An Introduction to Knowledge Representation and Ontology Development for Systems Engineers

Elisa Kendell
Steven Jenkins

Abstract

Effective communication requires a common vocabulary. An ontology provides a description of the terminology, concepts and relationships for a particular area of interest. An ontology may be viewed as a declarative encoding of the meaning of the domain vocabulary terms, thus making it a key to enabling communication. For systems that are used by people whose understanding of a domain is not necessarily consistent, an explicit description of the important terms can be extremely useful.

Many commercial companies have successfully deployed applications with increasing use of semantics such as taxonomy-based search and navigation services. Rule-based manufacturing, product configuration, and financial services systems have been relatively common in those industries for many years. Fewer organizations have successfully deployed semantically rich systems that incorporate ontology-based metadata, sophisticated reasoning and explanation support. The technology has been around for decades, though its use for web-based applications is relatively recent, and it remains difficult for some people to understand, let alone use effectively.

This tutorial provides an overview of the knowledge representation landscape and attempts to de-mystify some of the "black art" of ontology development. We will outline basic methodology steps developed from a combination of

- Domain analysis methodology from software engineering
- IDEF methods developed for the US Department of Defense
- Best practices developed through extensive experience and lessons learned, with a focus on problems in software and systems engineering

Examples from systems engineering will be provided, with emphasis on ontology development in UML using the Ontology Definition Metamodel, applications that lend themselves to vocabulary development in the Web Ontology Language (OWL), and use of the these technologies together with models developed using the OMG's Systems Modeling Language (SysML).

An Introduction to Knowledge Representation and Ontology Development for Systems Engineers

Elisa Kendell
Steven Jenkins

Biographies

Elisa KENDALL founded Sandpiper Software in October 1995 and has over 25 years professional experience in the design, development and deployment of enterprise-scale information management systems for communications, high technology, and aerospace applications. Since then, she has been instrumental in bridging emerging technologies from professional software engineering disciplines, such as the Object Management Group (OMG)'s Model Driven Architecture methodology, which provides a basis for automating metadata management, and knowledge representation and reasoning (Semantic Web) technologies. Ms. Kendall is principal architect of Sandpiper's UML-based knowledge representation, ontology analysis, and policy-based application framework, and has developed rigorous methodologies for domain assessment and ontology design and development. She is an active participant in the W3C OWL Working Group, ISO metadata standards development, co-chair of the OMG Ontology PSIG, and a key contributor to standards such as the Object Management Group's Ontology Definition Metamodel (ODM). She holds an MA in Linguistics from Stanford University and a BS in Mathematics and Computer Science from the University of California, Los Angeles. Nationality: USA.

Steven JENKINS is a Principal Engineer in the Systems and Science Division at the Jet Propulsion Laboratory, currently supporting JPL's Integrated Model-Centric Engineering Initiative. He previously worked on the system engineering teams for NASA Project Constellation and Project Prometheus. He has also managed the Institutional Computing System Engineering and Architecture Office and the Enterprise Information System Project. Dr. Jenkins holds a B.S. in Mathematics from Millsaps College, an M.S. in Applied Mathematics from Southern Methodist University, and a Ph.D. in Electrical Engineering (Control Systems) from the University of California, Los Angeles. He was awarded the NASA Outstanding Leadership Medal in 1999.

An Introduction to Knowledge Representation and Ontology Development for Systems Engineers

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

sandpiper software

Elisa Kendall, Sandpiper Software
Chief Executive Officer

Steven Jenkins
Principal Engineer, Systems and Software Division
Jet Propulsion Laboratory
California Institute of Technology

20th Annual INCOSE International Symposium
July 10-16, 2010

Agenda

- Part 1: Introduction to Knowledge Representation & Ontology
- Part 2: Ontology Development in UML: The Ontology Definition Metamodel
- Part 3: Integrating Ontologies & Systems Engineering via SysML

Part 1: Introduction to Knowledge Representation & Ontology

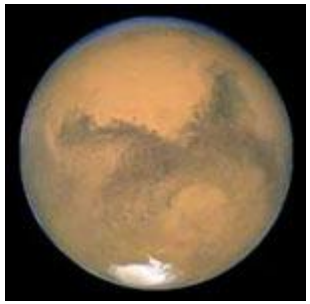
- A little history
- A few definitions
- Layers of abstraction & conceptual modeling
- Classifying ontologies
- A little methodology

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

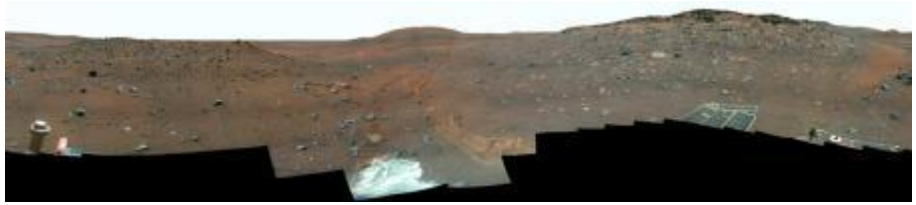
sandpiper software

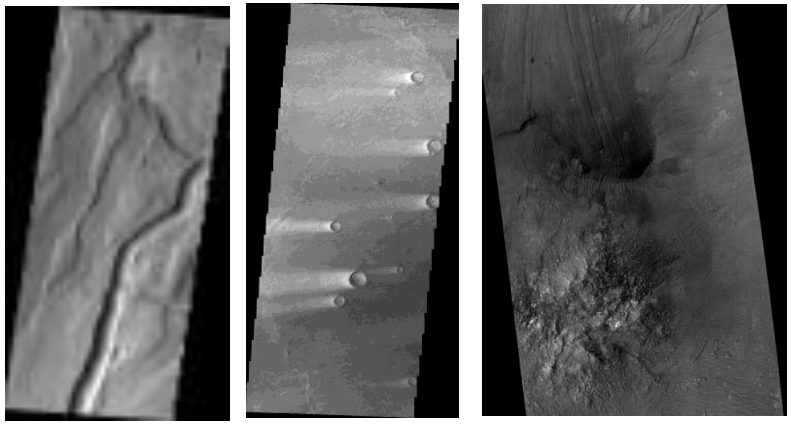
Content Management for Long-Term Retention & Reuse



Mars was photographed by the Hubble Space Telescope in August 2003 as the planet passed closer to Earth than it had in nearly 60,000 years. Image Credit: NASA, J. Bell (Cornell U.) and M. Wolff (SSI)

A sunset on Mars creates a glow due to the presence of tiny dust particles in the atmosphere. This photo is a combination of four images taken by Mars Pathfinder, which landed on Mars in 1997. Image credit: NASA/JPL





Recent images from instruments on board the Mars Reconnaissance Orbiter take much more detailed, narrower views of specific features of the Martian surface. Image credit: NASA/JPL

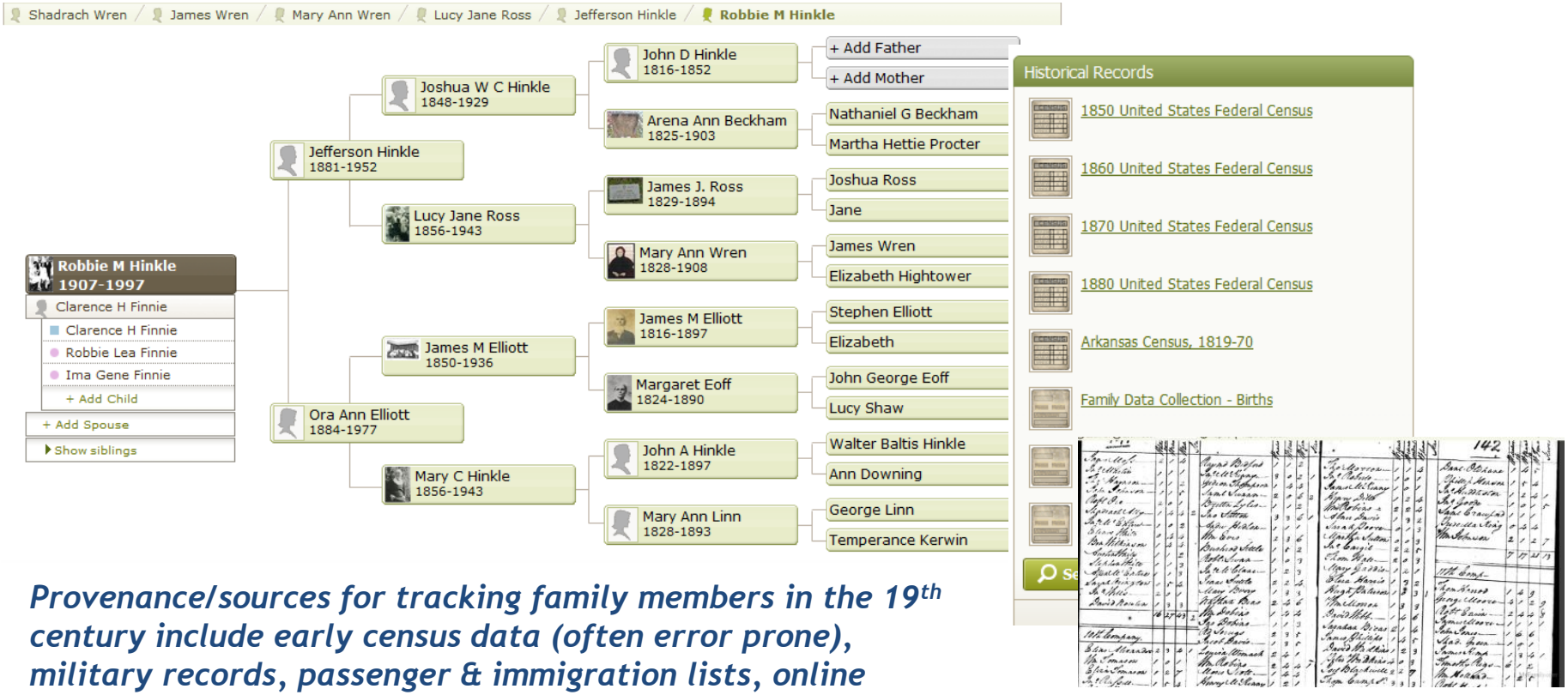
The Planetary Data Store (PDS) is a distributed repository of 40+ years' imagery & data taken by a range of instruments on many diverse missions, available for scientific research.

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

sandpiper software

Search & Retrieval Over Diverse Data Sources



Provenance/sources for tracking family members in the 19th century include early census data (often error prone), military records, passenger & immigration lists, online documents (e.g., county histories, church histories, etc.)

- *Historical/forensic research requires cross-domain search of a wide variety of resources within a given geo-spatial/temporal context*
- *Similar capabilities are essential for business intelligence, law enforcement, government applications – all require terminology reconciliation*

Historical Context

- Knowledge Representation
  - Cross-disciplinary field with historical roots in philosophy, linguistics, computer science, and cognitive science
  - Goal is to represent the meaning of knowledge *unambiguously*, so that it can be understood, shared, and used by computational agents acting on behalf of people to accomplish some task

- Philosophical origins
  - Socrates questioning, Plato's studies of epistemology – the nature of knowledge
  - Aristotle's shift to terminology, development of logic as a precise method for reasoning about knowledge
  - Arguments for the existence of God dating back to Anselm of Canterbury
  - Medieval theories of reference and of mental language, Scholastic logic

Plato and Aristotle at the School of Athens, by Raphael

Definitions

- ***An ontology*** is a specification of a conceptualization. – *Tom Gruber*
- ***Knowledge engineering*** is the application of logic and ontology to the task of building computable models of some domain for some purpose. – *John Sowa*
- ***Artificial Intelligence*** can be viewed as the study of intelligent behavior achieved through computational means. Knowledge Representation then is the part of AI that is concerned with how an agent uses what it knows in deciding what to do –*Brachman and Levesque - KR&R*
- ***Knowledge representation*** means that knowledge is formalized in a symbolic form, that is, to find a symbolic expression that can be interpreted. – *Klein and Methlie*
- The task of ***classifying*** all the words of language, or what's the same thing, all the ideas that seek expression, is the most stupendous of logical tasks. Anybody but the most accomplished logician must break down in it utterly; and even for the strongest man, it is the severest possible tax on the logical equipment and faculty. – *Charles Sanders Peirce*, letter to editor B. E. Smith of the *Century Dictionary*

Logic and Ontology

- Predicate logic is harder to read than the original English, but is more precise:

  Every semi-trailer truck has at least 3 axles.

