Large-Scale Knowledge Sharing for NASA Exploration Systems

Sidney C. Bailin[1], Ralph Hodgson[2], Paul J. Keller[3]

[1] Knowledge Evolution, Inc., 1221 Connecticut Ave NW, Suite 3B, Washington DC 20036, USA.
[2] TopQuadrant, Inc., NASA Research Park, Building 19, Suite 2043-45, Moffett Field, CA 94035, USA.
[3] NASA/Ames Research Center, Moffett Field, CA 94035, USA
{sbailin, rhodgson, paul.j. keller}@arc.nasa.gov

Abstract. NASA exploration initiatives encompass many disciplines within multiple organizations. Exchanging information without loss of meaning is a critical challenge. Differences in vocabularies are embedded in the variety of tools and systems used. This paper describes a large-scale project called NASA Exploration Initiative Ontology Models (NExIOM), which is meeting the challenge through a semantic approach to knowledge reuse.

Keywords: Ontology, semantic web, OWL, RDF, space exploration systems, NASA.

1 Introduction

NASA exploration initiatives encompass many groups and disciplines in multiple organizations. Exchanging information across discipline boundaries without loss of meaning is a critical challenge. Even people from the same discipline talk about the same concepts in different ways.

As engineers and managers we share a common vocabulary about some concepts, for example arithmetic operators, time and date, or longitude and latitude. But the engineering and scientific disciplines involved in an initiative will employ different systems of naming and connotations of meaning, often very locally defined. Differences in vocabularies, embedded in the variety of tools and systems used, include different names for variables, different data types, different units of measure, and different systems of identifiers. An engineer doing computational fluid dynamics in one part of NASA may find it difficult, without time consuming and error-prone translations, to use the results of another analysis by another engineer in a different part of NASA.

These are the motivations for a large-scale project called NASA Exploration Initiative Ontology Models (NExIOM), which takes a semantic approach to knowledge reuse within NASA. The goals of NExIOM include 1) creation and maintenance of consistent terminology; 2) enabling translations of concepts across multiple autonomous vocabularies; 3) improved specification of queries for

information retrieval; and 4) improved integration of data and interoperability of processes and tools across the lifecycle. The vision is a NASA–wide, lifecycle-aware knowledge infrastructure that will provide flexible and automated mediation of information flowing through all systems of the enterprise.

NExIOM currently consists of 126 ontologies represented in OWL, the Web Ontology Language [1]. The ontologies span 81 namespaces, each of which represents a distinct the subject area (see Table 1).

Table 1. An illustrative subset of the NExIOM namespaces.

address	command	infoasset	structure
alarm	comms	lifecycle	syseng
algorithm	component	maths	system
assembly	data	mechanics	telemetry
avionics	device	mission	temporal
bib	electrical	process	thermal
c3i	electronic	property	tool
celestialBody	equipment	security	units
chemical	hydraulics	spatial	vehicle

The outline of this paper is as follows. In Section 2 we provide a simple example of how various NExIOM ontologies interconnect with each other to support knowledge reuse. Sections 3 and 4 describe two particular ontologies, algorithms and units, in order to illustrate how we are modeling knowledge. Section 5 discusses our use of the Extensible Markup Language (XML) as a vehicle for collecting and disseminating the knowledge. Section 6 briefly touches on some related work, and in Section 7 we summarize where we have been and where we are planning to go.

2 Example of Interconnected Ontologies

As an example of the kinds of knowledge and interconnections we are modeling, consider a telemetry parameter. *Telemetry* is the broad term referring to data sent back from a spacecraft to the ground. It includes engineering and housekeeping data required to monitor the health and safety of the spacecraft and crew, guidance and navigation data, orbit and attitude data, and scientific data collected by instruments on the spacecraft.

A telemetry stream consists of a series of *parameters*. Part of the specification of a space mission is the definition of the telemetry parameters, including what they are, how they are to be represented, how they are to be interpreted, and how often they will be sent.

We use ontologies to specify this information and to connect it to other relevant knowledge about a mission. The parameters themselves are described using concepts in the Command, Control, Communications and Integration (C3I) ontology. This ontology refers, in turn, to concepts in other ontologies. For example, every telemetry parameter has a datatype. Classes for defining specialized datatypes are contained in the Datatypes ontology.

