On the Automatic Generation of an OWL Ontology based on the Joint C3 Information Exchange Data Model

Christopher J. Matheus[1] and Brian Ulicny[1]

[1] Versatile Information Systems, Inc.,
Framingham, Massachusetts, U.S.A
{cmatheus, bulicny}@vistology.com

Abstract. The JC3IEDM is a data model intended for the exchange of command, control and communication information. It is available as an ERwin data model for which there is an XML-based description of all entities, attributes, relations and codes, making it ripe for translation by XSLT. This paper describes the development of a set of transformation scripts that convert the JC3IEDM data model into an OWL ontology; it explains the major challenges encountered and discusses a number of issues concerning the practical use of the resulting ontology. While the primary purpose of this work is to provide the basis for a semantically rich ontology for use in representing and reasoning about command, control and communication operations, it is also intended to serve as an example of a general approach for translating ERwin data models into OWL ontologies.

Keywords: ontology generation, JC3IEDM, ERWin model, automated translation, XML, XSLT

1 Introduction

The Multilateral Interoperability Programme [1] (MIP) is a long-standing, NATO-supported program intended to foster international interoperability of Command and Control Information Systems (C2IS) through the development of standard data models and data exchange mechanisms. Significant joint coalition effort has gone into the development of the MIP data model which was first released in the mid to late 1990's as various version of the Generic Hub (GH) Data Model; in subsequent years it became known as the Land Command and Control Information Exchange Data Model (LC2IEDM), followed by the Command and Control Information Exchange Data Model (C2IEDM) and now it exists as the Joint Command, Control and Communication Information Exchange Data Model (**JC3IEDM**) [2]. The data model captures information about objects and their properties, situations made up of facts about objects and activities involving collections of objects. While JC3IEDM is intended foremost for the exchange of command, control and communication information between information systems, it is gaining increased considered as the basis for the general data models that underlie C3 information systems. A primary reason for this trend is the desire to leverage the great wealth of experience and knowledge that has gone into the development of JC3IEDM 3.0.

Our particular interest in JC3IEDM is its use as the basis for several ontologically-based reasoning applications that assist in establishing situation awareness [3],[4]. The first step in making this happen is the conversion of JC3IEDM into the Web Ontology Language OWL [5]. We demonstrated the feasibility of this task in an earlier effort that set out to capture a subset of the (then) C2IEDM sufficient for representing OTH-T GOLD Track data [6]. In that project we manually translated portions of the data model into OWL, which was reasonable given the scope of that problem. We recently became interested with the much more challenging task of developing a complete translation of JC3IEDM into OWL. Given there are 289 entities, 396 relationships between entities, 1729 entity attributes and nearly 7000 value codes, plus the fact that the MIP data model is updated on a regular (one might say "aggressive") basis, this task was clearly in need of automation. This paper describes the methods used to translate JC3IEDM 3.0 into OWL DL using a series of XSLT scripts. We believe this work will be of interest to others for the following reasons:

1. We are making the translated JC3IEDM OWL ontology freely available for others to use; this paper servers as both an announcement of its availability and as an explanation of why we chose to translate the various aspects of the data model as we did

2. The translation is performed using the XML document that specifies the JC3IEDM ERwin data model definition; due to the use of this XSD-defined document, other ERwin based data models can be translated into OWL using a similar strategy (and in many cases, the code) that is described here

3. There are a number of interesting questions that arose during the process of developing the translation, some of which are worth further discussion and contemplation by a larger community

This paper begins with an introduction to the JC3IEDM data model and the ERwin XML definition document. Explanations are then given for each of the translation scripts used to transform Entities, Attributes, Relationships and Codes. We conclude with a discussion of some open issues and questions.

2 The JC3IEDM

JC3IEDM is a relational data model that can be viewed from one of three perspectives: conceptual, logical and physical. The conceptual model is an abstract view of the important high-level data elements (e.g., ACTIONS, PERSONNEL, FACILITIES, etc.) and is useful for understanding the scope and general content of the data model. The logical model adds all of the more specific details needed to understand the logical connections between the elements while striving to be a model accessible to processing by humans. The physical model is concerned with the information necessary to implement the data model as a database schema; it extends the logical model with information about keys and redundant data used for efficiency purposes and is implemented using ERwin™ Version 3.5.2 software from Computer Associates International, Inc. The model of most interest to us is the logical model since we are interested in capturing the logical semantics of the model with no concern for its realization as an instance of a database schema. We use the conceptual model in this section to provide a brief overview of its contents.