  $(\forall x)(((SemiTrailerTruck(x) \wedge (\exists y)(SemiTrailer(y) \wedge (hasPart\ (x,y))) \wedge$
  $(SemiTrailerTruck(x) \wedge (\exists z)(TractorUnit(z) \wedge (hasPart\ (x,z))))$
  $\supset (\exists s)(set(s) \wedge (count(s,(\geq 3))$
  $\wedge (\forall w)(member(w,s) \supset (Axle(w) \wedge hasPart(x,w))))).$

- Logic is a simple language with few basic symbols.
- The level of detail depends on the choice of predicates – these predicates represent an *ontology* of the relevant concepts in the domain.
- Different choices of predicates represent different *ontological commitments*.

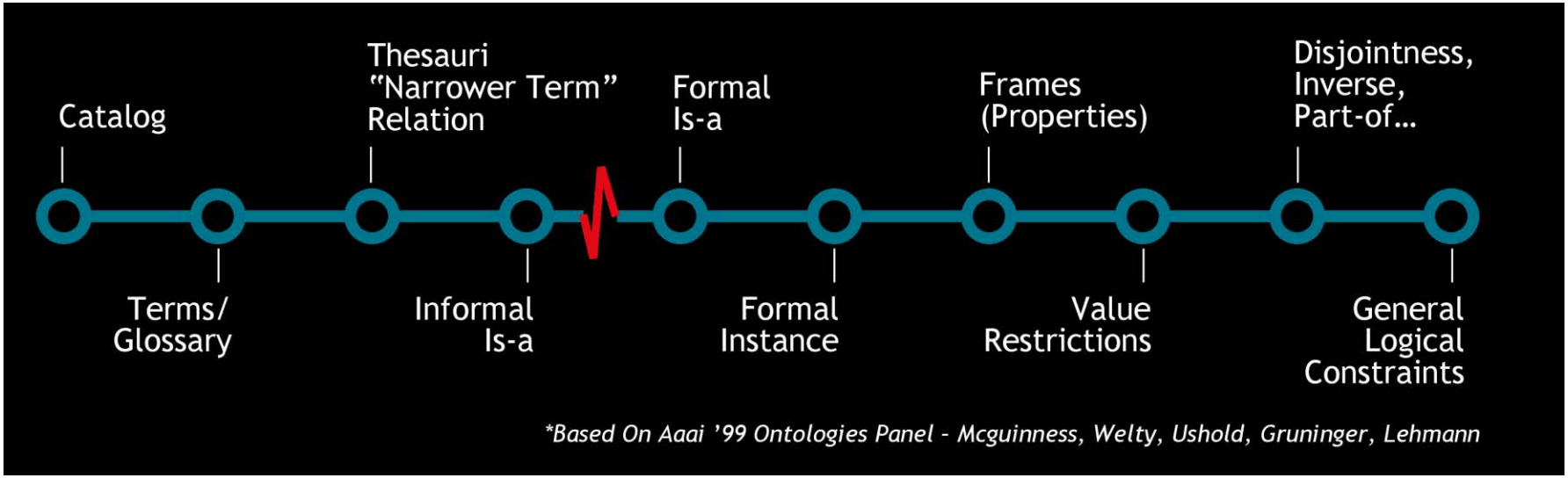
** Derived from Knowledge Representation: Logical, Philosophical, and Computational Foundations, John F. Sowa, Brooks/Cole, Pacific Grove, CA, 2000.*

What is an Ontology?

An ontology specifies a rich description of the

- Terminology, concepts, nomenclature
- Properties explicitly defining concepts
- Relations among concepts (hierarchical and lattice)
- Rules distinguishing concepts, refining definitions and relations (constraints, restrictions, regular expressions)

relevant to a particular domain or area of interest.



Ontology-based Technologies

- Ontologies provide a *common vocabulary* for use by independently developed resources, processes, services
- *Agreements* among organizations sharing common services can be made with regard to their *usage*; the *meaning* of relevant concepts can be *expressed unambiguously*
- By *composing / mapping* ontologies and *mediating* terminology across participating events, resources and services, independently-developed services can work together to share information and processes consistently, accurately, and completely
- Ontologies also ensure
  - Valid conversations among agents to collect, process, fuse, and exchange information
  - Accurate searching by ensuring context using concept definitions and relations instead of/in addition to statistical relevance of keywords

Features of KR Languages

- Vocabulary – a collection of symbols
  - Domain-independent *logical symbols* (*e.g.*, $\forall$ or $\supset$)
  - Domain-dependent *constants*, identifying individuals, properties, or relations in the application domain or *universe of discourse*
  - *Variables*, whose range is governed by quantifiers
  - *Punctuation* that separates or groups other symbols
- Syntax – *formation rules* that determine how symbols can be combined in well-formed expressions; rules may be stated in a linear grammar, graph grammar, or independent abstract syntax
- Semantics – a *theory of reference* that determines how the constants and variables are associated with things in the universe of discourse, and a *theory of truth* that distinguishes true statements from false statements
- Rules of Inference – rules that determine how one pattern can be inferred from another; if the logic is *sound*, the rules of inference must preserve truth as determined by the semantics

Description Logics

- A family of logic-based Knowledge Representation formalisms
  - Descendants of semantic networks and KL-ONE
  - Describe domain in terms of *concepts* (classes), *roles* (relationships), and *individuals* (instances)
- Distinguished by
  - Formal semantics
    - *Decidable* fragments of FOL
    - Closely related to Propositional, Modal, and Dynamic Logics
  - Provision of inference services
    - *Sound and complete decision procedures* for key problems
    - Implemented systems (*highly optimized*)
- Applications include
  - Configuration – product configurators, consistency checking, constraint propagation, first significant industrial application (*e.g.*, CLASSIC)
  - Ontologies – ontology engineering (design, maintenance, integration), reasoning with ontology-based mark-up, service description and discovery
  - Databases – consistency of conceptual schemata, schema integration, query subsumption (w.r.t. conceptual schemata)

Knowledge Bases, Data Bases, & Ontology

- An ontology is a conceptual model of some aspect of a particular universe of discourse (or of a domain of discourse)
- Typically, ontologies contain only “rarified” or “special” individuals, *metadata*, representing elemental concepts critical to the domain
- A knowledge base is the persistent repository for
  - The ontology and metadata representing the relevant individuals, facts, and rules about how they can be combined or relate to one another
  - The metadata only – in some applications and frameworks the ontology is separately maintained
- Most inference engines require in-memory deductive databases for efficient reasoning (including commercially available reasoners)
- A knowledge base may be implemented in a physical, external database, such as a relational database, but reasoning is typically done on a subset (partition) of that knowledge base in memory

Reasoning & Truth Maintenance

- Reasoning is the mechanism by which the assertions one makes in an ontology and related knowledge base are evaluated by an inference engine.
- In classical logic, the validity of a particular conclusion is retained even if new information is received.
- This may change if some of the preconditions are actually hypothetical assumptions invalidated by the new information.
- The same idea applies for arbitrary actions – new information can make preconditions invalid.
- Generally, there are two issues that a reasoner must address:
  - If some conclusion is invalid, which other conclusions are also invalid?
  - If some action cannot be performed, which others are at risk?
- The “housekeeping” associated with tracking the threads that support answering these questions is called *truth maintenance*.

Negation

- If all new information is “positive”, then all prior conclusions should remain valid.
- Problems are introduced if new information negates a prior assumption, causing it to be withdrawn.
- What does it mean to negate (withdraw) an assumption?
  - Conclusive information is not available?
  - The assumption cannot be proven?
  - The assumption is not provable using certain methods?
  - The assumption is not provable given a fixed quantity of time?
- The answer can result in different definitions of negation and differing interpretations by non-monotonic reasoners.
- Solutions include chronological and “intelligent” backtracking algorithms, heuristics, circumscription algorithms, justification or assumption based retraction, etc., depending on the reasoner and methods used for truth maintenance.
- Reasoning efficiency is dependent, in part, on the algorithms applied for truth maintenance.

Explanations and Proofs

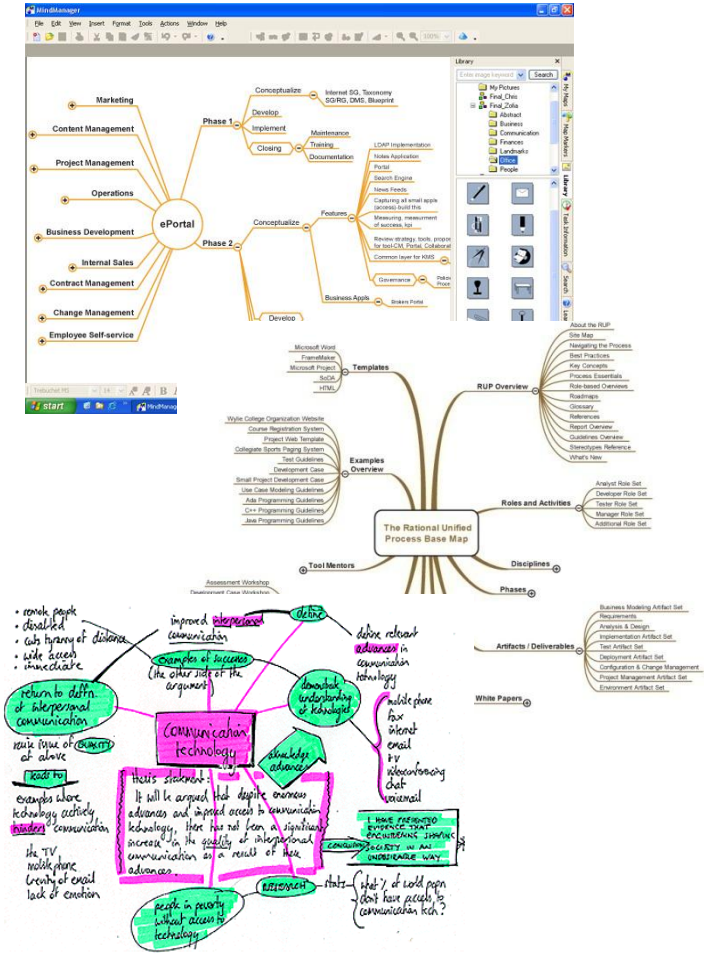
- When a particular conclusion is reached by a reasoner, many users and applications want to understand *why?*
- Primary motivations include interoperability, reuse, and trust
- Especially when web-based information is involved, understanding the *provenance* of the information /results is crucial
  - What information sources were used (source)
  - How recently they were updated (currency)
  - How reliable these sources are (authoritativeness)
  - Was the information directly available or derived, and if derived, how (method of reasoning)
- Methods used to explain why a reasoner reached a particular conclusion include explanation generation and proof specification

