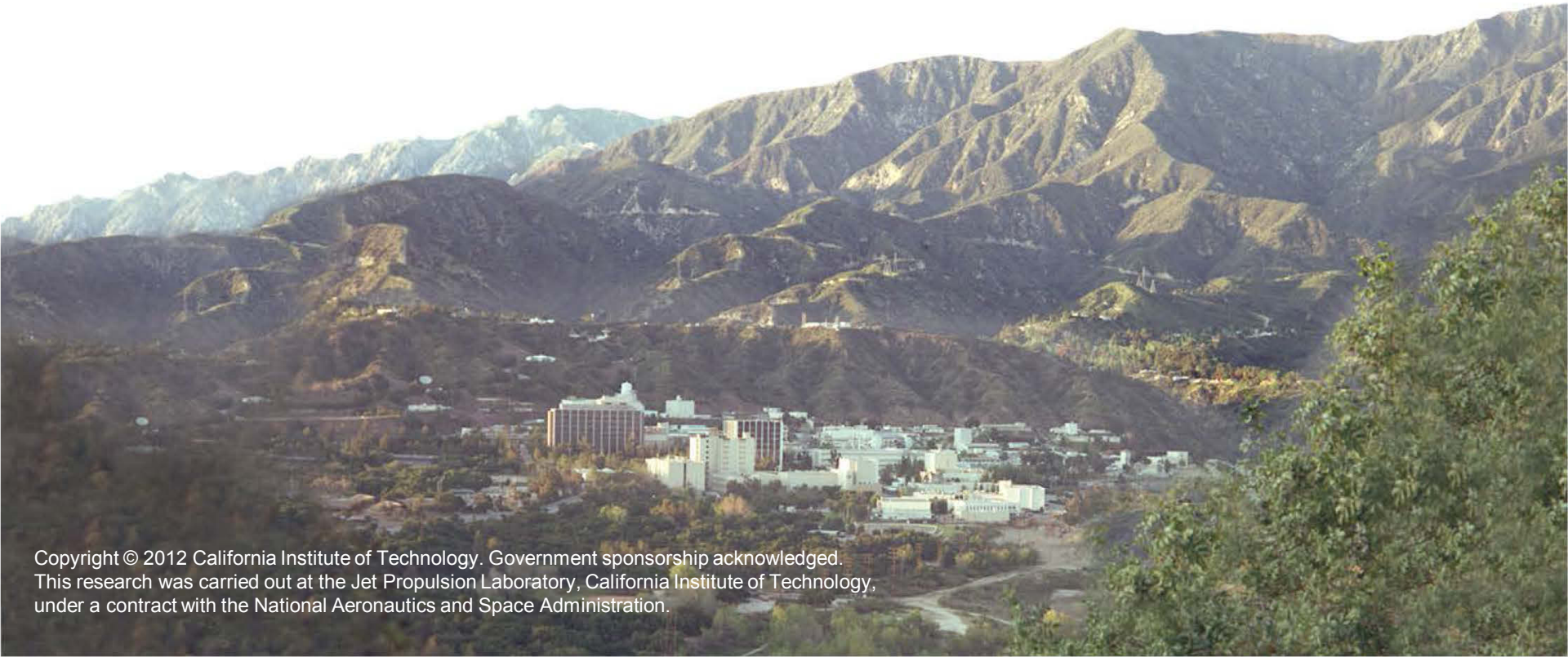
National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Progress on Integrating OWL and SysML

Steven Jenkins
Engineering Development Office
Systems and Software Division

Nicolas Rouquette
Systems Engineering Section
Systems and Software Division



Copyright © 2012 California Institute of Technology. Government sponsorship acknowledged.
This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology,
under a contract with the National Aeronautics and Space Administration.

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**

What Is An Ontology?

- **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**

Why Ontologies? Why OWL?

- ***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

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

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.