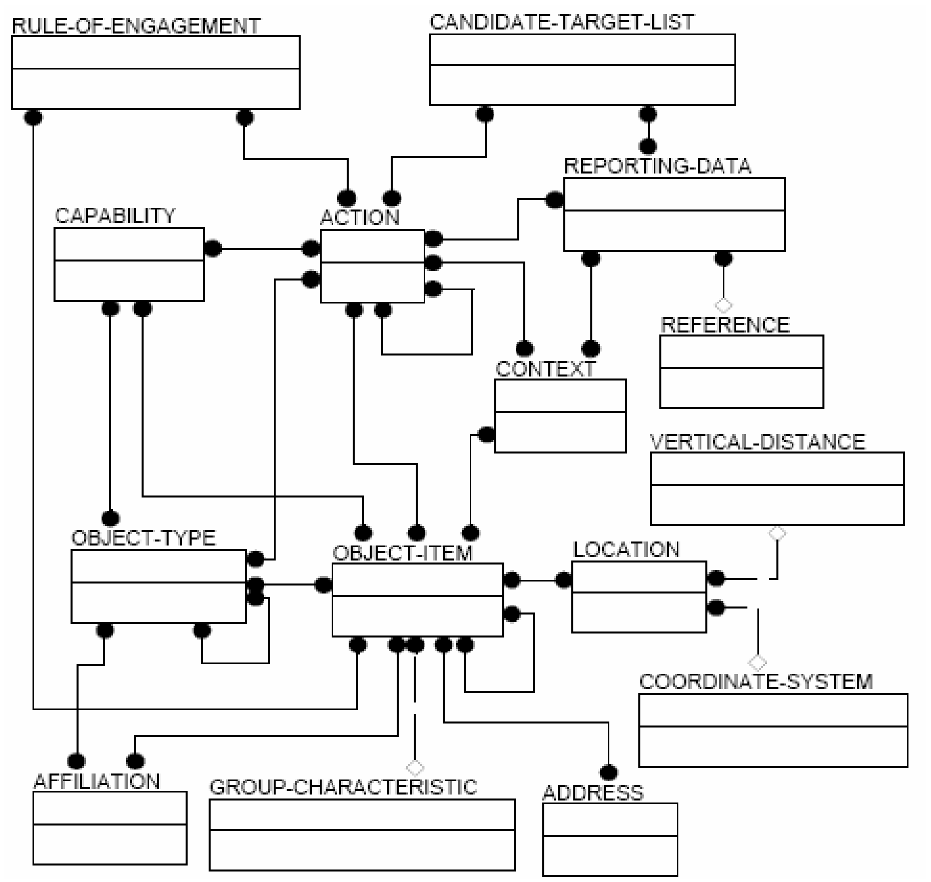

Fig 1. JC3IEDM Independent Entities shown in IDEF1X notation [7] as they appear in the Conceptual Model. (Copied from [2])

Fig 1 shows all of independent entities found at the highest level of JC3IEDM along with the conceptual relationships between them. These relationships represent conceptual aggregates of finer relationships and additional entities found in the logical model. Some things to note here include:

1) Most of these Entities are sub-classed in the logical model and in some cases the hierarchy of classes can be relatively deep (i.e., greater than 5).

2) There are two high-level object classes, OBJECT-TYPE and OBJECT-ITEM. OBJECT-TYPE is used for more static information associated with an entire class of objects (e.g., the track width of an Abrams Tank, its maximum speed, etc.) whereas OBJECT-ITEM is used to capture information specific to individuals (e.g., the speed of a tank, the fact it has 5 gallons of gas, etc.).

3) The OBJECT-TYPE and OBJECT-ITEM entities have parallel class/subclass hierarchies as shown (to a depth of one) in Fig 2. The hierarchies do not fully mirror each other, particularly deeper within the structures, but they are closely related.

4) REPORTING-DATA represents pedigree information that is used extensively to identify when, from whom and how reliable/credible a specific piece of information is.