Conceptual Modeling

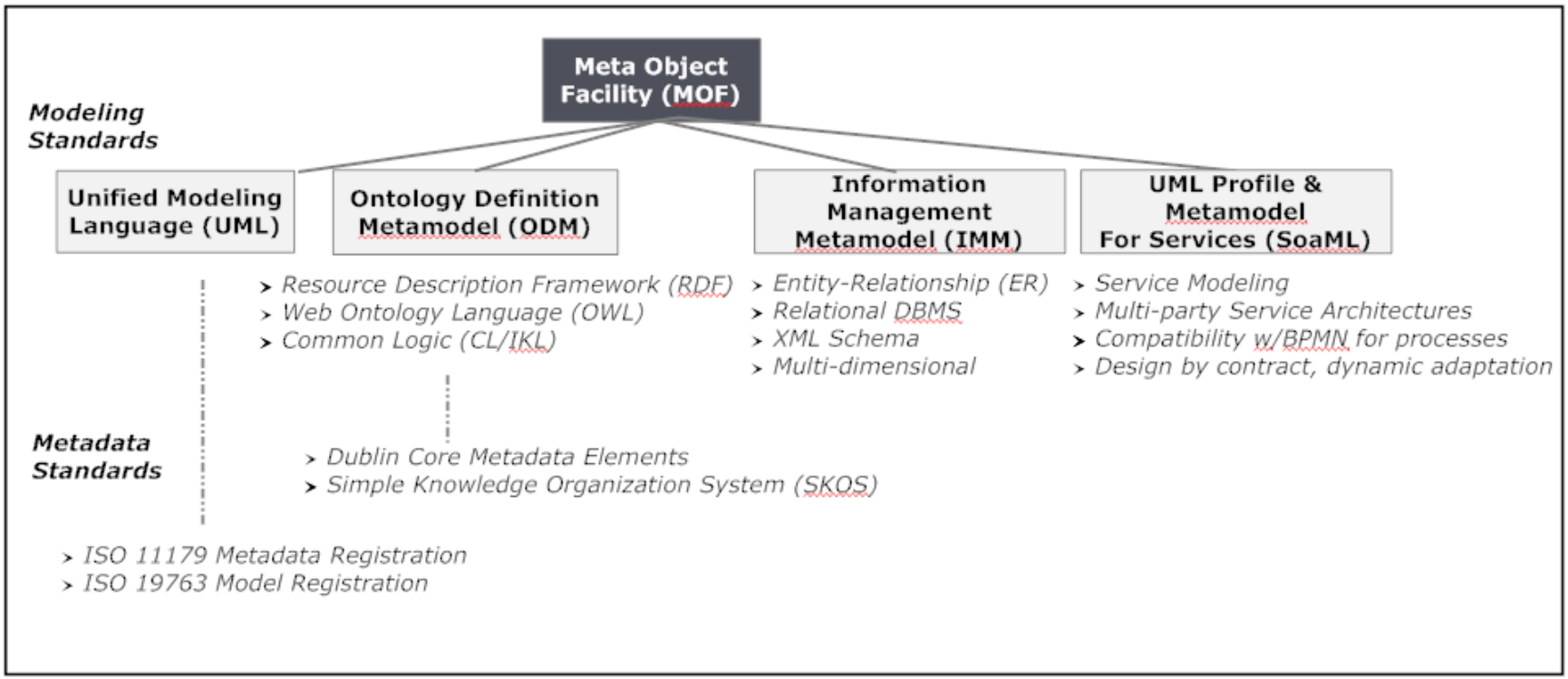
- "A data model describes data, or database schemas – an ontology describes the world"
  - *Adam Farquhar, "Ontology 101", Stanford University, 1997*
- View resources their relationships as they "*are*", or "*are … with respect to some application or domain*", not as they are defined in databases, tag systems or by programmers
- Librarians, linguists, business people with domain knowledge (subject matter experts, SMEs) – classify knowledge differently from someone interested in optimization of algorithms, or shoehorning information into an existing framework, coding system, or application
- Shortcuts at the top levels do not help; automation and mapping among ontologies and terminology at lower levels provides significant benefit

Conceptual Modeling

- Definitions range from high-level mind mapping and brainstorming … to detailed collaboration, dialog, and information modeling to support knowledge sharing
- Tools are equally diverse, from inexpensive brainstorming tools and university shareware to sophisticated ontology and software model development environments
- Common capabilities include
  - "drawing a picture" that includes concepts and relationships between them
  - producing sharable artifacts, that vary depending on the tool – often including web sharable drawings



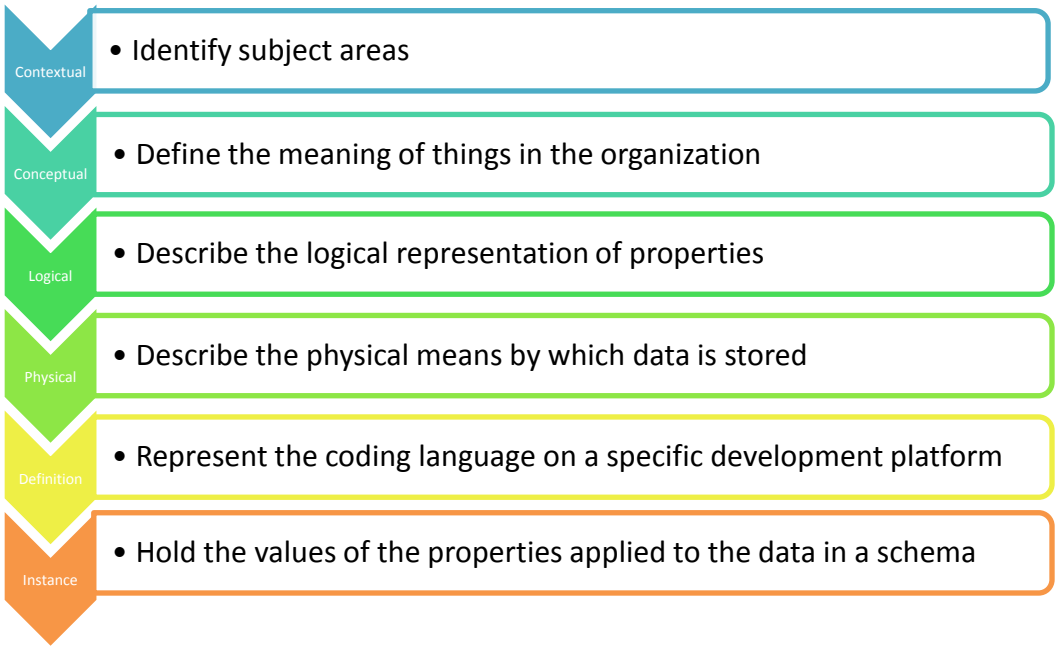
A Few Relevant OMG Standards



Grounding in MOF/UML facilitates

- *Model interoperability*
- *Reuse of common vocabulary, logical models across modeling approaches & asset types*

Modeling Strategies in a Broader Context



Ontology / ODM/OWL, SBVR Vocabulary …

ODM/OWL, E-R, BPMN, UML, SysML, SoaML, AMP…

RDBMS/IMM, ACT, XSD …

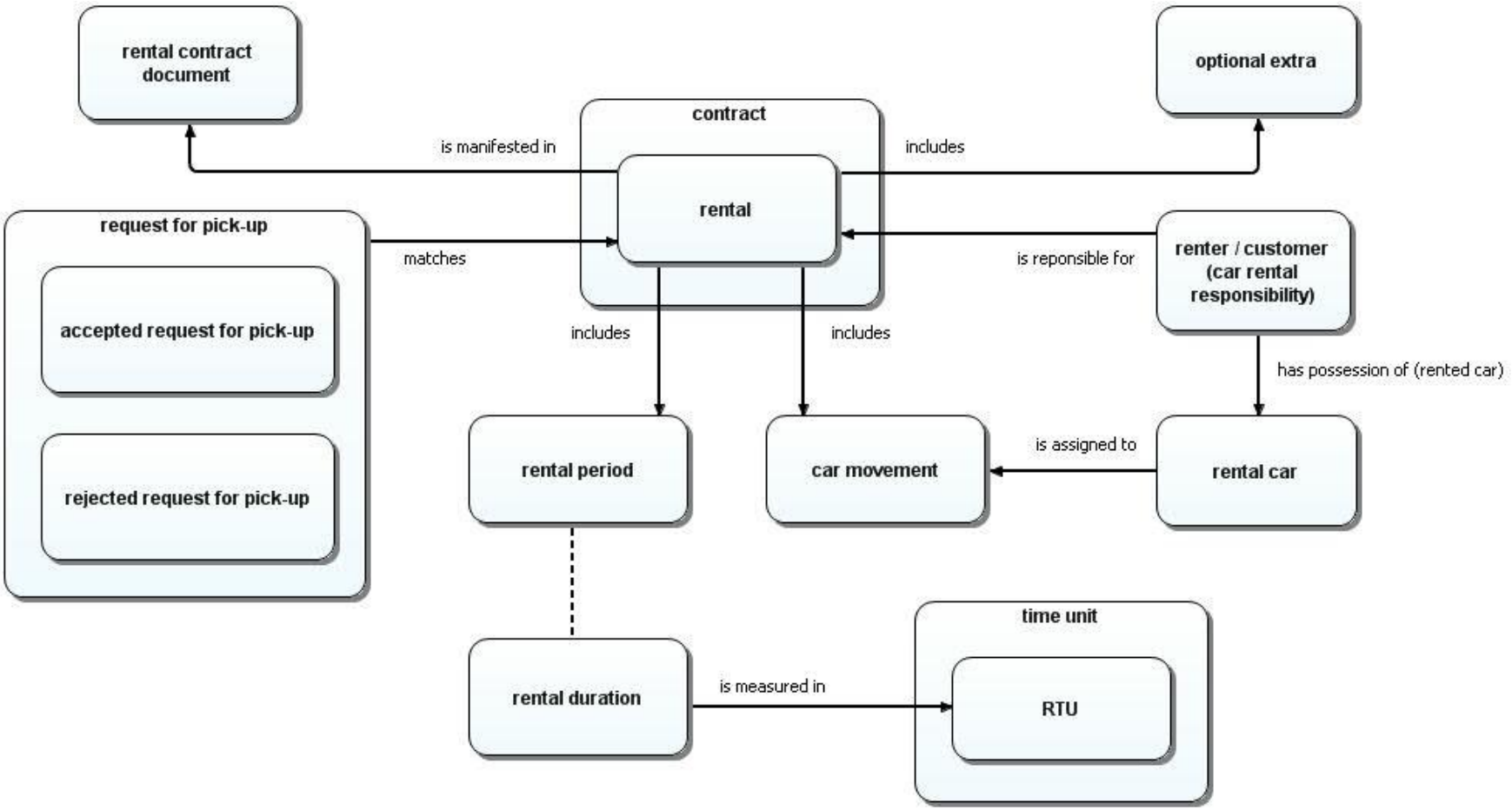
UML, source, scripting languages, stored procedures…

