Progress on Integrating OWL and SysML

Steven Jenkins
Engineering Development Office
Systems and Software Division

Nicolas Rouquette
Systems Engineering Section
Systems and Software Division
Quick Review of Objectives

- We want to use the formalism of ontologies to represent knowledge in fields of interest to us:
  - Space flight in particular
  - Systems engineering in general
  - Fundamental phenomena underlying the above: physics, chemistry, economics, psychology, politics, probability, etc.

- We want these knowledge representation conventions to be stable and durable: independent of particular programs, projects, organizations, and software tools

- We want to customize or adapt our modeling and analysis tools to support our knowledge representation conventions
  - At least to translate to/from internal representations
  - At best to teach the tool to operate on our concepts and properties as extensions or specializations of its native counterparts
An ontology is more than a vocabulary or taxonomy
In practical terms, it’s a grammar for a particular domain of discourse
  - It sets rules for well-formed sentences
Some (simplified) well-formed sentences in Systems Engineering:
  - Curiosity has type Component.
  - Curiosity has mass 850 kg.
  - Curiosity contains Science Payload.
  - Rover Work Package supplies Curiosity.
  - MSL Project authorizes Curiosity.
Some not-well-formed sentences:
  - Curiosity supplies 850 kg.
  - Rover Work Package contains Curiosity.
  - Curiosity authorizes Science Payload.
An ontology is an agreement on usage, not a dictionary
Some agreement on usage is necessary for effective information interchange

Formal ontology standards, and OWL\(^1\) in particular, have large communities of practice with tools and training

OWL includes serialization standards; defining an OWL ontology necessarily defines standard (XML-based) knowledge interchange formats

OWL reasoners can find errors in ontology rules and facts

OWL reasoners can draw inferences (entailments) from rules and facts

OWL supports powerful query languages\(^2\) for application-specific reasoning, transformation, and reporting

OWL/RDF\(^3\) databases\(^4\) have excellent scaling properties
Some Simple Reasoning Examples

<table>
<thead>
<tr>
<th>Type</th>
<th>Given this input</th>
<th>A reasoner concludes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistency</td>
<td>“has mass” is a functional property. Curiosity has mass 850 kg. Curiosity has mass 900 kg.</td>
<td>Inconsistent: at least two facts are mutually contradictory.</td>
</tr>
<tr>
<td>Satisfiability</td>
<td>Work Package and Organization are disjoint concepts. Every Project is both a Work Package and an Organization.</td>
<td>Unsatisfiable: no Project can exists that satisfies all rules.</td>
</tr>
<tr>
<td>Rules Entailment</td>
<td>Every Spacecraft is a Component. Every Orbiter is a Spacecraft.</td>
<td>Every Orbiter is a Component.</td>
</tr>
<tr>
<td>Facts Entailment</td>
<td>Every Spacecraft is a Component. MSL Rover (an individual, not a class) is a Spacecraft.</td>
<td>MSL Rover is a Component.</td>
</tr>
</tbody>
</table>

These examples are given in “equivalent” natural language, not OWL. The purpose is to show the kinds of problems for which reasoning is useful, not to demonstrate the mechanics.
This example illustrates an actual ontology hygiene check we apply.

Literally it says “If \( p_1 \) and \( p_2 \) are distinct properties such that \( p_1 \) is a subproperty of \( p_2 \), and \( p_1^{-1} \) and \( p_2^{-1} \) exist, then report whether \( p_1^{-1} \) is a subproperty of \( p_2^{-1} \)”

```sparql
select distinct ?p1 ?p2 ?inverse_ok
where {
    { ?p1 owl:inverseOf ?p1_inverse } union { ?p1_inverse owl:inverseOf ?p1 }

    bind (exists { ?p1_inverse rdfs:subPropertyOf ?p2_inverse } as ?inverse_ok)
    filter (?p1 != ?p2)
}
```

Most important features to note: it’s short and fast.