For more detailed information about the JC3IEDM conceptual and logical models the reader is referred to the following online MIP documents: JC3IEDM Overview [8], JC3IEDM Main [2], JC3IEDM Logical Model Diagram [9].

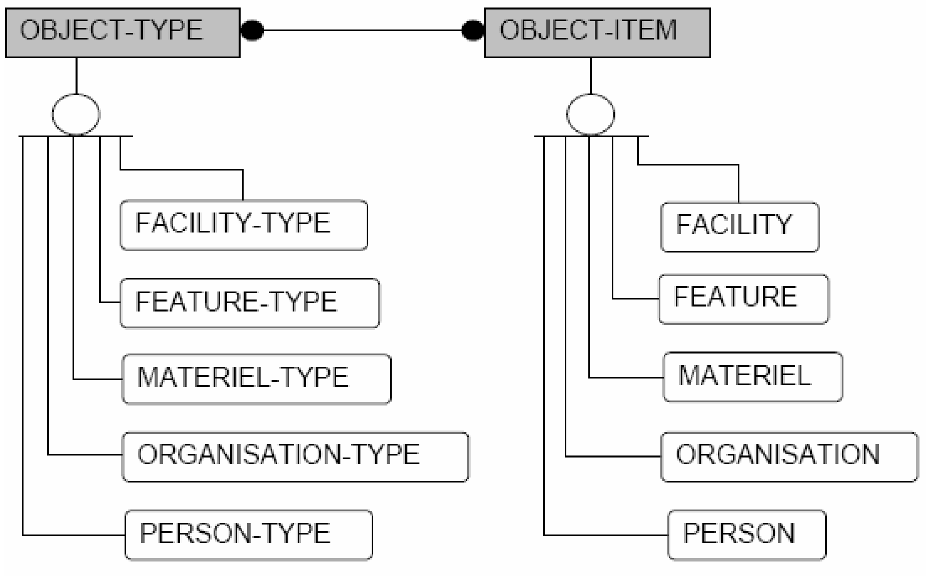

Fig 2. Conceptual relationship between OBJECT-TYPE and OBJECT-ITEM Entities along with a depiction of the first level of subclasses for each suggestive of the parallel hierarchies used for OBJECT representation in JC3IEDM. (Copied from [2])

3 ERwin XML Definition

The JC3IEDM 3.0 release comes complete with thorough documentation, a Microsoft Access database and an XML distribution package that includes code and support files for generating (in terms of XML Schemas) both an object-oriented mapping of the data model to XML and a relational mapping of the data model to XML. These XSD are generated from an ERwin XML definition document that contains the entire model for JC3IEDM (i.e., both logical and physical views). It is this XML definition document that we used as the basis for our translation effort.

For our purposes we are only interested in the logical aspects of the ERwin model. The following abstract code based on the ERwin XML definition document shows the structure of the relevant fragments of the document that our translation scripts focus on.

```
<ERwin4>
  <Entity_Groups>
    <Entity>
      <EntityProps/>
      <Attribute_Groups>
       <Attribute/>
           …
     </Attribute_Groups>
   </Entity>
   <Domain_Groups>
     <Domain/>
        …
   <Domain_Groups>
   <Relationship_Groups>
     <Relationship/>
           …
   </Relationship_Groups>
   <Validation_Rule_Groups>
     <Validation_Rule/>
            …
   </Validation_Rule_Groups>
</ERwin4>
```

All of the elements of interest are located within five named element groups: Entity_Groups, Attribute_Groups, Relationship_Groups, Domain_Groups and Validation_Rule_Groups. The elements in the Entity_Group include specific Entities that will become owl:Classes along with their corresponding Attribute_Groups which in turn contain the specific Attributes for the corresponding Entities; each of these Attributes will become either an owl:ObjectProperty or an owl:DatatypeProperty. The Relationship_Groups contain the Relationships that can occur between Entities, each of which will be turned into owl:ObjectProperties. The Domain_Groups in conjunction with the Validation_Rules_Groups contain the values that are permitted for the domains and ranges of the Attributes; some of these correspond to specific Codes that imply specific meaning and will be captured within enumeration classes within OWL [10].

