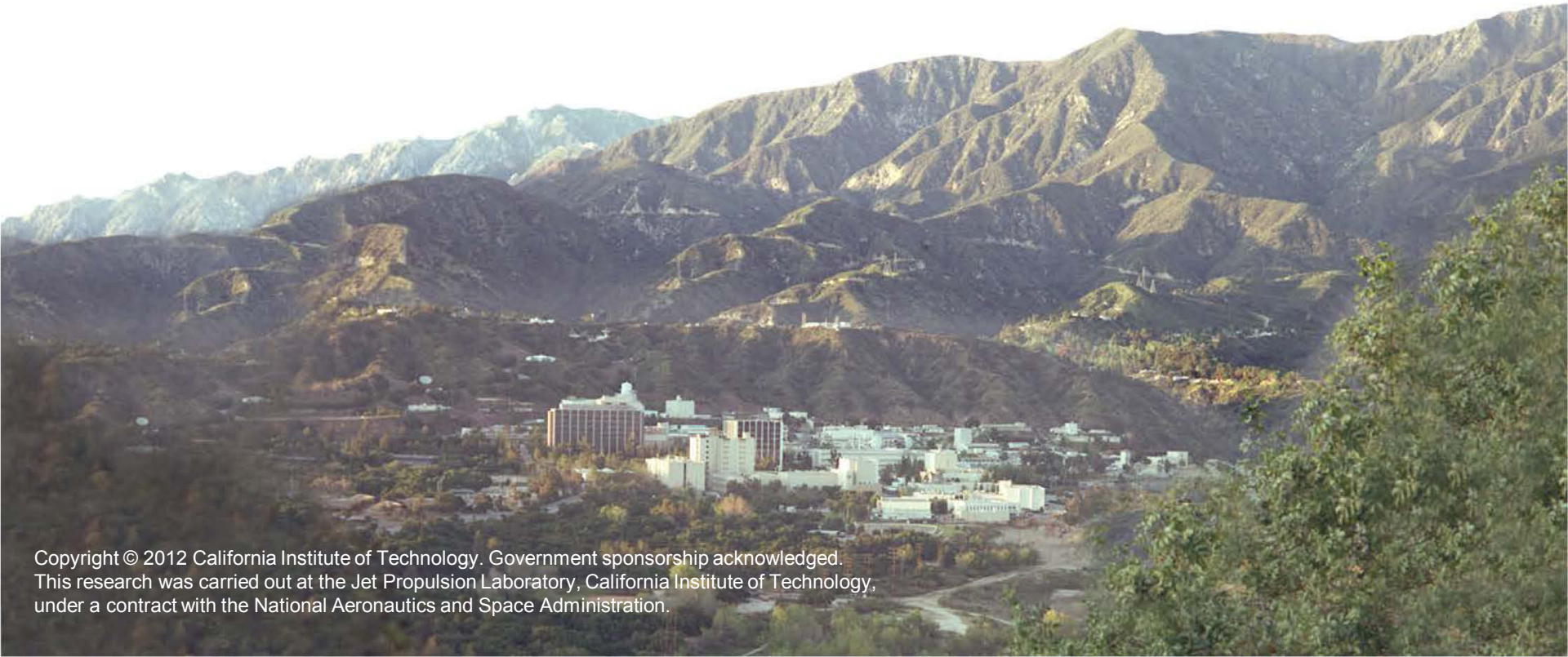
National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Semantically-Rigorous Systems Engineering Using SysML and OWL

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.

Overview

- **SysML is a widely-accepted graphical language for systems engineering**
  - **Defined by industry-consensus OMG specification**
  - **With tool support from multiple suppliers**
  - **Supported by books, training materials, consultants**
  - **Strengths: graphical notation, systems engineering orientation**
- **OWL is a widely-accepted knowledge representation language**
  - **Defined by industry-consensus W3 specification**
  - **With tool support from multiple suppliers**
  - **Supported by books, training materials, consultants**
  - **Strengths: logical formalism, reasoning, generality**
- **Our objective is to combine the two in order to ally their strengths**

Logical Reasoning in Systems Engineering

- **Logical reasoning is essential to systems engineering, but we don't often refer to it by name**
- **Examples:**
  - **Requirements Tracing: every requirement except those levied against the top-level component must have a parent in the next higher component**
  - **Interface Compatibility: every pair of interfaces joined must be of compatible type**
  - **Viewpoint Consistency: the system realization viewpoint (i.e., the thing to be acquired) must properly reflect the design choices made in multiple complementary viewpoints, e.g., thermal, mechanical, electrical, test and verification, etc.**
- **Each example has aspects that can be written in the form of a predicate that could be evaluated by a reasoner**

OWL Ontologies for Systems Engineering

- **We have developed a set of OWL ontologies for systems engineering**
- **They define broad concepts and properties in hierarchical categories:**
  - **Foundation**
    - **Base**
    - **Mission**
    - **Analysis**
    - **Project**
  - **Discipline**
    - **Electrical**
    - **Mechanical**
    - **etc.**
  - **Application**
    - **Rover (example only)**
- **Remainder of the presentation is about Foundation only**

