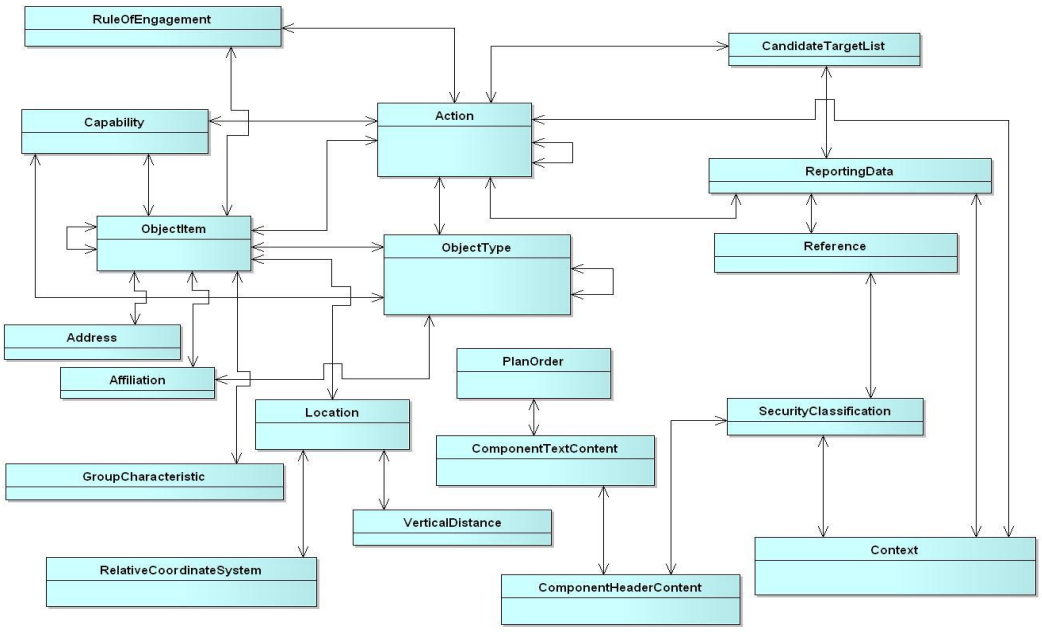

Figure 1. Major JC3IEDM Concepts and Associations

Proceedings of OWL: Experiences and Directions 2009 (OWLED 2009),
Rinke Hoekstra and Peter F. Patel-Schneider, editors. http://www.webont.org/owled/2009

The basic transformation of the elements shown in Figure 1 is straightforward; it is described briefly below. More interesting is the transformation of JC3IEDM's business rules, discussion of which forms the bulk of this paper.

The JC3IEDM is expressed as a UML[1] class model.[2] It contains about 300 classes. Most of these classes have attributes. The attributes' types include numbers and strings. About half are enumerated, which is significant because the majority of business rules involve enumerated attributes.

Each JC3IEDM class maps 1:1 to an OWL class. JC3IEDM has a class hierarchy, which the OWL mapping preserves. JC3IEDM classes have textual descriptions; these are mapped to RDFS comments. Attributes map to a property, an object type if enumerated and a data type if not. The values of the enumerated type are mapped to an OWL enumerated class.

The JC3IEDM uses both one-to-many and many-to-many associations. Both map to OWL object properties, the domain and range of which derive from the classes at the association ends. If both ends of the association are navigable, the OWL mapping specifies inverses.

In the JC3IEDM, sibling classes are disjoint. The subclasses of ObjectItem include Person, Organisation, and Feature; an instance of Person cannot be a Feature or an Organisation. The ontology mapping proscribes instances from being a member of more than one JC3IEDM class. In many (but not all) cases a parent is completely defined by its subclasses, a fact captured in the ontology; thus $\mathsf{PlanOrder} \equiv \mathsf{Plan} \cup \mathsf{Order}$.

All JC3IEDM attributes are functional, a fact we carry over into OWL. Some JC3IEDM attributes are required. For example, an ObjectItem must have a name. For such attributes, we add an exactly cardinality restriction of 1 to the ontology.

Non-enumerated JC3IEDM attributes usually have restrictions beyond those expressed by XSD domains. For example, a string has a maximum length, and an angle must be between 0 and 359.9999, inclusive. The mapping translates these restrictions to SWRL. Class ActionLocation has an attribute named bearingAngle, so the ontology includes the rule:

$$\mathsf{ActionLocation}(?l) \,\&\, \mathsf{ActionLocation\_bearingAngle}(?l, ?a) \rightarrow \\ \mathsf{swrlb:greaterThanOrEqual}(?l, 0.0) \,\&\, \mathsf{swrlb:lessThanOrEqual}(?l, 359.9999)$$

(Our ontology is expressed in OWL 1.0. OWL 2 can express this constraint directly, and we expect to switch when the OWL 2 recommendation is accepted and widely implemented.)