Physical KBs, databases, asset repositories…

**Layering diagram courtesy Kenn Hussey*

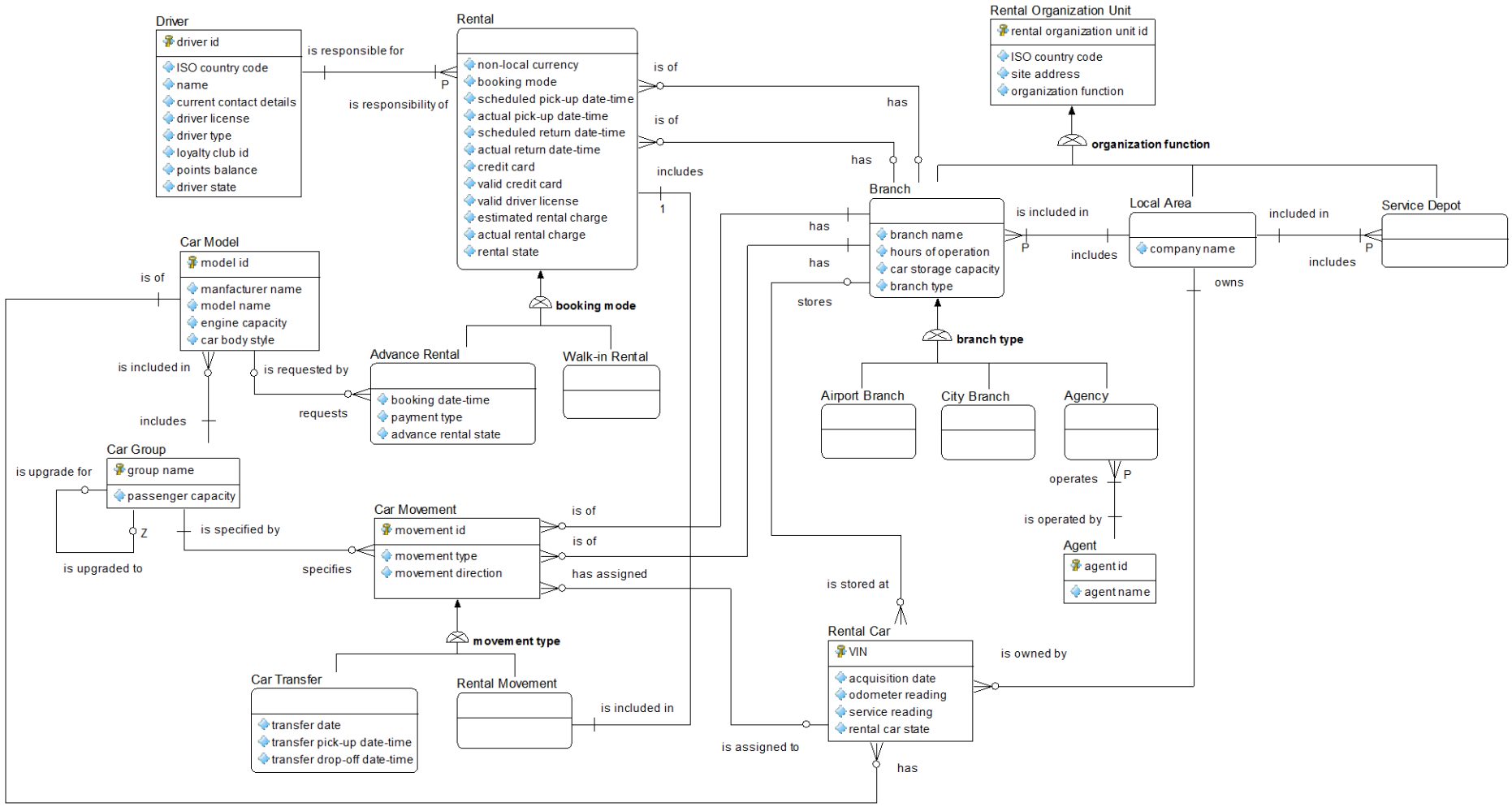
- ***Knowledge Representation / Management for Large Scale Applications***
  - ***Provide broad metadata, process, service & asset management facilities (including feedback/lessons learned…)***
  - ***Enable rich cross-domain, cross-process, cross organizational modeling supported by mapping & transformation services to provide maximum flexibility, interoperability***
  - ***Leverage standards and best practices in information architecture, metadata modeling, management, registration, and governance, and asset management & registration***
  - ***Provide incremental reasoning capabilities for model validation, transformation services***
- ***Repeatable, reusable, interoperable***

Conceptual Model of EU-Rent



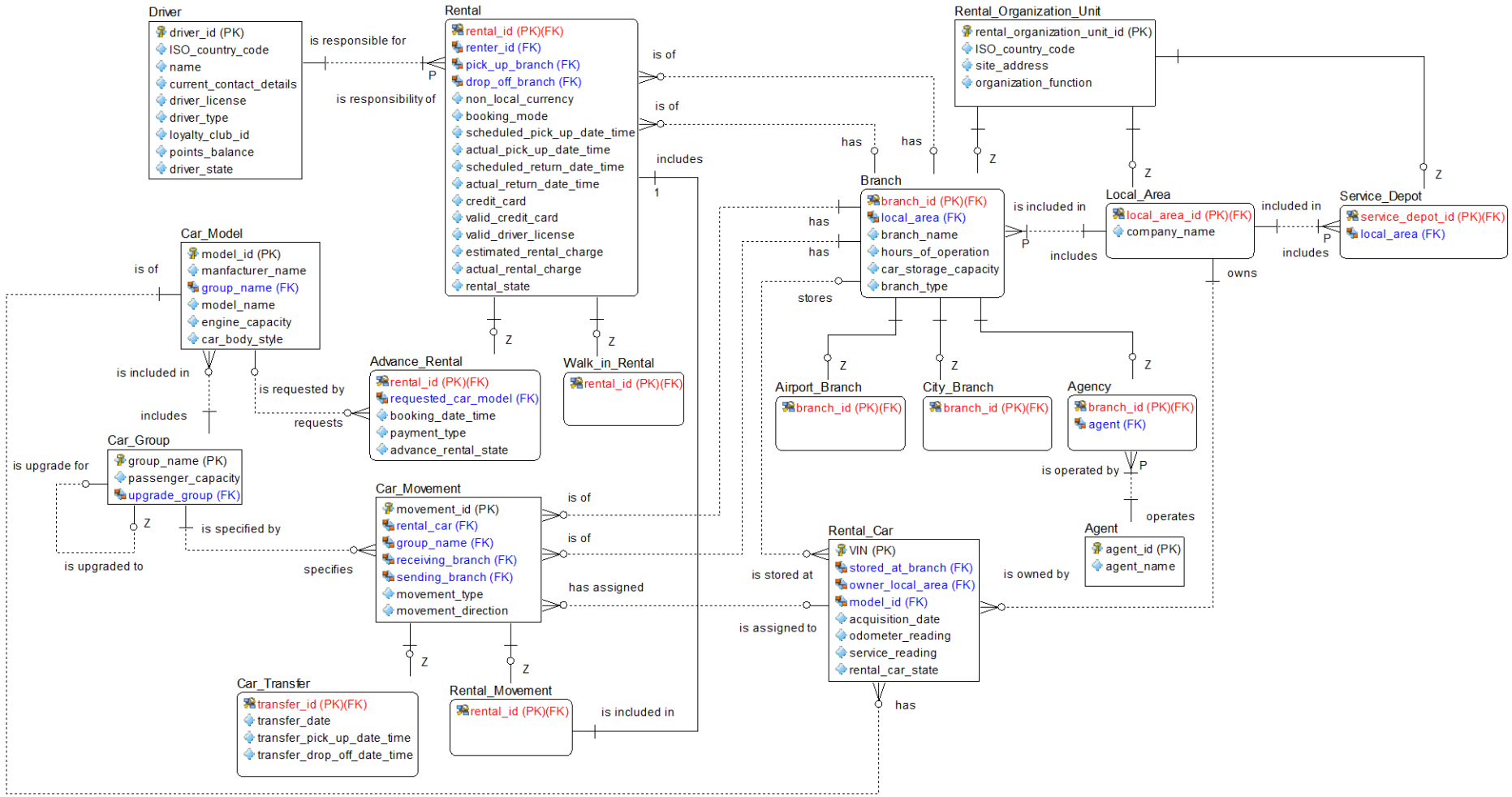
Produced using Embarcadero EA/Studio Business Modeler Edition, courtesy Kenn Hussey

Logical Model of EU-Rent



Produced using Embarcadero ER/Studio, courtesy Kenn Hussey

Physical Model of EU-Rent

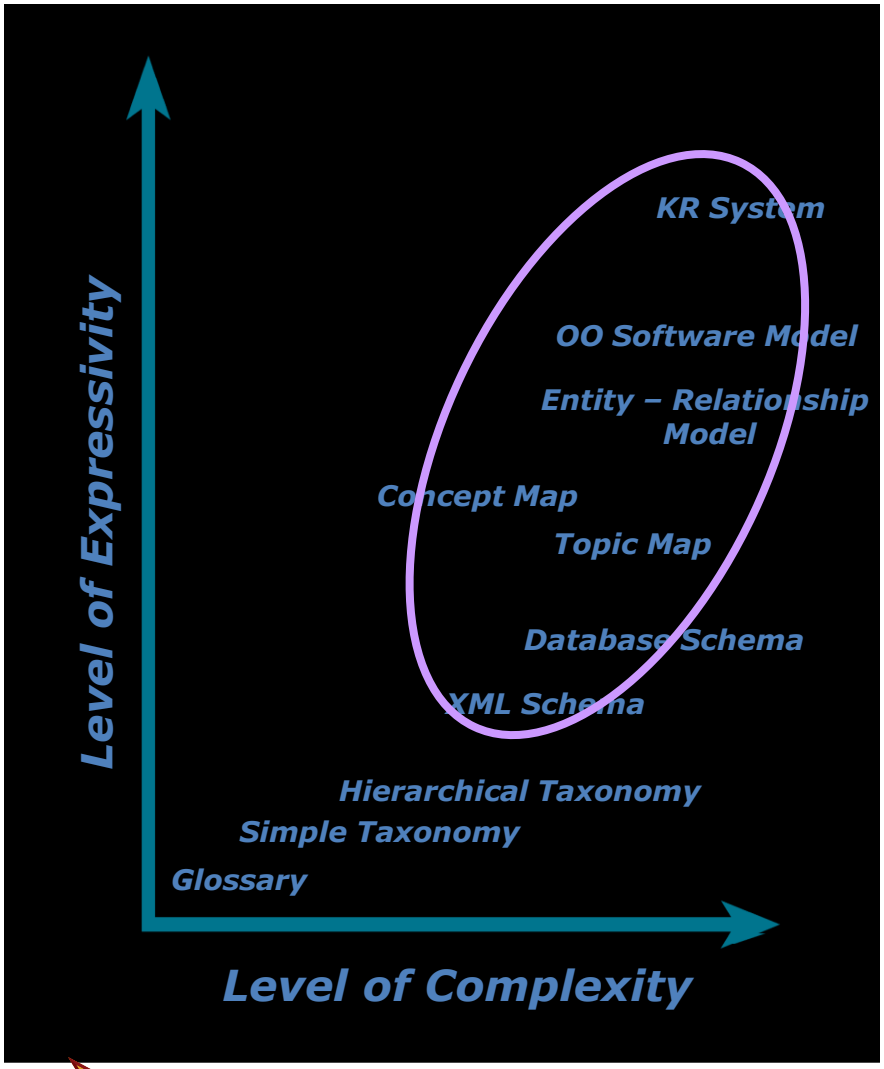


Produced using Embarcadero ER/Studio, courtesy Kenn Hussey

Classifying Ontologies

Classification techniques are as diverse as conceptual models; and generally include understanding

- Level of Expressivity
- Level of Complexity / Structure
- Granularity
- Target Usage, Relevance
- Amount of Automation, Reasoning Requirements
- Prescriptive vs. Descriptive / Reliability / Level of Authoritativeness
- Design Methodology
- Governance
- Vocabulary Management, Metrics



Framework of Dimensions

From NIST's Ontology Summit 2007

- Semantic Dimensions
  - Expressiveness: represents how well *a KR language* addresses increasingly complex semantics
  - Structure: represents how well *an ontology* encodes semantics, with the same or less expressivity than the KR language
  - Granularity: represents the level of detail specified in an ontology
- Pragmatic Dimensions
  - Intended use: the original use case(es), or purpose for developing a particular ontology
  - Automated reasoning: the extent to which the ontology is designed to be used for automated reasoning
  - Prescriptive vs. Descriptive: the extent to which an ontology was intended to be used for descriptive purposes vs. normative prescriptive use (i.e., with high degree of concern for correctness)

Considerations

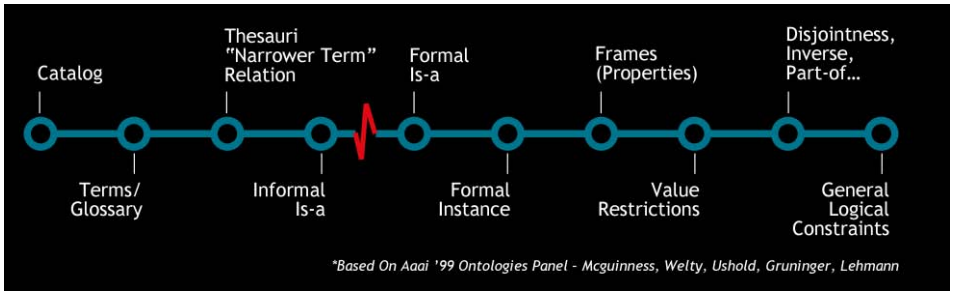
- Intended use of ontologies, including domain requirements (*e.g.*, scientific and engineering apps require formulas, units of measure, computations that may be challenging to represent)
- Intended use of systems that implement or use them, including reasoning requirements, questions to be answered
- What kinds of transformations are required among processes, resources, services to support semantic mediation
- Ontology and system alignment / de-confliction / ambiguity resolution requirements
- Ontology and system composition requirements, dynamic vs. static composition, in what environment and under what constraints
- Performance, sizing, timing requirements of target
- For distributed environments, the number and kinds of resources, processes, services requiring ontologies – how distributed, how unique, developed collaboratively or independently, dynamic community participation or static