Example Concepts and Properties

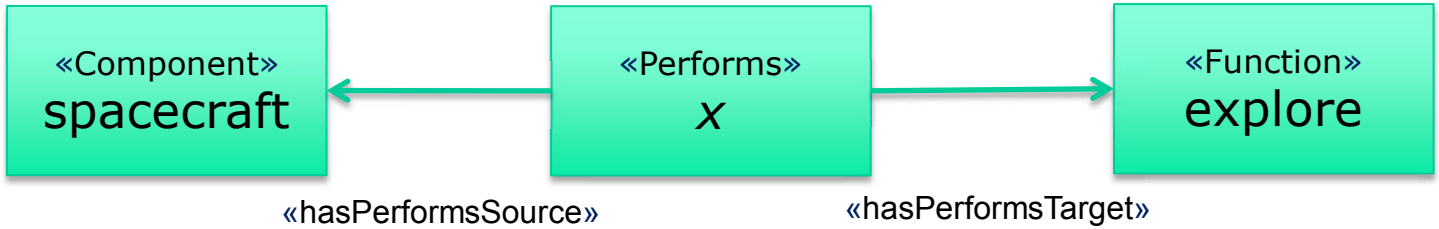
- **Mission**
  - **Component**
    - **contains Component, performs Function, presents Interface**
  - **Function**
    - **invokes Function**
  - **Interface**
    - **contains Interface**
  - **Junction**
    - **joins Interface**
  - **Requirement**
    - **specifies { Component, performs, presents }, refines Requirement**
- **Project**
  - **WorkPackage**
    - **authorizes anything, supplies Component**

Embedding into SysML

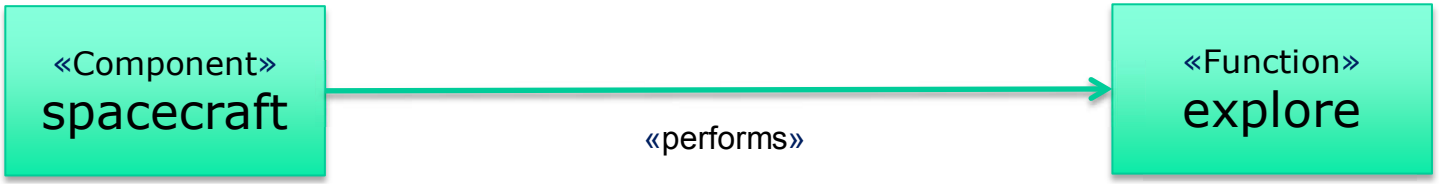
- **We create (by transformation) OWL ontologies for SysML (and UML and other dependencies)**
  - **These ontologies express certain features (including class taxonomy) of SysML/UML in OWL**
- **We can write embedding axioms that relate our concepts and properties to the best match in SysML**
  - **e.g., mission:Component owl:subClassOf SysML:Block**
- **Embedding classes in this way is straightforward**
- **Embedding object properties is more complex**
  - ***owl:inverseOf* relationship requires Extended MOF semantics**
  - **Occurrences of object properties are not reified in OWL**
    - **There's no way to say "this requirement specifies the *performs* relationship between this component and this function"**
    - **A particular occurrence of *performs* has no identity**

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:**



implies (by effect of the property chain axiom):