The material presented so far shows a simple, more or less one-to-one mapping of JC3IEDM and OWL concepts. The JC3IEDM's business rules are more complex. The rules are written in the Object Constraint Language (OCL). OCL is a simple UML-based expression language. Given a UML "context" (a class, attribute, etc.), one writes an expression using elements accessible from that context. OCL is especially useful for writing invariants (conditions that must be satisfied), which exactly captures the purpose of a JC3IEDM business rule. For example, class AircraftType contains the following invariants:

---

1 See http://omg.org/ for descriptions of UML concepts and technologies.

2 Available at http://mda.cloudexp.com/downloads.html.

```
context AircraftType
inv: categoryCode = LighterThanAir implies
    Set { Balloon, Dirigible, NotOtherwiseSpecified }->includes(airframeDesignCode)
    or airframeDesignCode.oclIsUndefined()
inv: categoryCode = SpaceVehical implies
    Set { Satellite, NotOtherwiseSpecified } ->includes(airframeDesignCode)
inv: Set { NotKnown, NotOtherwiseSpecified }->includes(categoryCode) implies
    airframeDesignCode.oclIsUndefined()
```

which specify the values attribute airframeDesignCode may assume based on the value of attribute categoryCode.

Each invariant, being an implication $A \Rightarrow B$, can be expressed in the form $\neg A$ or B. The sequence of invariants can be expressed in set notation:

$$\begin{aligned} \text{AircraftType} \subseteq \{ & (\text{not } (\text{categoryCode} = \text{LighterThanAir}) \\ & \quad \text{or airframeDesignCode} \in \{ \text{Balloon, Dirigible, NotOtherwiseSpecified} \} \\ & \quad \text{or } |\text{airframeDesignCode}| = 0) \\ \cup \ & (\text{not } (\text{categoryCode} = \text{SpaceVehicle}) \\ & \quad \text{or airframeDesignCode} \in \{ \text{Satellite, NotOtherwiseSpecified} \}) \\ \cup \ & (\text{not } (\text{categoryCode} \in \{ \text{NotOtherwiseSpecified, NotKnown} \} \\ & \quad \text{or } |\text{airframeDesignCode}| = 0) \end{aligned}$$

This set, in turn, can be stated in OWL as a subclass restriction on AircraftType.

The approach of converting implications to set expressions captures almost ¾ of the JC3IEDM's business rules. There are of course some far more complex rules. The above invariants all dealt with attributes in a single class. Many JC3IEDM business rules deal with multiple classes. For instance:

- An ObjectItem has an associated ObjectType; the ObjectType's subclass is constrained by the kind of ObjectItem. For instance, an instance of Person can only be associated with an instance of PersonType.
- An ObjectItem may have associated instances of Capability. The allowed capabilities depend on the ObjectItem's kind.
- ObjectItem and Action may have associated instances of Location, and in both cases the kind of Location depends on the nature of the ObjectItem or Action.

In the sense that an attribute is really a kind of association (just as both data types and object types are properties), translating these kind of invariants is the same as translating those whose attributes are within a single class. However, the resulting set expressions are more complicated, involving levels of nesting. This nesting translates to nested property restrictions. Consider the invariant:

```
context ActionTask
inv: activityCode = MCM37 implies
    is_geometrically_defined_through->forAll( oclIsKindOf(Point) )
```

This states that any ActionTask whose activityCode attribute has the value MCM37 can only be associated with locations that are instances of class Point. We can translate this to OWL by noting that the forAll construct is equivalent to a restriction on the is_geometrically_defined_through property in which all values are restricted to be from class Point. We can extend this pattern further, using nested all-values-from and some-values-

from restrictions, and cover every JC3IEDM business rule that involves enumerated attributes and subclassing restrictions.

Some JC3IEDM business rules cannot be expressed in OWL. They fall into three categories:

1. Any collection that's not a set. For instance, JC3IEDM defines a line as a set of integer-ordered points; the integers must begin with 1 and increase by 1.
2. A value comparison that's not to a pre-specified individual. In JC3IEDM all points of an ellipse must be in a plane; the points of this plane are dynamically determined.
3. A rule involving complex computations. OWL 1.0 has no computational ability. SWRL, used for some rules (see above), is limited.

We have written a Java application that fully automates the transformation from UML to OWL. Using this application has the obvious advantage of repeatability and, from a user's perspective, simplicity. The most recent translation (the JC3IEDM is an evolving model) has 791 OWL classes, 1,348 properties (969 object, 379 datatype), and about 150,000 RDF triples.

Automated transformation is not entirely advantageous. The OWL ontology is split into four parts: basic elements (classes and properties), enumerated domains, business rules that can be expressed in OWL-DL, and business rules that can't be expressed in OWL-DL but can be expressed in SWRL. Even with this division, some of the ontologies are so big they confound many of the reasoners we've tried. We originally declared the individuals distinct in the enumerations and promptly exceeded the memory limitations of every 32-bit reasoner we tried (not to mention several OWL editors). We are studying how to split the elements more intelligently. Intelligent splitting may require human intelligence and not just heuristics, an area we view as important in future research.