4 Entity to Class Translation

The script to translate entities into classes separately processes each Entity element in the ERwin definition document. It uses the Name attribute of the Entity element for the rdf:ID of the owl:Class and selects the EntityProps/Name element textNode to use as the rdfs:label of the class[1]. The EntityProps/Definition element's contents is used as the rdfs:comment for the class as it provides a English text description of what the class represents. All Entities are defined as owl:Classes using the following minimal format (which borrows from XSLT's convention of using {} to indicate references to XPATH addresses within the current element):

```
<owl:Class rdf:ID=”{@Name}”>
  <rdfs:label>{EntityProps/Name}</rdfs:label>
  <rdfs:comment>{EntityProps/Definition}</rdfs:comment>
</owl:Class>
```

For example, the following code is generated when the @Name=”ACTION”, EntityProps/Name=”ACTION” and EntityProps/Definition is equal to the text shown in the rdfs:comments element:

```
<owl:Class rdf:ID="ACTION">
 <rdfs:label rdf:datatype="&rdf;Literal">ACTION</rdfs:label>
 <rdfs:comment>An activity, or the occurrence of
    an activity, that may utilise resources and may
    be focused against an objective.
 </rdfs:comment>
</owl:Class>
```

Many of the Entities exist as sub-classes of at most one other entity (i.e., the data model is represented as a collection of trees). An Entity is the subclass of a parent Entity if it exists as a Valid_Value for the parent Entity's category-code Attribute. Since each Entity has at most one parent, it is possible to search to see if there is a Validation_Rule that has the child Entity within its list of Valid_Values and then look up the Entity that has an Attribute that has a Parent_Domain that uses that Validation_Rule. In shorthand XSLT code, this reads as follow:

```
ruleID="//Validation_Rule[Valid_Value_Groups/Valid_Value/Valid_ValueProp
s/Display=$entityName]/@id"

domainID="//Domain[DomainProps/Validation_Rule_Ref=$ruleID]/@id”

parentEntityName="//Entity[Attribute_Groups/Attribute/AttributeProps/Par
ent_Domain=$domainID]/@Name"
```

An Entity for which the parentEntityName is non-null is designated to be a owl:subClassOf the parent Entity.

[1] The Name attribute and the EntityProps/Name element are always equal in the current JC3IEDM 3.0 definition but there is no explicit reason for this to remain the case in future releases.

Unfortunately, all classes of objects are not defined as entities – only those that have additional Attributes or Relations appear as Entities and all others are represented by values of category-codes. A category-code value is a string the uniquely identifies a sub-class of an Entity – that is to say, the string is unique among the subclass names for a particular Entity, but it may be used as a subclass name for more than one Entity. Entities that have subclasses defined in this way will have an Attribute whose name is the entity's name in lowercase with the string "category-code" appended to it, e.g. "object-type-category-code" for the entity OBJECT-TYPE. Furthermore, there will be a Domain for this Attribute with the same name. To define all of the subclasses defined by this Domain it is necessary to iterate over all of the Domain's Valid_Values as defined by its Validation_Rule, and for each value that is not the name of a defined Entity (category-code values include all subclasses, both those that are actual Entities and those that are not) a new owl:Class is created exactly as described above for Entities but in this case the parent class is already known and so the new class is always defined to be a owl:subClassOf its parent.

There is one slight complication to the construction of these category-code classes. As indicated above, it is possible for a category-code string to be used as the name for a subclass in more then one category-code Attribute. This means that it is not possible to use the category-code value alone as the class' rdf:ID, owing to the requirement that rdf:ID be unique within a single ontology. A test is therefore performed to see if there is an earlier use of the category-code string by another Entity and if so, the name of the parent class is prepended to the string to construct the class's rdf:ID. We could have simplified the code by always prepending the parent class name but in striving to make the rdf:IDs as human-friendly as possible we felt it was better to leave the parent class names off whenever possible.