which is what we want for SysML-to-OWL transformation

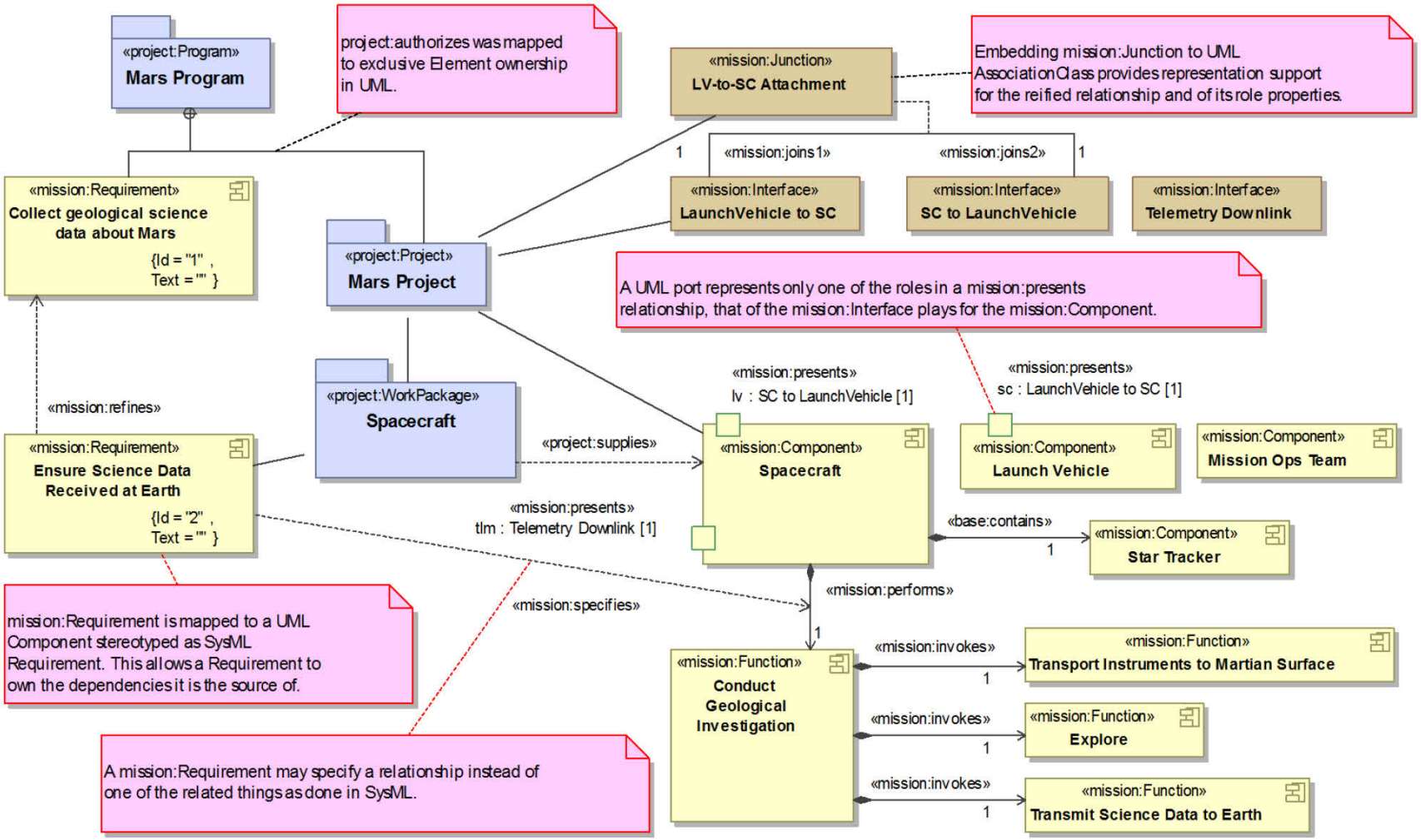
Validating Ontologies and Embedding

- **We subject our ontologies (including embedding axioms) to a battery of tests**
  - **Consistency**
    - **no axioms contradict other axioms**
  - **Satisfiability**
    - **every class can be nonempty**
  - **Well-Formedness**
    - **every class embedded in SysML**
    - **every property embedded in SysML**
    - **domain and range of super/subproperty pairs consistent**
    - **every object property has reification apparatus**
    - **consistent embedding of super/subclass pairs**
    - **etc.**
- **These tests run under a continuous integration system (Jenkins, no relation) whenever an ontology changes**
- **Current ontology set yields over 45 000 test cases**

Generating SysML Profiles from Ontologies

- **We load our ontologies into a Sesame repository and use SPARQL queries to create digests that simplify profile construction**
  - **Object property range after applying range restriction**
  - **Valid predicates for each subject class**
  - **Valid object classes for each subject/predicate pair**
  - **etc.**
- **A transformation in Operational Query/View/Transform (QVTo) operates on digests to produce profiles**
- **The QVTo transforms also produce MagicDraw-specific user interface customization**
  - **To assist the modeler in complying with profile rules**

Building Profiled System Models



Processing and Validating Profiled Models

- **Models with profiles applied can be transformed from SysML back into OWL using QVTo**
- **In essence, we extract the ontological commitments from the profiled model**
- **The OWL representation is suitable for**
  - **Validation for well-formedness**
  - **Validation for adherence to local business rules, e.g.,**
    - **every Component performs at least one Function**
    - **every Function is performed by exactly one Component**
    - **every *presents* relationship is specified by at least one Requirement**
    - **etc.**
  - **Feature extraction and transformation for specialized analysis tools, e.g.,**
    - **Maple, Mathematica**
  - **Long-term archival, data warehousing**

Conclusions and Future Work

- **Conclusions**
  - **Transforming SysML/UML specifications to OWL and embedding our ontologies has proven flexible**
  - **Pre-processing ontologies with SPARQL simplifies profile generation code**
  - **QVTo has proven to be powerful**
    - **once some performance issues were addressed**
  - **SPARQL and Sesame are powerful for analyzing and transforming SysML models with our profiles applied**
- **Future Work**
  - **Adding support for datatype properties**
  - **Enhancing the SysML-to-OWL transformation**
  - **Developing analysis tooling in the OWL domain**
  - **Developing discipline and application ontologies that extend foundation concepts, e.g.,**
    - **electrical, mechanical, verification, etc.**