A parameter may have a calibration algorithm to convert from a raw form of data, as collected by an instrument, to a calibrated form specified in engineering units. Calibration algorithms (and many others) are described in the Algorithms ontology (see Section 3). The algorithms themselves use mathematical concepts defined in the Maths ontology.

A parameter typically measures some real-world property. These are defined in the Properties ontology. As measurements, they have units, which are defined in the Units ontology (see Section 4). The parameter has a sampling rate that is tied to the device issuing the measurements. Further information about the device may be formulated through the Device ontology. The interconnection of devices, in turn, may be described through the System ontology.

These are but a few examples of the many interconnections between concepts in the NExIOM ontologies. These structures support knowledge reuse both in software development and in operations. Multiple groups across the agency may need to write software that processes telemetry parameters for one purpose or another. The parameter descriptions provided by the ontologies can be reused either by accessing them, i.e., treating the instance data as a database, or by using the ontology classes as a basis for an object-oriented class structure in the software.

In operations, numerous scenarios require access to the same knowledge. For example, reconfiguration of a spacecraft's hardware and software may require referring to, and possibly updating the telemetry stream descriptions, and such updates are facilitated by the links to device descriptions and other related knowledge. The same kind of traversal of knowledge interconnections is necessary in fault diagnostic and repair scenarios. By locating the knowledge in one virtual location, the ontologies eliminate the need to find different pieces of the puzzle in different forms from different sources, and ensure that the pieces fit coherently together

3 Algorithms Ontology

This section and the next present details of two particular ontologies — Algorithms and Units — in order to provide a sense of the range of knowledge represented and the way in which it is organized. Algorithms obviously play a central role in software development, but also in other aspects of a mission, such as systems engineering and science. Units are fundamental to almost every aspect of mission planning, design, and operation; consistency in the use of units is crucial, yet does not happen without concerted effort given the large scale and distributed development of space exploration systems. In addition, both of these areas are relatively mature and thus fertile ground for ontological representation.

The ontology represents an algorithm in terms of its goal (the problem it solves) and the computational solution. The top level goal classes include:

- Calibration
- Commanding
- Communications
- Encoding

- Guidance, Navigation, and Control
- Orbit Determination
- Range
- Security
- Signal Processing
- Telemetry

Below this level is a richer, more specific classification.

A Goal can be expressed through a set of mathematical expressions. Some Goals will not be represented mathematically but rather by descriptive text, and for some, the descriptive text may refer to the mathematical expressions.

ComputationalSolutions are the step-by-step descriptions of how to achieve a certain computational goal. The detailed content of the solution is not explicitly represented in the ontology; rather, a bibliographic reference points to a resource that provides the details. Like the Goal, the ComputationalSolution may refer to one or more mathematical expressions and it may refer to other ComputationalSolutions upon which it builds. Time and space consumption can be specified through the *spaceBound* and *timeBound* properties, both of which take values such as Constant, Logarithmic, Exponential, etc. The *computationalModel* property allows one to specify whether the solution is, for example, deterministic or non-deterministic.

In summary, the Algorithms ontology provides classes for organizing algorithms by the problems they solve, and properties for describing their content and computational characteristics. The intention is not to require that every piece of software written anywhere in NASA be entered as an instance. Rather, the ontology provides a way to share algorithms that are known to be used across multiple projects, or that appear to have that potential.

4 Units Ontology

The Units ontology serves as a single point of reference for all units used in NASA exploration systems. As a formal knowledge structure, it also serves to disambiguate units with shared names, such as *degree* (which might refer to temperature or to angle), and units whose common abbreviations are shared (such as A for both Ampere and Angstrom).

The units ontology is actually a collection of ontologies, divided among disciplines. Classes in the ontologies represent useful conceptual groupings of units. Multiple inheritance provides distinct axes along which units are classified. The units themselves are instances of these classes.

The principle axes of classification in the Units ontology are:

- Physical vs. Resource
- Si vs. UsedWithSi vs. NotUsedWithSi
- Basic vs. Derived

Each of these is a subclass of the class Unit. The second and third axes are restricted to Physical units. The classes along each axis are disjoint from each other, but through multiple inheritance (or, more accurately, multiple typing of instances), a unit may be a member of one class along each of the axes. For example, Meter is an instance of the Si class and the Basic class.