A collection of similar queries can form the basis of a continuous validation suite for ontology and model development.
• **SysML inherits the profile mechanism from UML for customization and/or extension**
  – In fact, SysML is a profile of UML (almost)
• **We can (in principle) generate a profile by automated transformation from an OWL ontology**
  – Correspondences between OWL to SysML appear to be direct:

<table>
<thead>
<tr>
<th>OWL</th>
<th>SysML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Stereotype extending Class</td>
</tr>
<tr>
<td>Object Property</td>
<td>Stereotype extending Relationship</td>
</tr>
<tr>
<td>Datatype Property</td>
<td>Value Property</td>
</tr>
</tbody>
</table>

– **Looks can be deceiving….**

• **As a first step toward integration, we transformed UML and SysML into corresponding OWL representations**
• OMG artifacts required considerable cleanup to repair internal inconsistencies/omissions
  – All raised as issues to OMG and resolved in UML 2.4.1
  – Tools lag behind; adaptation required
• Eclipse QVTo implementation required considerable performance improvements and usability enhancements
• All transforms run under continuous integration system using Jenkins software
A Simplified View of JPL Ontologies

- **Divided into three main categories:**
  - **Foundation**
    - General concepts and properties
    - Examples at right
  - **Discipline**
    - Specializations for electrical, mechanical, etc.
    - Mostly about describing properties
  - **Application**
    - Specializations for cross-discipline use cases (e.g., orbiter, lander, observatory, etc.)
- **Each ontology may import other ontologies**
Each mapping ontology contains axioms that relate a JPL ontology to SysML/UML
- via subclass and subproperty axioms
Most class mappings are straightforward

The key is to align ontological commitments

Examples:
- `base:Container subclass of UML:NamedElement`
  - Abstract class
- `mission:Component subclass of SysML:Block`
  - Components perform Functions and present Interfaces
- `mission:Function subclass of UML:Activity`
- `mission:Interface subclass of SysML:InterfaceBlock`
  - Used as a type on a port
- `mission:Requirement a subclass of SysML:Requirement`
- `project:WorkPackage subclass of UML:Package`
  - A unit of model organization and authority delegation
Occurrences of relationships are handled differently in OWL and UML
- In OWL, the statement “A is related to B” has no identity
- Among other things, this means that a relationship can’t refer to another relationship

In UML, relationships are reified (they have identity)
OWL provides a mechanism to unify these approaches
• For a given object property, e.g., *performs*
• Create a corresponding reification class *Performs*, corresponding object properties *hasPerformsSource* and *hasPerformsTarget*, and OWL property chain axiom
• An instance of this reification class:

   «Component» spacecraft
   «Performs» nnnnn
   «Function» explore

   «hasPerformsSource»
   «hasPerformsTarget»

   is considered by OWL to imply

   «Component» spacecraft
   «Function» explore

   «performs»

   which is exactly what we want for SysML-to-OWL mapping
Generating Profiles

• Start with an ontology bundle
• Transform into an OMG-grade UML profile
  – Compliant with all applicable OMG specifications
• Transform into a tool-specific UML profile
  – In our case, MagicDraw 17.0sp4
  – Taking into account tool-specific features and limitations
  – Including user interface customization
• End-to-end process implemented in QVTo and various black-box helpers
• Runs under Jenkins continuous integration
• A major feature of the implementation is to use SPARQL to generate bundle digests that offload reasoning that’s much easier to do in SPARQL than QVTo
  – Example: user interface dialog showing legal subject-predicate-object triples for editing relationships
Helping Modelers Think In Axioms

IMCE Relationship Sentence Browser/Editor
IMCE BundleProfile (bundle=project, build=1366, revision=2374, classes=103, object properties=78, datatype=31, datatype properties=0)

Types

Subject Types:
- Type here to filter subjects
- Reset #1
  2 of 36 subjects selectable

Forward Predicates:
- Type here to filter predicates
- Reset #2
  6 of 37 predicates selectable

Object Types:
- Type here to filter objects
- Reset
  1 of 37 objects selectable

Reverse Predicates:
- Type here to filter predicates
- Reset
  13 of 35 predicates selectable
• Selected Foundation Ontology documents are cleared for public release
  – Work in progress, standard caveats apply
• Ontologies are approved for release but export-controlled
  – We are working to reverse this determination; our intent is to release the ontologies under an open source license
• Status of generated profiles is indeterminate
  – Should have same status as ontologies
  – Focus is on ontologies for now
• Enhancements to QVTo are in release approval process now
  – We intend to donate them to the Eclipse Foundation
1. OWL 2 Web Ontology Language,  
http://www.w3.org/TR/owl-overview/

2. SPARQL 1.1 Query Language,  
http://www.w3.org/TR/2012/WD-sparql11-query-20120105/

3. RDF Primer,  
http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

4. Sesame RDF Triple Store,  
http://www.openrdf.org/index.jsp