A Little Methodology …

- Requirements, domain & use case analysis are critical
  - Develop initial source/reference material
  - Focus on system or application requirements
  - Iterative development starting with a “thread” that covers basic capabilities can ground the work and prioritize decisions
- Need to understand and communicate
  - Architectural trade-offs, cost & technical benefits
  - The nature of the information & kinds of questions that need to be answered drive the architecture, approach, and ontology scoping and design
- Reuse standards, available ontologies whenever possible

What to look for

- A controlled vocabulary
- A hierarchical or taxonomic structure (for query expansion)
  Hardware, Flight Hardware, Fight Avionics, Star Tracker, …
- Knowledge supporting structured queries
  Find all requirements specifying functions performed by a given component
- Efficient inference (*i.e.*, limited expressive power) vs. increased expressivity (potentially expensive or resource bounded computation)
- Custom reasoning for temporal relations, geospatial, dynamic evaluation of engineering equations, process-specific, conditional operations
- Computational tractability



Start with canonical definitions

- General concepts as well as domain-specific knowledge
- Basic starting point – cross-domain definitions
  - Namespace definitions, related metadata, registration, governance policies
  - Commonly used structures & vocabularies: messaging, event processing, service descriptions, general systems & software engineering terminology
  - Common metadata for asset/artifact management (*e.g.*, documents, images & multimedia, engineering artifacts)
- Domain vocabularies must be prioritized, selected based on business requirements, clear ROI
- Common early targets include domain taxonomies supporting
  - service registration
  - asset/artifact repository search & retrieval
  - partial service & related metadata generation
  - automated verification

IDEF5 (Integrated Definition Methods) Ontology Capture Method Analysis

- Layout a high-level architecture for ontology components
- Identify the relationships among components – roles, domain, interface, process, utility
- For each ontology component
  - Describe its domain and scope, how it will be used
  - Identify example questions and anticipated/sample answers for the application(s) it will support
  - Identify key stakeholders, ownership, maintenance, resources for instance knowledge
  - Describe anticipated reuse/evolution path
  - Identify critical standards, resources that it must interoperate with, dependencies
- Resources
  - http://www.idef.com/IDEF5/html
  - http://www.kbsi.com/technology/methods/sbont.htm

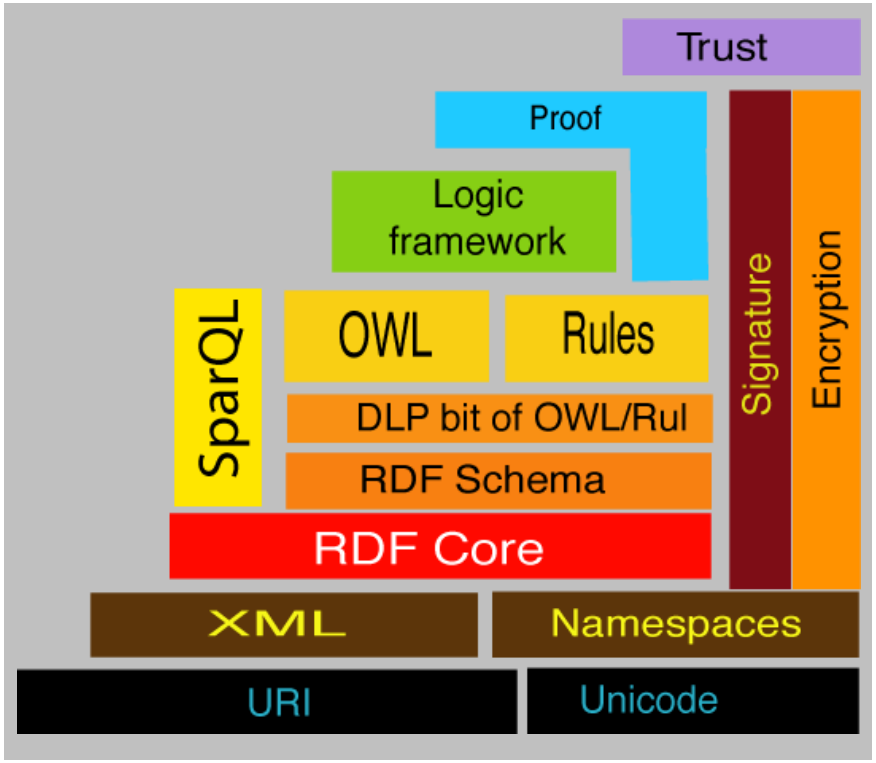
Part 2: Ontology Development in UML: Web Ontology Language (OWL) & Ontology Definition Metamodel

- Intro to RDF & OWL
- Motivation for using UML
- Ontology development using the Ontology Definition Metamodel (ODM)
- Relationships to other OMG & ISO Standards
- Planned work at OMG

Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

-- Tim Berners-Lee



Semantic Web stack from "Putting the Web back into Semantic Web", Tim Berners-Lee, ISWC2005 Keynote

Guiding Principles

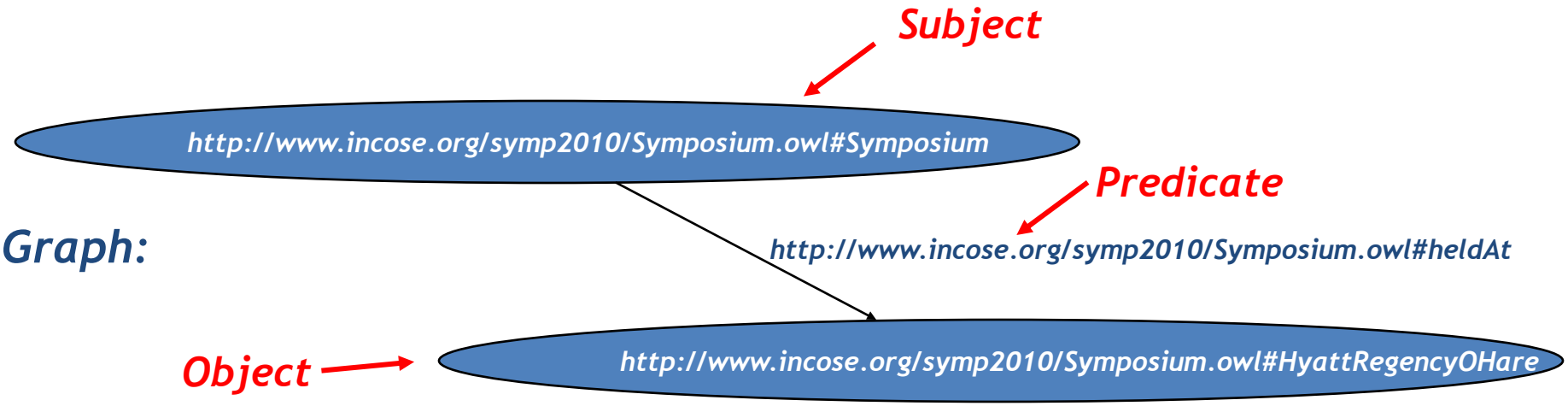
- Historically, knowledge representation and reasoning systems have operated under *closed world* assumptions
- Uncertainty is magnified under *open-world*, "wild, wild web" conditions, making reasoning much more difficult
- Semantic web languages are designed to support less certainty, to provide "better" search results, informed answers to questions, not absolutes
- Because they are based on XML, such languages can assist businesses in leveraging existing investment in mark-up, content, and data
  - To augment business intelligence/analysis and knowledge mining
  - To support knowledge sharing and collaboration, augment enterprise information integration
  - Enrich web services and other applications
  - Support policy-based applications and ensure compliance with policy

  at a lower cost with higher potential ROI than traditional computing methods

Resource Description Framework (RDF)

- Describes relationships
- Uses URIs used for naming
- Language has
  - graph based model
  - RDF/XML serialization (exchange syntax)
  - other presentation syntaxes (N3, Turtle, …)
- Specification, W3C presentations, tools are available at
  - Semantic Web: http://www.w3.org/standards/semanticweb/
  - Linked Data: http://www.w3.org/standards/semanticweb/data
  - RDF: http://www.w3.org/standards/techs/rdf#w3c_all

RDF Notation Options



XML/RDF:

```
<rdf:Description rdf:ID="Symposium"/>
  <symp2010:heldAt rdf:resource="#HyattRegencyOHare"/>
</rdf:Description>
```

N-triples: `symp2010:Symposium symp2010:heldAt symp2010:HyattRegencyOHare .`

Courtesy Evan Wallace, NIST

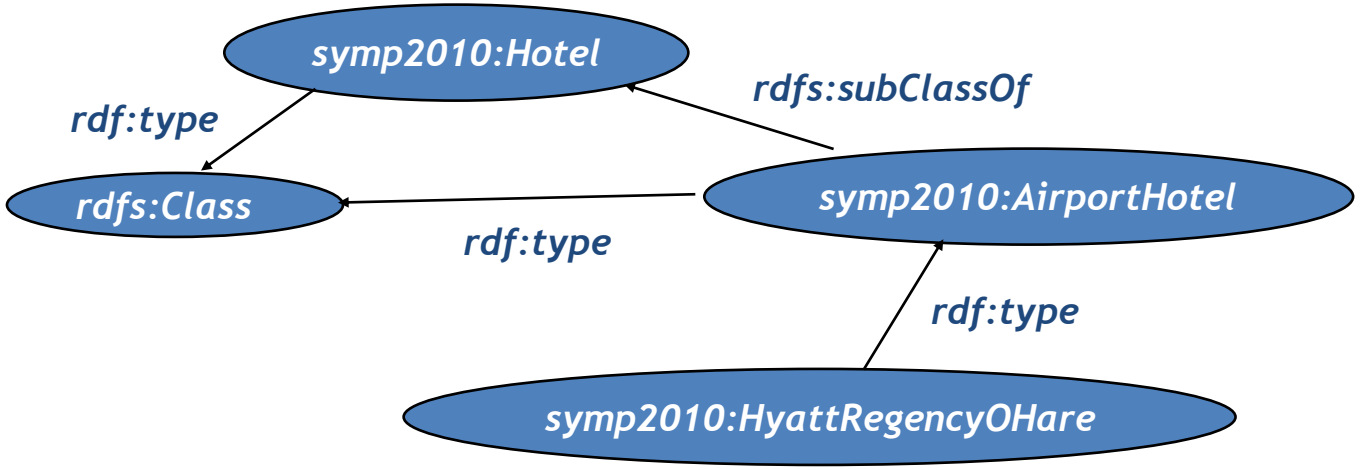
RDF Schema (RDFS)

- An RDF vocabulary that provides for identifying:
  - classes,
  - subsumption (inheritance) relations for classes,
  - subsumption (inheritance) relations for properties,
  - domain and range for properties

Courtesy Evan Wallace, NIST

RDF Schema (RDFS)

Graph:



XML/RDF:

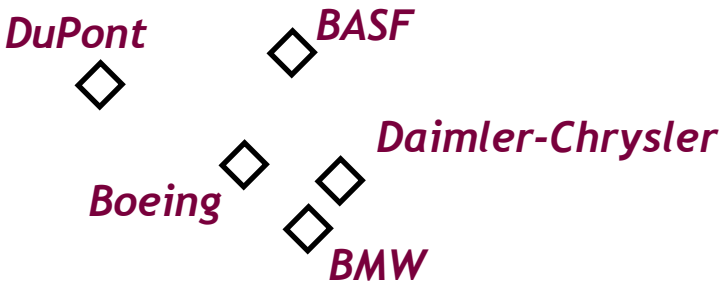
```
<rdf:Description rdf:ID="Hotel">
   <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 </rdf:Description>

<rdfs:Class rdf:ID="AirportHotel">
   <rdfs:subClassOf rdf:resource="#Hotel"/>
 </rdfs:Class>

<symp2010:AirportHotel rdf:ID="HyattRegencyOHare"/>
```