5 Attribute to Property Translation

Every Attribute element in the JC3IEDM definition is processed and turned into either an owl:DatatypeProperty or an owl:ObjectProperty provided its @Name attribute does not contain any of the following strings: `-category-code`, `-id`, `-index`, `ent_cat_code`, `-update_seqnr` and `-uodate_seqnr`. If the @Name attribute contains "-category-code" then it will be handled by the script that translates Entities (as described in the preceding section) and its values will be turned into owl:Classes. The -id Attributes used in the physical model are not necessary in the OWL ontology as they are taken care of by each instance's unique rdf:ID attribute. The -indexes and ent_cat_code Attributes are part of the physical model and are thus excluded. The update_seqnr Attributes are used in the JC3IEDM model for "replication management" and specify the relative seniority of a data element; since this information is not relevant to the representation of the data in OWL these Attributes are ignored. The appearance of an Attribute containing the string "uodate_sequr" is assumed to be an error in the definition file (it appears that the "o" should have been a "p").

Attributes can represent either owl:ObjectProperties or owl:DatatypeProperties. Attributes that range over a category-code are identified as owl:ObjectProperties because all category code's are translated into owl:Classes. The range for these properties are defined by the Attribute's corresponding Parent_Domain; the name of the owl:Class for the range is obtained by removing hyphens from the Domain name and capitalizing the first letter of each hyphenated substring. For example, the Domain name angle-precision-code becomes the class name AnglePrecisionCode.

For Attributes corresponding to owl:DatatypeProperties it is necessary to do some further processing to determine which XSD datatype should be used for the rdfs:range. Datatype information is either encoded at the level of the AttributeProps or at the level of the the Domain specified by the Attribute's Parent_Domain. If the former is specified it is assumed to take precedence over the later. In either case the value of the Datatype element will match to one of the patterns in the following table and the corresponding XSD datatype is used as the range:

Datatype value	XSD Datatype
NUMBER(*,*)	xsd:decimal
NUMBER(*)	xsd:integer
CHAR(*) | VARCHAR(*) | BLOB	xsd:string
DATE	xsd:dateTime

There is one exception to the handling of Attributes as just described. Attributes that end with the string "-dimension" do not always have an explicit Datatype associated with itself or with its Parent_Domain. In all of these cases the value should be a numeric value (at least sometimes a number with some decimal places) and xsd:decimal is the XSD datatype that is used.

The following shows a typical DatatypeProperty produced by the translation script:

```
<owl:DatatypeProperty
    rdf:ID="action-aircraft-employment-ingress-direction-angle">
  <rdfs:comment>The numeric quotient value that represents the
     portion of a whole OBJECT-ITEM that is estimated in a specific
     ACTION-EFFECT-ITEM to have the result specified in ACTION-EFFECT.
   </rdfs:comment>
   <rdfs:domain rdf:resource="#ACTION-AIRCRAFT-EMPLOYMENT"/>
   <rdfs:range rdf:resource="&xsd; decimal"/>
</owl:DatatypeProperty>
```

6 Relationship to ObjectProperty Translation

Every Relationship in JC3IEDM is mapped into a an owl:ObjectProperty having a unique rdf:ID. Because the data model often uses relationship names for multiple relationships (e.g., has, is-the-object-of, etc.) it was necessary to either define new classes consisting of the union of the multiple classes that these relations used for their domains and ranges, or derive a naming scheme that would guarantee unique relation names. The first option was deemed undesirable because it would have meant the loss of semantic content and would have permitted relations to be formed between pairs of classes that could not occur with JC3IEDM. The second option does not suffer from this problem but it results in names that are much longer and that are not always as easy to read and write (from a human processing perspective). The simplistic (but perfectly disambiguating) approach whereby the rdf:ID for an owl:ObjectProperty is created by pre-pending the domain class with the relation name and then appending the range class was unsatisfactory to the authors as the names became exceedingly long (e.g. `OPERATIONAL-INFORMATION-GROUP-ORGANISATION-ASSOCIATION-has-OPERATIONAL-INFORMATION-GROUP-ORGANISATION-ASSOCIATION-STATUS`). To remedy this we decided to turn the class names into abbreviations using the first letter from each of its hyphened strings (e.g. OIGOA-has-OIGOAS). Unfortunately, there where a few cases where different relations resulted in the same name identifier resulting from the identical abbreviations for OBJECT-ITEM-AFFILIATION and OBJECT-ITEM-AFFILIATION. We contemplated augmenting our script to detect such cases and then disambiguating the relation names by adding an additional character to one of the identifiers. But given that there was only one case in which there was a problematic clash in class name abbreviations, we opted to simply catch this case in the XSLT script and force the abbreviation for OBJECT-ITEM-AFFILIATION to be OIAf instead of OIA. It is possible that future releases of JC3IEDM may introduce new class names that will cause additional name clashes but since we always intend to automatically check for the consistency of the generated OWL ontology we are sure to catch these cases and patch them, or reconsider a more automated fix.