Under both the Physical and Resource classes there are subclasses representing various knowledge disciplines. For example:

Physical	Resource
AtomicPhysics	Counting
Biology	DataLength
Chemistry	DataRate
ElectricityAndMagnetism	Event
Heat	Financial
Light	Human
Mechanics	Statistics
Radiology	
SpaceAndTime	

The Si class represents units in the standard metric system, commonly known as SI metrics [3]. Some units are not part of the SI system but are identified within the SI specification as being "used with SI." Any non-SI units that are not so identified are, by definition, "not used with SI."

A derived unit is defined in terms of the units from which it is derived, and the arithmetic operation by which it is derived. Unfortunately, OWL lacks support for arithmetic reasoning, so there is no way in OWL itself to infer arithmetically equivalent derivations. We are investigating ways to overcome this limitation.

4.2 Dimensional Semantics of Units

The dimension of a unit is the property that the unit measures, such as area, temperature, etc. It is in the treatment of unit dimensions that the formal semantics of the ontology comes into play, particularly in relation to derived units. Unit derivations provide a means to relate not just the units to each other, but also their dimensions. That is, if u_a, belonging to dimension d_a, is derived via operation O from units u_b and u_c, belonging to dimensions d_b and d_c respectively, then dimension d_a contains precisely those units derived via operation O from units in d_b and d_c, respectively. Formally:

$$\left(u_a \in d_a \;\&\; u_b \in d_b \;\&\; u_c \in d_c \;\&\; u_a = O(u_b, u_c)\right) \rightarrow \\ d_a = \left\{ x \mid \exists y,z \left(y \in d_b \;\&\; z \in d_c \;\&\; x = O(y, z)\right) \right\} \quad (1)$$

While this may seem trivial, expressing it in OWL realizes two significant benefits. First, it allows for automatic classification of derived units into the appropriate dimension. This offloads the classification task from those who may need to add units

without being thoroughly familiar with the ontology's class structure. Second, expressing these semantics in OWL allows us to employ OWL's description logic reasoning to check that manually classified units have been placed into the correct dimension. We have already found that manual classification is error-prone, even in our own maintenance of the ontology.

5 XML as a Distribution Mechanism

One of the challenges in implementing such a large-scale knowledge reuse program is the relative novelty of OWL, which raises issues of technology transfer, learning curve, tool support, and resistance. Ontologies cannot be static entities. The class structures will certainly evolve over time, and the instances must be populated by engineers from multiple NASA centers and numerous contractors.

As a pragmatic approach to addressing these challenges in the short term, we are using XML [4] as a vehicle for presenting and collecting ontological information. This use of XML is different from the standard XML serialization of RDF and OWL (there are other serializations of RDF and OWL, and we find it helpful not even to think of OWL as being built on top of XML). Rather, the XML that we are using as a distribution and collection mechanism is a straightforward representation of information that engineers may need from the ontologies, or that we, as the ontology curators, may need from them.

There are two aspects to this, corresponding to classes and instances in OWL. The simpler of these is our use of XML to represent instances. Thus, for example, all the units in the Units ontology are represented in an XML file that is generated automatically from the ontology. This XML file serves, in effect, as a controlled vocabulary for units. Here is a small portion of it:

```
<units:Unit ID="units.Bit" />
<units:Unit ID="units.BitsPerSecond"/>
<units:Unit ID="units.KilobitsPerSecond"/>
<units:Unit ID="units.MegabitsPerSecond"/>
<units:Unit ID="units.Byte"/>
<units:Unit ID="units.Mole" />
<units:Unit ID="units.PoundMole" />
<units:Unit ID="units.MolePerCubicMeter"  />
<units:Unit ID="units.Katal" />
<units:Unit ID="units.MoleKelvin" />
<units:Unit ID="units.WattPerSteradian" />
<units:Unit ID="units.Statohm" />
```

Similarly, we currently have controlled-vocabulary XML files for datatypes and for enumerations.

The more challenging use of XML is to represent the class structure of the ontologies. This enables engineers to provide us with instance information. For example, we need to collect information about the tools that are being used. In order

to do this, we must inform the engineers who are providing this information about the kinds of metadata that we require, and, in particular, with the tool classes that are defined in the tool ontology. The same holds for parameter instances and other types of knowledge.

The challenge, then, is to represent the class structure of our OWL ontologies in XML. Of course, we could use the standard serialization of OWL in XML, but that would defeat the purpose, which is to provide a representation that is familiar to most engineers and that does not require training in OWL in order to be comprehended and used.