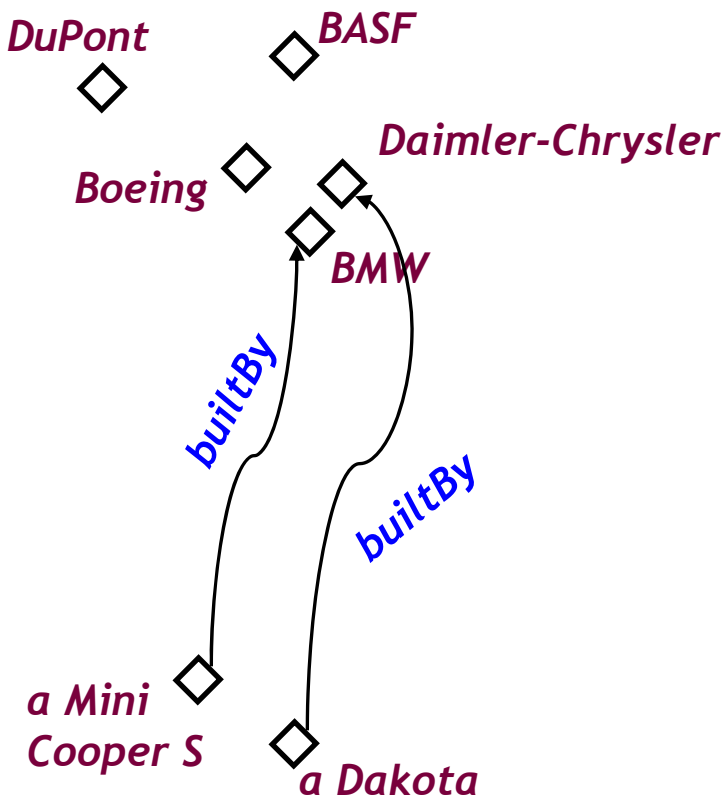
Courtesy Evan Wallace, NIST

OWL Individuals



Courtesy Evan Wallace, NIST

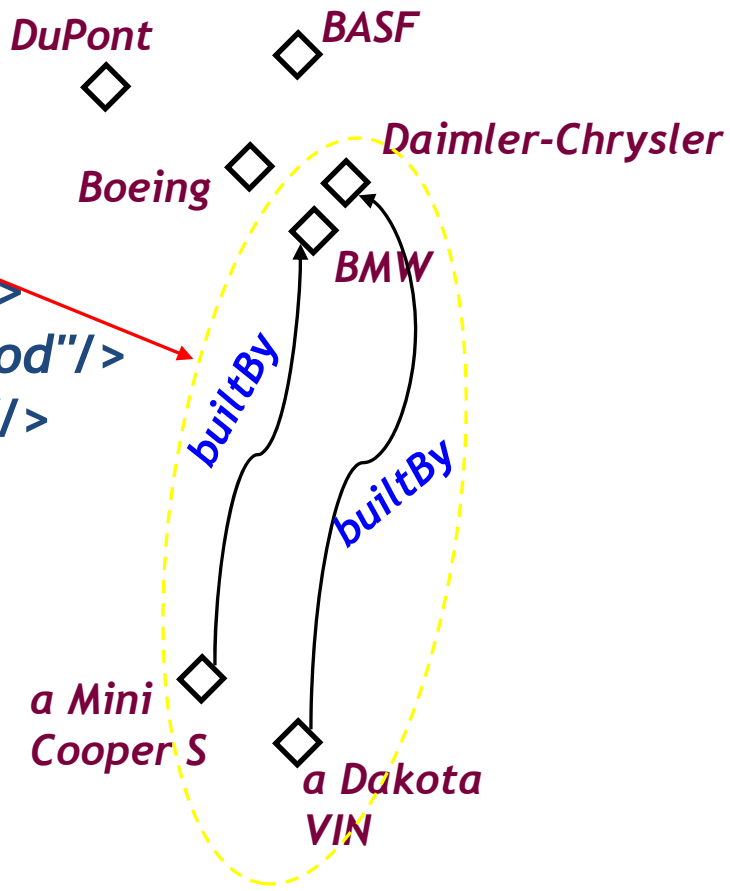
OWL Statements



Courtesy Evan Wallace, NIST

OWL ObjectProperty

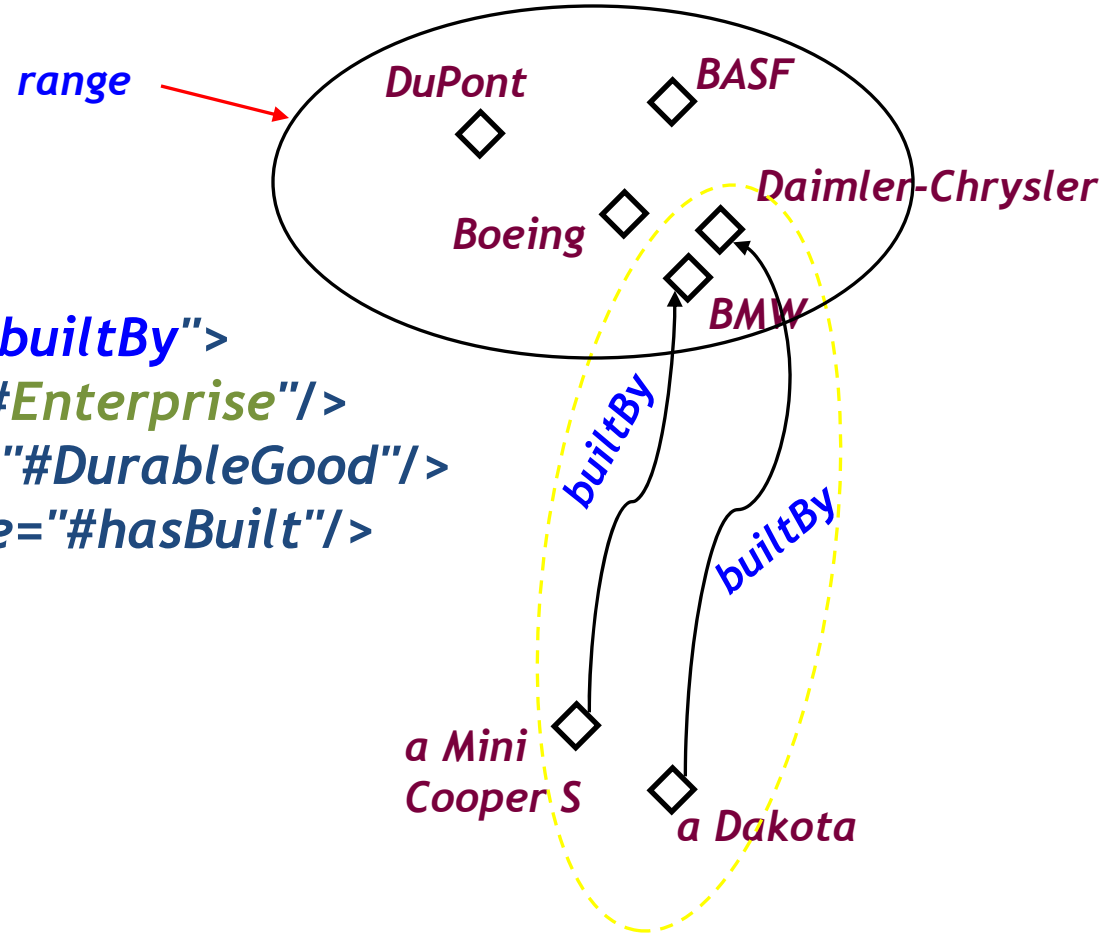
```
<owl:ObjectProperty rdf:ID="builtBy">
  <rdfs:range rdf:resource="#Enterprise"/>
  <rdfs:domain rdf:resource="#DurableGood"/>
  <owl:inverseOf rdf:resource="#hasBuilt"/>
</owl:ObjectProperty>
```



Courtesy Evan Wallace, NIST

OWL ObjectProperty

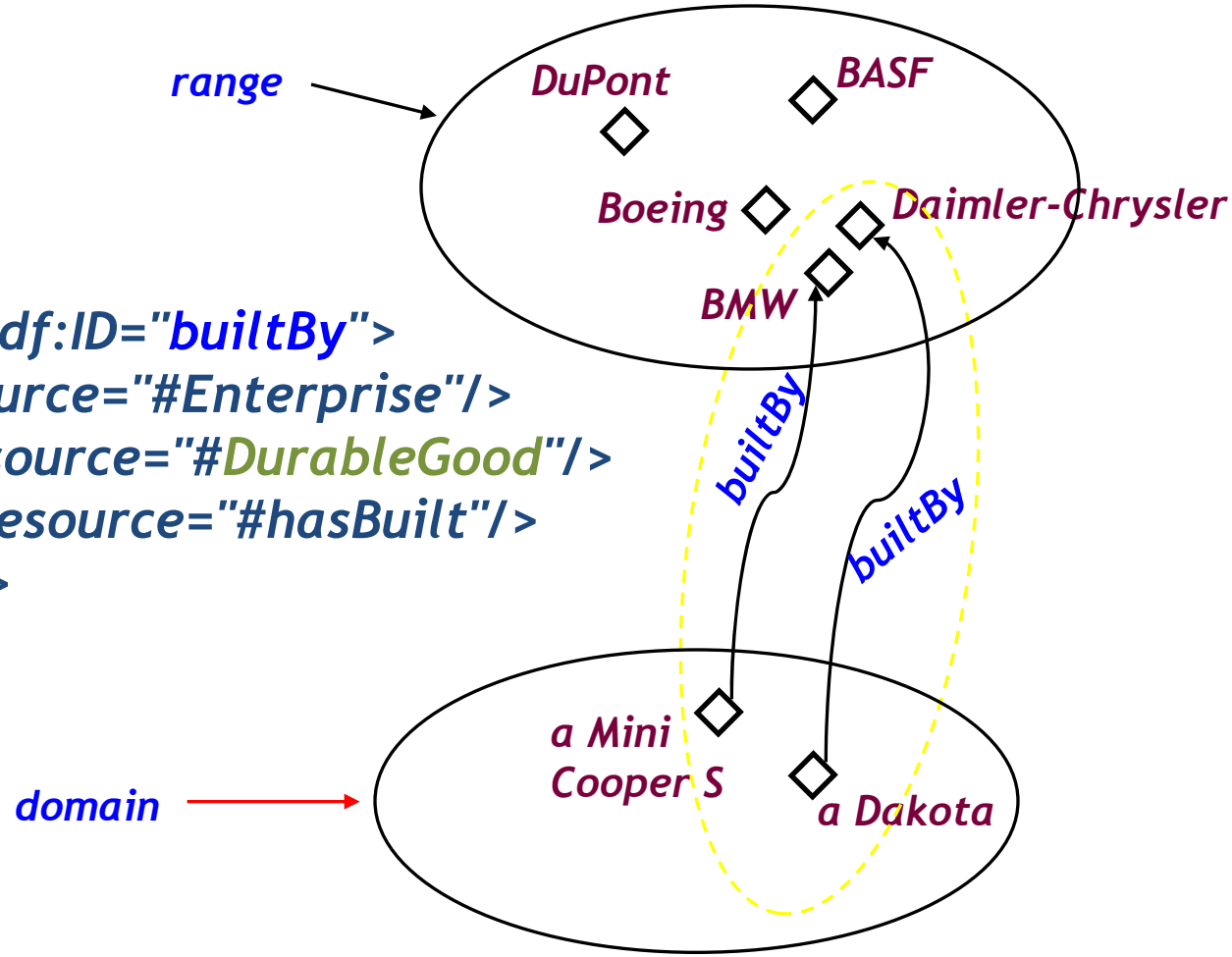
```
<owl:ObjectProperty rdf:ID="builtBy">
  <rdfs:range rdf:resource="#Enterprise"/>
  <rdfs:domain rdf:resource="#DurableGood"/>
  <owl:inverseOf rdf:resource="#hasBuilt"/>
</owl:ObjectProperty>
```