SPARQL Query Example

- **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}"**

```
select distinct ?p1 ?p2 ?inverse_ok
where {
    ?p1 rdfs:subPropertyOf ?p2 .
    { ?p1 owl:inverseOf ?p1_inverse } union { ?p1_inverse owl:inverseOf ?p1 }
    { ?p2 owl:inverseOf ?p2_inverse } union { ?p2_inverse owl:inverseOf ?p2 }

    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**

Putting OWL and SysML Together

- **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:**

OWL	SysML
Class	Stereotype extending Class
Object Property	Stereotype extending Relationship
Datatype Property	Value Property

  - **Looks can be deceiving….**
- **As a first step toward integration, we transformed UML and SysML into corresponding OWL representations**

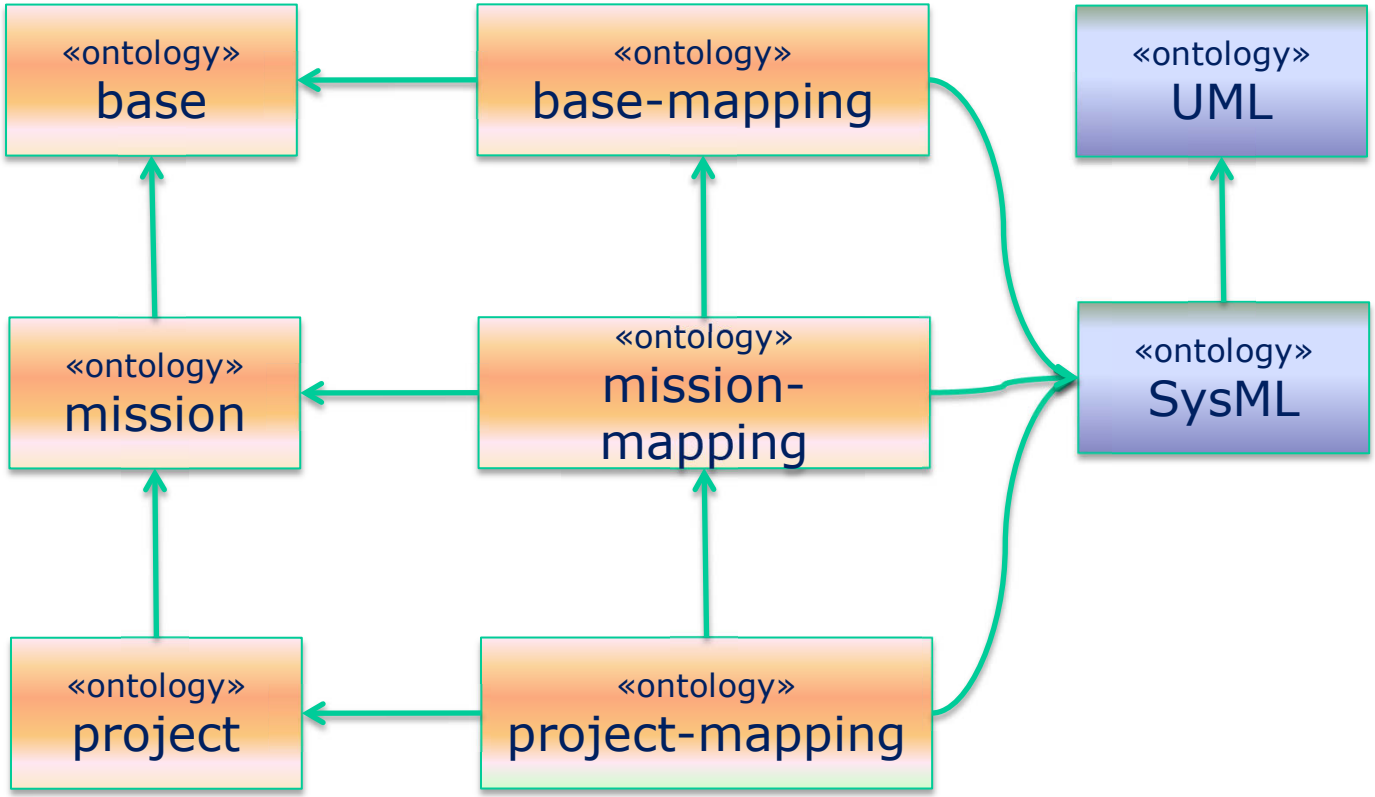
Constructing OMG Ontologies by Transform

- **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**

Relating JPL and OMG Ontologies



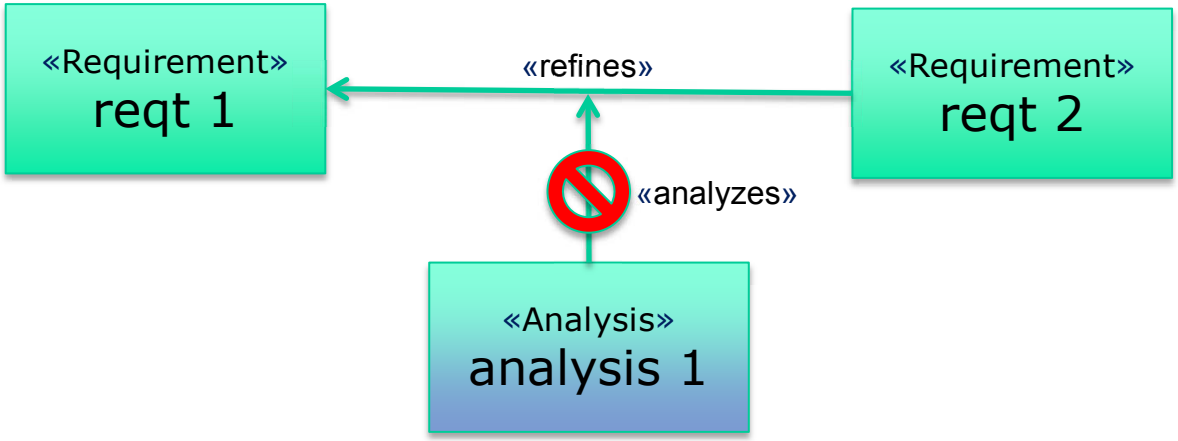
- **Each mapping ontology contains axioms that relate a JPL ontology to SysML/UML**
  - **via subclass and subproperty axioms**

Mapping Classes

- **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**

Mapping Object Properties

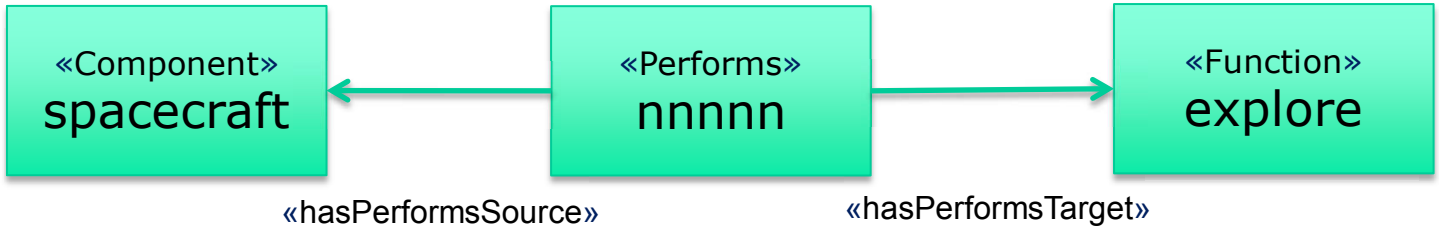
- **Occurrences of relationships are handled differently in OWL and UML**
  - **In OWL, the statement "A is related to B" has no identity**
  - **Among other things, this means that a relationship can't refer to another relationship**



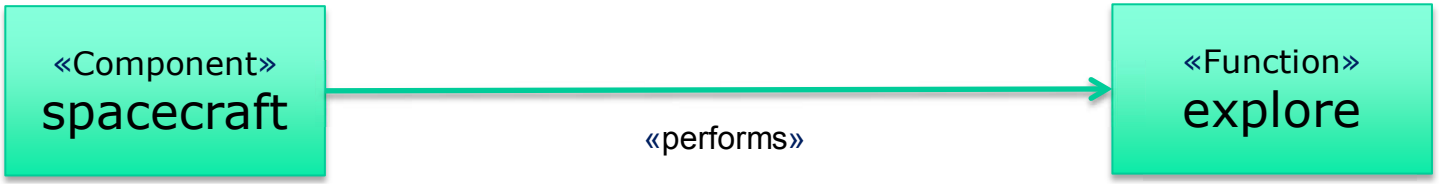
- **In UML, relationships are reified (they have identity)**
- **OWL provides a mechanism to unify these approaches**