In addition to defining the rdf:ID, rdfs:domain and rdfs:range for each owl:ObjectProperty defined by each relation it was also possible for a relation to be an inverse of some other relation and to have a cardinality constraint. The inverse was easy to identify from the RelationshipProps/Child_To_Parent_Phrase; all that was required to obtain the unique rdf:ID was to apply the same naming convention as for the relation's rdf:ID except for swapping the domain for the range and the range for the domain and using the Child_To_Parent_Phrase for the relation name.

Cardinality constraints were also easy to determine by testing for the presence of the Child-cardinality-code optionally specified in the Relationship_Props. The possible values for this code in the current release of JC3IEDM are:

Code	Meaning	OWL implications
PM	one or more	minCardinality="1"
ZO	zero or one	FunctionalProperty
ZM	zero, one or more	nothing

In cases where the code ia "ZO", an rdf:type of owl:FunctionalProperty is added to the property, as in the following example:

```
<owl:ObjectProperty rdf:ID="AT-is-used-in-the-definition-of-FC">
  <rdfs:domain rdf:resource="#AMMUNITION-TYPE"/>
  <rdfs:range rdf:resource="#FIRE-CAPABILITY"/>
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
</owl:ObjectProperty>
```

When the code is “PM” an owl:Restriction is created for the domain class specifying a minCardinality=“1” on the relevant owl:ObjectPropety. Here is an example of the translation of a relation with an inverse relation and a minimum cardinaility of 1:

```
<owl:ObjectProperty rdf:ID="CA-has-CAS">
  <rdfs:domain rdf:resource="#CONTEXT-ASSOCIATION"/>
  <rdfs:range rdf:resource="#CONTEXT-ASSOCIATION-STATUS"/>
  <owl:inverseOf rdf:resource="#CAS-is-ascribed-to-CA"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="CAS-is-ascribed-to-CA"/>

<owl:Class rdf:about="#CONTEXT-ASSOCIATION">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#CA-has-CAS"/>
   <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
   </owl:minCardinality>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

7 Codes to Enumeration Classes Translation

Many of the Attributes in the JC3IEDM range over values that are codes having corresponding text descriptions of their meaning. These codes are organized into Domains that have associated Validation_Rules to define the set of Valid_Values. It would be possible to write an XSLT script to run through the Domains and create an enumeration class in OWL that is populated with instances derived from the Validation_Rules. The MIP distribution, however, also has these codes organized within an XSD file making them even easier to process: for each simpleType create an owl:Class is created using the simpleType/@name as the rdf:ID and defining the class as being owl:oneOf a Collection of class instances, one instance corresponding to each restriction/enumeration element, the rdf:ID of which is set to the restriction/enumeration/@value. This would be all that was required except for the recurring problem of name clashes: many codes are used repeated as values in multiple Domains. Our work-around for this situation is to count the number of previous occurrences of the current code and if it is greater than zero we append the count to the name of the class rdf:ID. The following is an example of a very short Code enumeration class:

```
<owl:Class rdf:ID="ActionTaskOvertCovertCode">
 <rdfs:comment> The specific value that represents the property
   of an ACTION-TASK to be overt or covert.
 </rdfs:comment>

  <owl:oneOf rdf:parseType="Collection">
    <ActionTaskOvertCovertCode rdf:ID="COVERT">
```

```
          <rdfs:label>COVERT</rdfs:label>
          <rdfs:comment>
           The ACTION-TASK is to be conducted secretly.
          </rdfs:comment>
        </ActionTaskOvertCovertCode>

        <ActionTaskOvertCovertCode rdf:ID="OVERT">
          <rdfs:label>OVERT</rdfs:label>
          <rdfs:comment>
           The ACTION-TASK is to be conducted openly.
          </rdfs:comment>
        </ActionTaskOvertCovertCode>