Courtesy Evan Wallace, NIST

OWL ObjectProperty

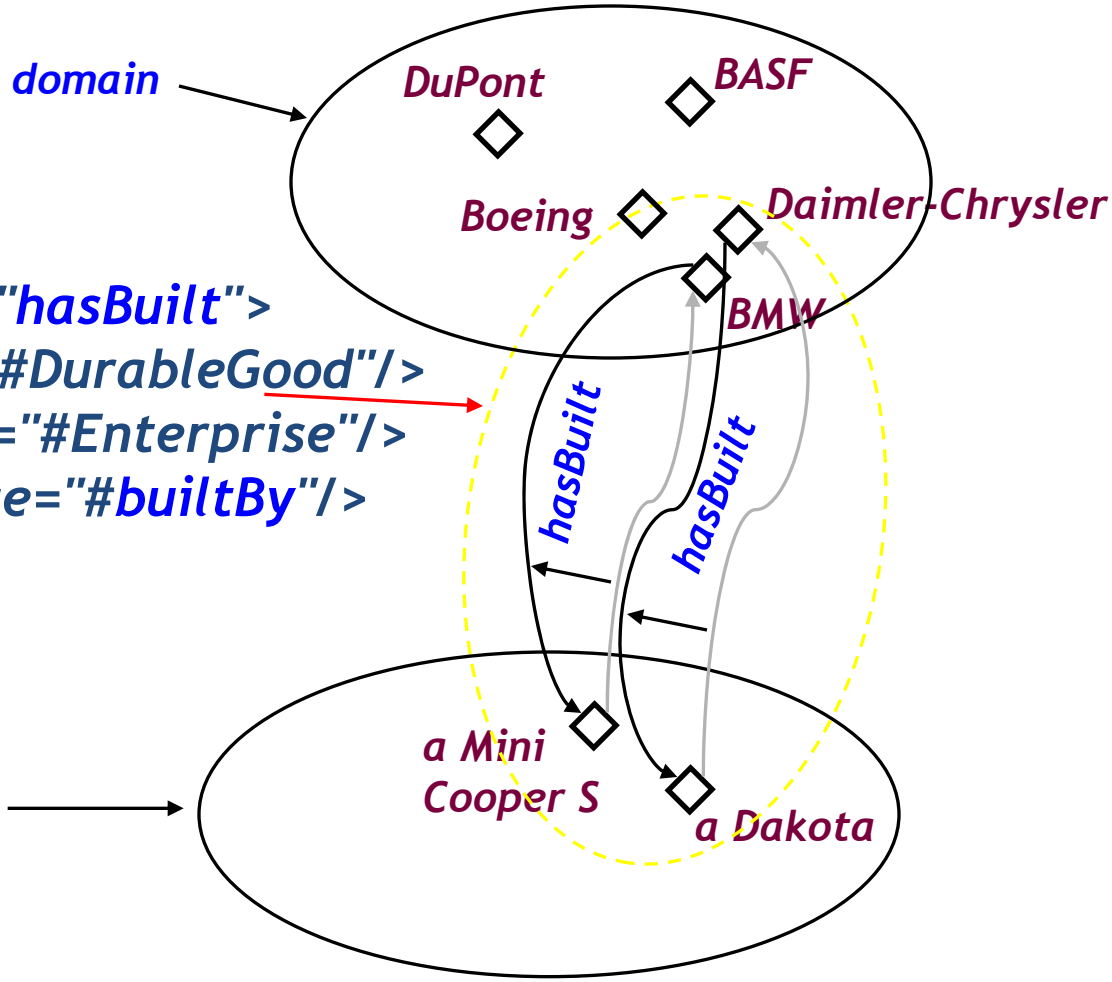
```
<owl:ObjectProperty rdf:ID="builtBy">
   <rdfs:range rdf:resource="#Enterprise"/>
   <rdfs:domain rdf:resource="#DurableGood"/>
   <owl:inverseOf rdf:resource="#hasBuilt"/>
</owl:ObjectProperty>
```



Courtesy Evan Wallace, NIST

Inverse Properties

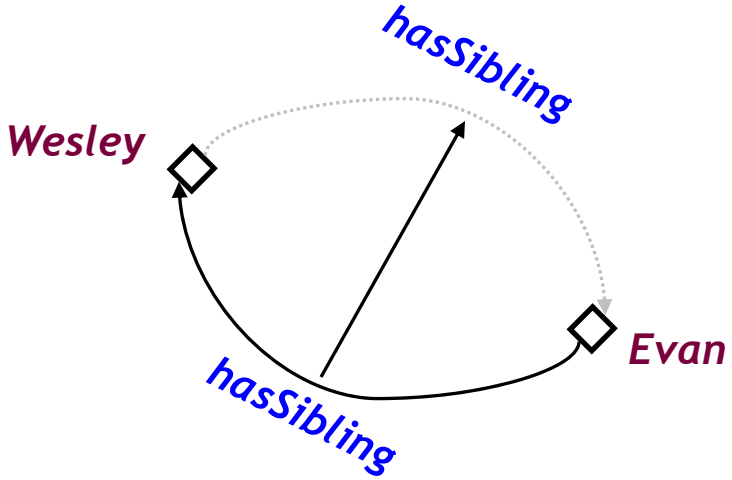
```
<owl:ObjectProperty rdf:ID="hasBuilt">
  <rdfs:range rdf:resource="#DurableGood"/>
  <rdfs:domain rdf:resource="#Enterprise"/>
  <owl:inverseOf rdf:resource="#builtBy"/>
</owl:ObjectProperty>
```



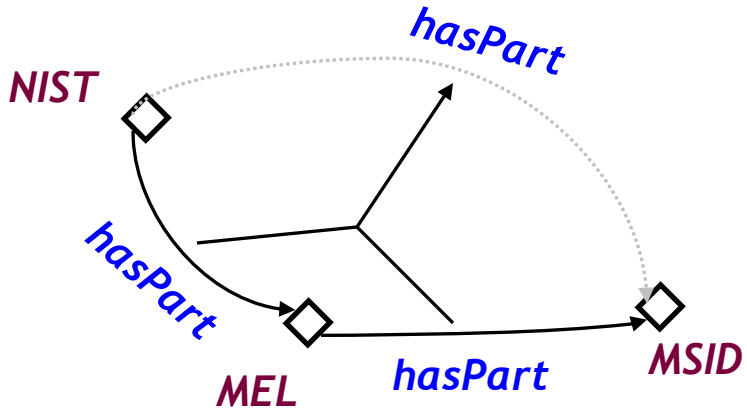
Courtesy Evan Wallace, NIST

Meta-Properties – Logical

- Symmetric



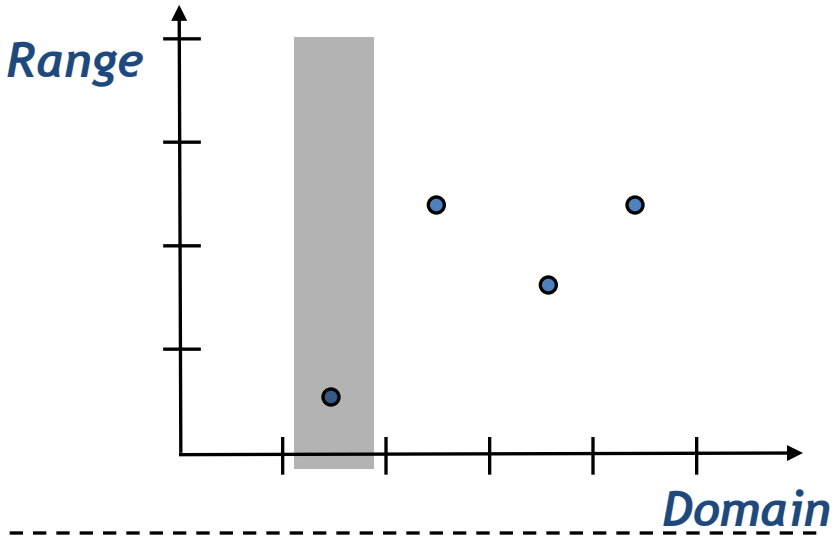
- Transitive



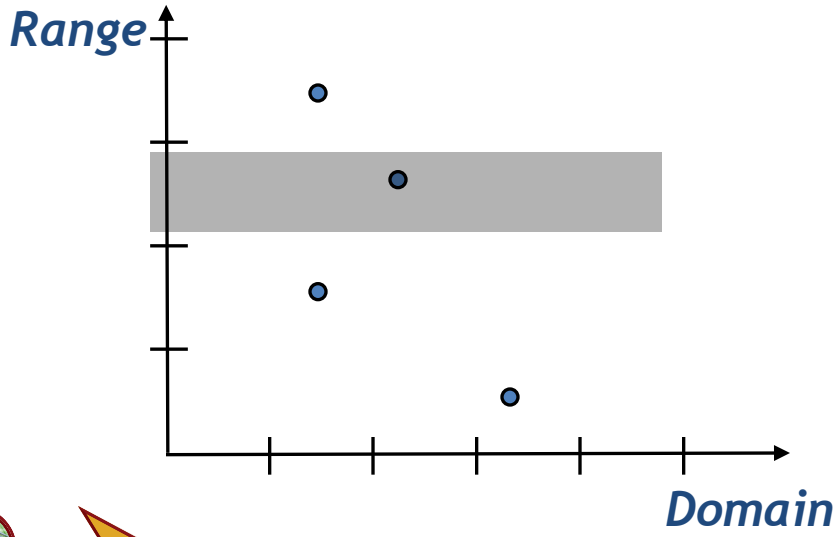
Courtesy Evan Wallace, NIST

Meta-Properties – Global Cardinality Restriction

- Functional

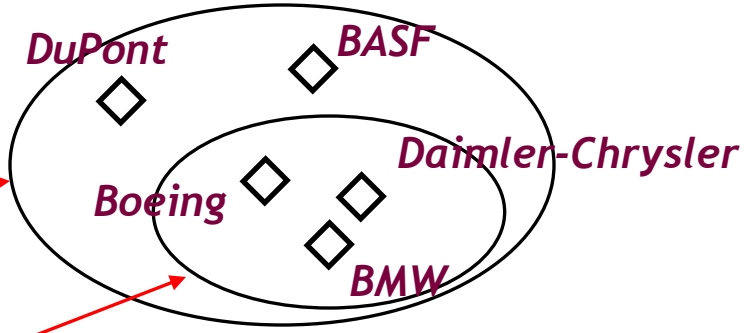


- Inverse Functional



Courtesy Evan Wallace, NIST

OWL Class




```
<owl:Class
rdf:ID="ManufacturingEnterprise"/>
```

```
<owl:Class rdf:ID="DiscreteManufacturingEnterprise">
 <rdfs:subClassOf>
     <owl:Class rdf:about="#ManufacturingEnterprise"/>
 </rdfs:subClassOf>
</owl:Class>
```