Our first approach uses the XML Schema language [5]. Each ontology (more accurately, the ontology group corresponding to a given namespace) is automatically converted to a corresponding XML Schema. The generated schemas use the XML Schema forms of "inheritance" — type restriction and type extension — to simulate the inheritance structures in the ontologies.

Unfortunately, XML Schema does not support multiple inheritance. Indeed, type restriction and extension are not true inheritance mechanisms in the object-oriented sense. We developed several conventions to work around the limitations. For example, to deal with multiple inheritance we select a single parent class for each class (using heuristic criteria to guess the principal parent concept), and treat the other parent classes as mixins whose properties are explicitly reproduced. This makes for some rather verbose XML. It also results in overloaded property names, which we disambiguate through other conventions.

Generation of the schemas is performed by a program written in Prolog that analyzes the OWL structures and performs the subsumption reasoning necessary to represent the inheritance structures (using our conventions) in XML Schema. We generate one schema per namespace. Thus, there are 126 OWL files processed, and 81 XML Schema files generated. This takes roughly 6 minutes on a 1.66GHz Dell Latitude D620 laptop computer, traversing and reasoning about inheritance relations and property linkages across the entire set of ontologies.

There is a connection between the controlled-vocabulary files and the XML schemas. For example, the units.xml file —the controlled vocabulary for units — is an XML instance document for the XML Schema file generated for the units namespace.

We provide the XML schemas and controlled vocabularies to the engineering groups, who infer from them the metadata required to describe their engineering products. The engineering groups provide us with these metadata in the form of XML instance documents, which they validate against the corresponding XML schemas. We convert the XML instance data automatically into OWL instances of the OWL classes from which the XML schemas had been generated. This completes the round-trip, providing a way to collect knowledge and represent it in OWL without requiring fluency in OWL on the part of our clients, nor their use of special tools.

The XML schemas are not as simple or accessible as we would like. We have considered existing alternatives to XML Schema, such as Relax NG [6], but have concluded that they do not solve the problem. We are currently embarking on a new effort to represent the OWL constructs more directly. This may be thought of as a multiple-inheritance version of XML Schema, but it will likely be tailored to our specific requirements, and will not be as general as XML Schema.

6 Related Work

While there are many efforts underway to develop OWL ontologies in various fields, we know of no such effort with as comprehensive a scope as NExIOM. We have built on previous work, both in and outside NASA, to develop space-related ontologies. For example, the Jet Propulsion Laboratory has developed SWEET, the Semantic Web for Earth and Environmental Terminology [7]. An effort is currently underway at NIST to develop a standard ontology for units, building on their contributions to the UnitsML language[8].

Our work on the Algorithm ontology has been informed by the MathML standard [9] and the related OpenMath XML language [10], and especially by the Monet ontology [11]. Monet is intended to serve as a language for specifying mathematical services that may be invoked over the web. In Monet, functions are not statically defined entities to be reasoned about or displayed, but rather web services than can be invoked on input objects, returning as output the result of the function application. This is different in intent from the Algorithms ontology, but the classification of mathematical functions in Monet has proven useful to us.

7 Conclusions

NASA's goals for the next generation of manned exploration systems are ambitious, and effective knowledge reuse will be a key component of meeting the challenges. The NExIOM project represents an ontology-based approach to knowledge reuse, pushing the limits of current technology such as OWL to formalize knowledge, while employing widely accepted technologies such as XML to disseminate and collect the knowledge.

The effort is in a relatively early stage, but we have already created a comprehensive set of ontologies, generated several controlled vocabularies from them, and disseminated the resulting XML Schemas to mission projects. We have begun to receive input from those projects with which to populate the ontologies with instance data. Thus, the ongoing process of knowledge evolution and ontology maintenance begins.

References

1. Web Ontology Language (OWL), http://www.w3.org/2004/OWL/.
2. Resource Description Framework (RDF), http://www.w3.org/RDF/.
3. *American National Standard for Use of the International System of Units (SI): The Modern Metric System*. IEEE/ASTM SI 10™-2002.
4. Extensible Markup Language (XML), http://www.w3.org/XML/.
5. XML Schema, http://www.w3.org/XML/Schema.
6. RELAX NG Home Page, http://www.relaxng.org/.
7. Semantic Web for Earth and Environmental Technology, http://sweet.jpl.nasa.gov/ontology/
8. Units Markup Language (UnitsML), http://unitsml.nist.gov/Presentations/UnitsML_for_TC.pdf.