      </owl:oneOf>
    </owl:Class>
```

8 Discussion

The JC3IEDM OWL ontology produced by our translation scripts was split up across five files: one each for the code to represent Entities, Attributes, Relations and Codes plus one top level file to import the other four. These files were submitted to ConsVISor (Versatile Information Systems' free RDF/OWL consistency checking Web service) [11],[12],[13] and (once the scripts were fully debugged) the ontology passed with no detected errors or warnings.

To indicate the relative size of the JC3IEDM OWL ontology and the code used to generate it, we provide in Table 1 a summary of quantitative characteristics such as the number of lines of code, the number of classes, the number of various types of properties and the number of unique rdf:IDs.

Table 1: Quantitative Summaries

Element/Attribute	**Quantity**
Lines of XSLT code	470
Lines of OWL code	>50,000
rdf:IDs	7932
All Classes	2921
Enumeration Classes	272
All Properties	923
ObjectProperties	617
DatatypeProperties	306
InverseProperties	116
minCardinality=1	9
FunctionalProperties	3

The resulting ontology is clearly very large and would likely be a challenge to the processing capabilities of most RDF/OWL reasoners. For our purposes, only a small subset of the ontology is ever necessary for a reasoning task and we will likely be implementing a “partial import” functionally similar to that proposed in [14].

Even though the ontology is quite large it is fair to ask how much of the actual semantics of the domain has been capture? In virtually all cases an element in the model is accompanied by a Definition that provides English text description of its meaning. It is clear from simple observation of the samples shown in this paper that there is much more meaning in these Definitions than is captured by the relationships among the classes or the restrictions placed on the use of properties. It is not at all clear, however, how one would begin the effort to extend the semantic content of the ontology and any such effort would be ad hoc without some way of validating the extensions with the authors of the text definitions. This is perhaps the most significant open problem related to the use of our proposed JC3IEDM OWL ontology.

One might argue that the class hierarchies reflect important structural relationships that have direct relevance to reasoning about inherited characteristics of instances. Unfortunately part of this type of reasoning that is built into OWL cannot be fully leveraged due to the parallel hierarchies of OBJECT-TYPE and OBJECT-ITEM. If an instance of a UNIT (a subclass of ORGANIZATIN which is a subclass of OBJECT-ITEM) has the property `is-classified-as` (in the OWL ontology this property is OI-is-classified-as-OIT) whose value is that of an instance of a TANK (a subclass of WEAPONRY-TYPE, EQUIPMENT-TYPE, MATERIAL-TYPE, and finally OBJECT-TYPE) the UNIT instance will not automatically inherit the properties of the TANK instance, even though this is clearly the intention. This failure in hierarchical reasoning results because OBJECT-TYPE and OBJECT-ITEM are related through another class (OBJECT-ITEM-TYPE) via the is-classified-as property rather than via the owl:subClassOf property. To obtain this form of is-a reasoning between OBJECT-ITEMS and OBJECT-TYPES will require going outside of OWL (e.g., implementing the logic in a rule language such as SWRL); alternatively one might contemplate the merging of the two parallel hierarchies (as suggested in [6]) but this would come at a cost as described in the next paragraph.

Although the use of parallel OBJECT hierarchies throws a monkey wrench into the use of OWL semantics it affords the feature of being able to change the type of an object overtime and in fact it is possible to have an instance be associated with multiple disjoint types at once. This capability is important when dealing with “reported” type information, such as when an enemy UNIT is reported to be a “T80 Tank” by one spot report and a “piece of Artillery” by some other report. If it was necessary, in this case, to force the UNIT to be an instance of both classes this could result in an inconsistency. In JC3IEDM, the UNIT is merely reported as being of some OBJECT-TYPE by some reporting entity through a REPORTING-DATA instance that specifies a specific time and data source. In this manner, all reported information is “reified” via their REPORTING-DATA instances making it possible for reported data to be different at different times or even be incompatible with other reported data without violating OWL’s monotonic imperative.