Object Property Reification

- **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:**



is considered by OWL to imply



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 Relationship Sentence Browser/Editor

IMCE BundledProfile {bundle=project, build=1366, revision=2374, classes=103, object properties=78, datatypes=31, datatype properties=0}

Types

Subject Types:

Type here to filter subjects | Reset #1

2 of 36 subjects selectable

- analysis:Assumption
  - analysis:Fact
- analysis:Characterization
  - analysis:Analysis
    - analysis:CostEstimate
    - analysis:DrivingRequirementsAnalysis
    - analysis:KeyRequirementsAnalysis
    - analysis:TradeStudy
  - analysis:Quantity
- analysis:Criterion
- analysis:Metric
- base:Package
  - project:Program
  - project:Project
  - project:WorkPackage
- mission:Component
- mission:Environment
- mission:Flow
- mission:Function
- mission:Interface
- mission:InterfaceJunction
- mission:Item
  - mission:MaterialItem
  - mission:Message
- mission:Mission
- mission:Requirement
- project:Concern
  - mission:Objective
- project:Deliverable
  - project:DevelopmentState
- project:Facility
- project:Organization
- project:Policy
- project:Process
- project:Product
- project:Stakeholder

Forward Predicates:

Type here to filter predicates | Reset #2

6 of 37 predicates selectable

- base:contains
- mission:deploys
- mission:emits
- mission:induces
- mission:influences
- mission:ingests
- mission:invokes
- mission:joins
- mission:performs
- mission:presents
- mission:pursues
- mission:receives
- mission:refines
- mission:sends