While the focus in this paper has been on JC3IEDM the majority of the work in developing the scripts was spent figuring out how to extract the appropriate information from the ERwin XML definition in order to generate the appropriate classes and properties. This work (and the major portion of the scripts) can be reused on any other ERwin data model for which an XML definition document has been generated.

9 Conclusions

This paper presented the authors' efforts to automatically translate JC3IEDM into an OWL ontology for use by systems that reason about C3 and situation awareness. Four XSLT scripts where written to convert 1) Entities into owl:Classes, 2) Attributes into owl:DatatypeProperties and owl:ObjectProperties, 3) Relationships into owl:ObjectProperties and 4) Codes into OWL enumeration classes. The biggest challenges were encountered in devising automated means for arriving at nearly 8000 unique rdf:ID's. In the discussion section a number of issues were raised, in particular, the shear size of the ontology, the question of semantic content in the ontology relative to the meaning encoded in the text Definitions of the Entities, Attributes and Codes, the inability to inherit from the OBJECT-TYPE hierarchy, the advantage of reifying reported information through the use of the REPORTING-DATA class and, finally, the potential use of this work for translating other ERwin based data models.

Acknowledgments. The work described in this paper was partially supported by U.S. ONR STTR Contract Number N00014-05-C-0367 and U.S. Army SBIR Contract Number W15P7T-05-C-T204.

References

1 Multilateral Interoperability Programme. *http://mip-site.org/*

2 MIP, Joint C3 Information Exchange Data Model (JC3IEDM Main), Greding, Germany, December 2005.

http://mip-site.org/publicsite/Baseline_3.0/JC3IEDM-Joint_C3_Information_Exchange_Data_Model/JC3IEDM-Main-UK-DMWG-Edition_3.0_2005-12-09.pdf

3 C. Matheus, M. Kokar, K. Baclawski and J. Letkowski, An Application of Semantic Web Technologies to Situation Awareness. In Proceedings of the 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November, 2005.

4 C. Matheus, M. Kokar, K. Baclawski, J. Letkowski, C. Call, M. Hinman, J. Salerno and D. Boulware, SAWA: An Assistant for Higher-Level Fusion and

Situation Awareness. In Proceedings of SPIE Conference on Multisensor, Multisource Information Fusion, Orlando, FL., March 2005.

5 W3C, OWL Web Ontology Language. http://www.w3.org/TR/owl-features/.

6 E. Dorion, C. Matheus and M. Kokar, Towards a Formal Ontology for Military Coalitions Operations. In Proceedings of the 10th International Command & Control Research and Technology Symposium, McLean, VA, June 2005.

7 MIP, JC3IEDM Annex I: Summary of IDEF1X Data Modelling Methodology and Notation. Greding, Germany, December 2005.

8 MIP, JC3IEDM Overview, Greding, Germany, December 2005.

http://mip-site.org/publicsite/Baseline_3.0/JC3IEDM-Joint_C3_Information_Exchange_Data_Model/JC3IEDM-Overview-UK-DMWG_Edition_3.0_2005-12-09.pdf

9 MIP, JC3IEDM Model Diagram. Greding, Germany, December 2005. *http://mip-site.org/publicsite/Baseline_3.0/JC3IEDM-Joint_C3_Information_Exchange_Data_Model/JC3IEDM-MODEL%20DIAGRAM-Edition_3.0_2005-12-09.pdf*

10 W3C Working Group Note. Representing Specified Values in OWL: "value partitions" and "value sets" Rector, A (Ed.), 17 May 2005. http://www.w3.org/TR/swbp-specified-values/

11 The ConsVISor Web Service for checking RDF/OWL ontologies. *http:www.vistology.com/consvisor*

12 K. Baclawski, M. Kokar, R. Waldinger and P. Kogut, Consistency Checking of Semantic Web Ontologies. In Proceedings of the First International Semantic Web Conference (ISWC)}, Lecture Notes in Computer Science, LNCS 2342, Springer, pp. 454--459, 2002.

13 K. Baclawski, C. Matheus, M. Kokar, J. Letkowski and P. Kogut, Towards a Symptom Ontology for Semantic Web Applications. In Proceedings of Third International Semantic Web Conference, Hiroshima, Japan, pp. 650-667, November, 2004.

14 Kendall Grant Clark, Bijan Parsia, Bryan Thompson, Bradley Bebee, A Semantic Web Resource Protocol: XPointer and HTTP. In Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, pp. 564-575, November 2004.