- mission:specifies
- mission:traverses
- project:authorizes

Reverse Predicates:

Type here to filter predicates | Reset

13 of 35 predicates selectable

- analysis:isAnalyzedBy
- analysis:isCharacterizedBy
  - analysis:isExplainedBy
    - analysis:isValidatedBy
- analysis:isCriterionFor
- analysis:isMeasuredBy
- analysis:isQuantifiedBy
- base:isAggregatedIn
- base:isContainedIn
- mission:isDeployedBy
- mission:isEmittedBy
- mission:isInducedBy
- mission:isInfluencedBy
- mission:isIngestedBy

Object Types:

Type here to filter objects | Reset

1 of 37 objects selectable

- analysis:Assumption
  - analysis:Fact
- analysis:Characterization
  - analysis:Analysis
    - analysis:CostEstimate
    - analysis:DrivingRequirementsAnalysis
    - analysis:KeyRequirementsAnalysis
    - analysis:TradeStudy
  - analysis:Quantity
- analysis:Criterion
- analysis:Metric
- analysis:QuantityValue
- base:Package
  - project:Program
  - project:Project
  - project:WorkPackage
- mission:Component
- mission:Environment
- mission:Flow
- mission:Function
- mission:Interface
- mission:InterfaceJunction
- mission:Item
  - mission:MaterialItem
  - mission:Message
- mission:Mission
- mission:Requirement
- project:Concern
  - mission:Objective
- project:Deliverable
  - project:DevelopmentState
- project:Facility
- project:Organization
- project:Policy
- project:Process
- project:Product
- project:Stakeholder

IMCE Diagram
IMCE Browser/Editor

Elements & Relations

Status | Close

Status of Products

- **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**

References

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