Courtesy Evan Wallace, NIST

Class Descriptions

6 Types

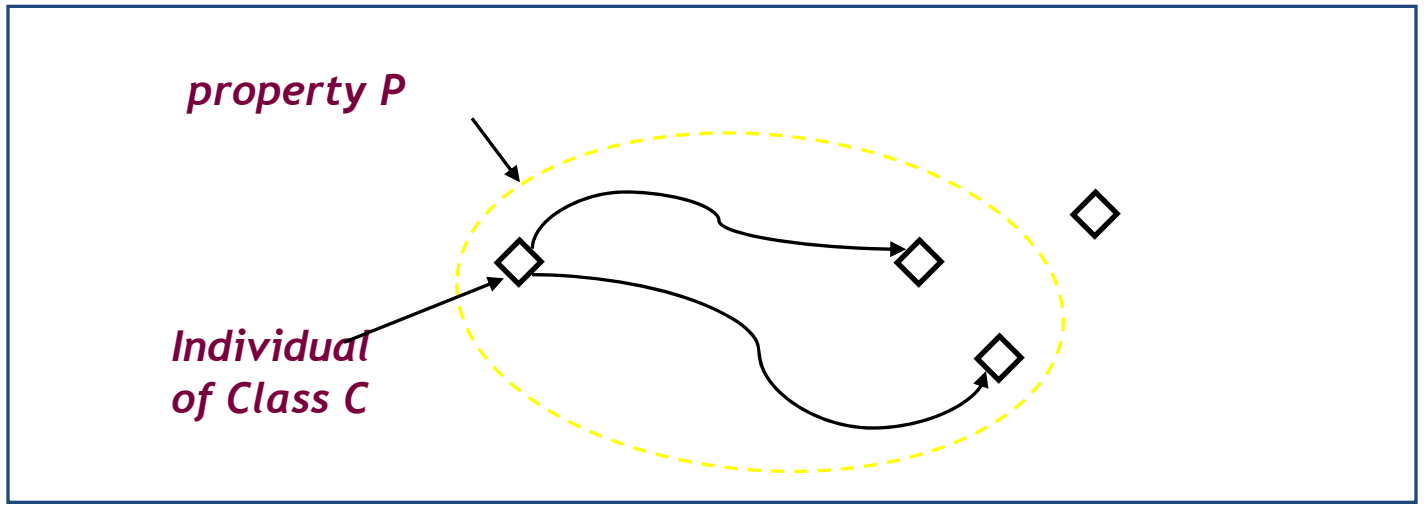
1. class identifier
2. enumeration
3. property restriction
4. intersection
5. union
6. complement

anonymous

nested

Courtesy Evan Wallace, NIST

Class Descriptions – Property Restriction



- Quantified property restriction (type)
  - Universally quantified – allValuesFrom
  - Existentially quantified - someValuesFrom
- hasValue property restriction (value)
- Property cardinality restriction (# of values)

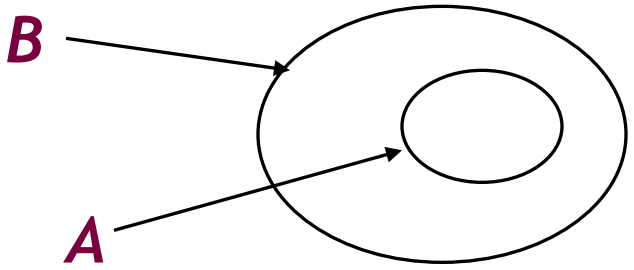
Courtesy Evan Wallace, NIST

Class Axioms

- Subsumption (necessary)
  - A ⊆ B where B

  is a class description

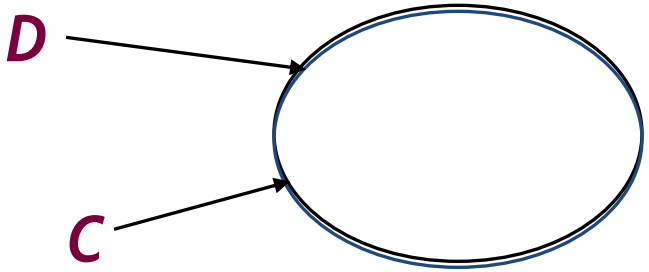
  partial or *primitive class*



- Definition (necessary and sufficient)
  - C ≡ D where D

  is a class description

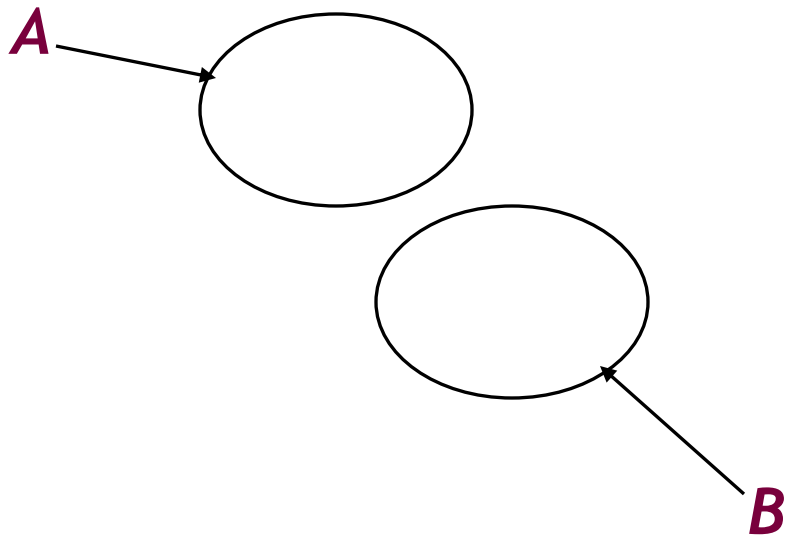
  complete or *defined class*



Courtesy Evan Wallace, NIST

Class Axioms

- Disjointness
  - A disjointWith B



Courtesy Evan Wallace, NIST

Why UML for Ontology Modeling?

- A little review:

  Every semi-trailer truck has at least 3 axles.

  $$(\forall x)(((SemiTrailerTruck(x) \wedge (\exists y)(SemiTrailer(y) \wedge (hasPart\ (x,y))) \wedge (SemiTrailerTruck(x) \wedge (\exists z)(TractorUnit(z) \wedge (hasPart\ (x,z)))) \supset (\exists s)(set(s) \wedge (count(s,(\geq 3)) \wedge (\forall w)(member(w,s) \supset (Axle(w) \wedge hasPart(x,w))))).$$

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

sandpiper software

XML angle brackets can be equally difficult to read

File Edit View Insert Format Help

```
<owl:Class rdf:ID="SemiTrailerTruck">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >

</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasAxle"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Axle"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasPart"/>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="#TractorUnit"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#SemiTrailer"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasPart"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >SemiTrailerTruck</rdfs:label>
  <rdfs:subClassOf rdf:resource="#ArticulatedVehicle"/>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Truck"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasAxle"/>
      </owl:onProperty>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#no
      >3</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Altova XMLSpy - [ssi-ANSC-test.owl]

File Edit Project XML DTD/Schema Schema design XSL/XQuery Authentic Convert View Browser Tools Window Help

```
211   </owl:Class>
212   <owl:Class rdf:ID="SemiTrailerTruck">
213    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
214    >
215
216   </rdfs:comment>
217    <rdfs:subClassOf>
218     <owl:Restriction>
219      <owl:onProperty>
220       <owl:ObjectProperty rdf:about="#hasAxle"/>
221      </owl:onProperty>
222      <owl:allValuesFrom rdf:resource="#Axle"/>
223     </owl:Restriction>
224    </rdfs:subClassOf>
225    <rdfs:subClassOf>
226     <owl:Restriction>
227      <owl:onProperty>
228       <owl:ObjectProperty rdf:about="#hasPart"/>
229      </owl:onProperty>
230      <owl:someValuesFrom rdf:resource="#TractorUnit"/>
231     </owl:Restriction>
232    </rdfs:subClassOf>
233    <rdfs:subClassOf>
234     <owl:Restriction>
235      <owl:someValuesFrom rdf:resource="#SemiTrailer"/>
236      <owl:onProperty>
237       <owl:ObjectProperty rdf:about="#hasPart"/>
238      </owl:onProperty>
239     </owl:Restriction>
240    </rdfs:subClassOf>
241    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
242    >SemiTrailerTruck</rdfs:label>
243    <rdfs:subClassOf rdf:resource="#ArticulatedVehicle"/>
244    <rdfs:subClassOf>
245     <owl:Class rdf:ID="Truck"/>
246    </rdfs:subClassOf>
247    <rdfs:subClassOf>
248     <owl:Restriction>
249      <owl:onProperty>
250       <owl:ObjectProperty rdf:about="#hasAxle"/>
251      </owl:onProperty>
252      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"
253      >3</owl:minCardinality>
254     </owl:Restriction>
255    </rdfs:subClassOf>
256   </owl:Class>
```

Info | Elements | Attributes | Entities

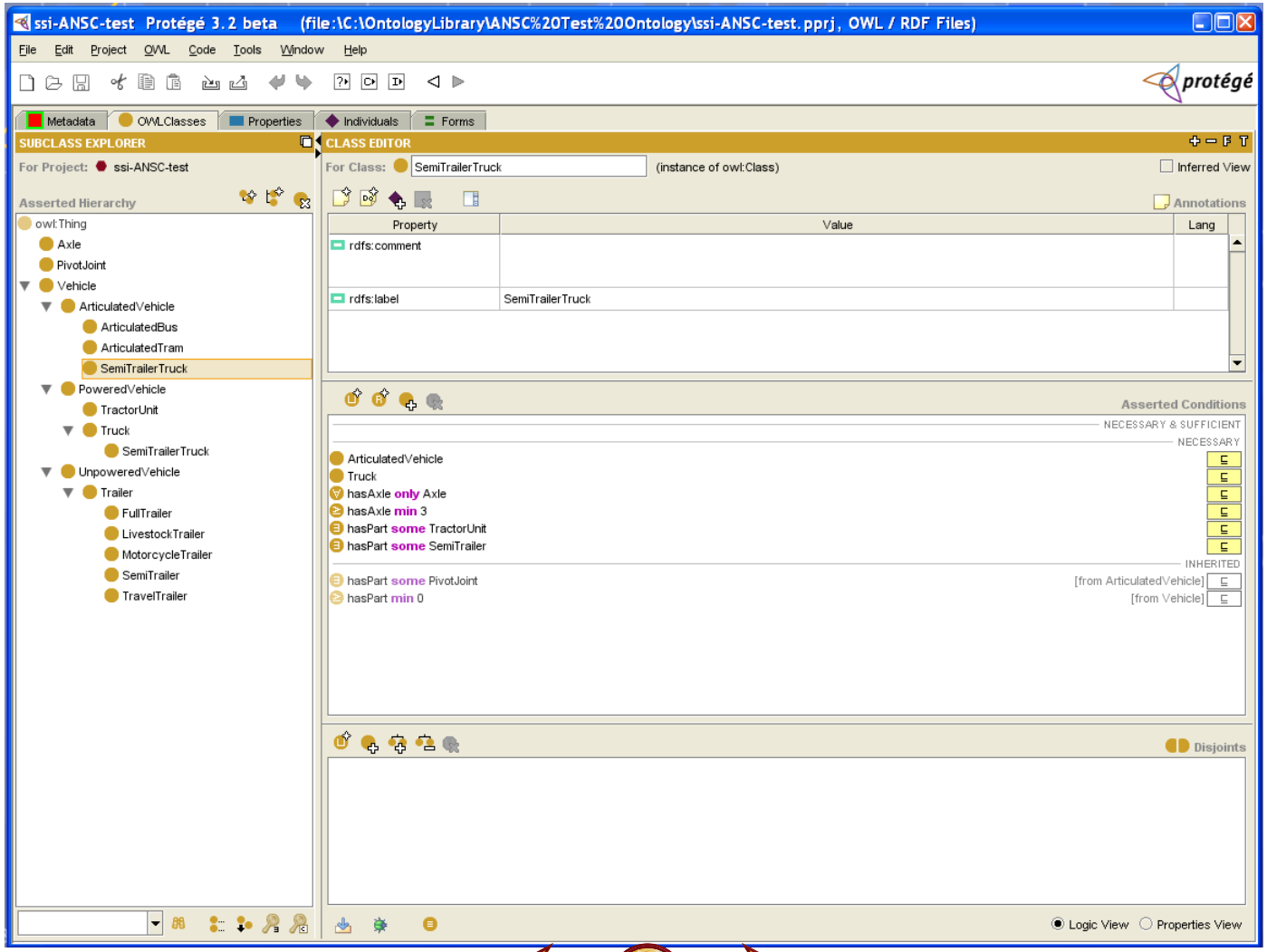
Ent amp &
Ent apos '
Ent gt >
Ent lt <
Ent quot "

Text Grid Schema/WSDL Authentic Browser

ssi-ANSC-test.owl

XMLSpy v2006 U Registered to Elisa F Kendall (Sandpiper Software, Inc.) ©1998-2005 Alto Ln 1, Col 1 CAP NUM SCRL

Editors such as Protégé are better …

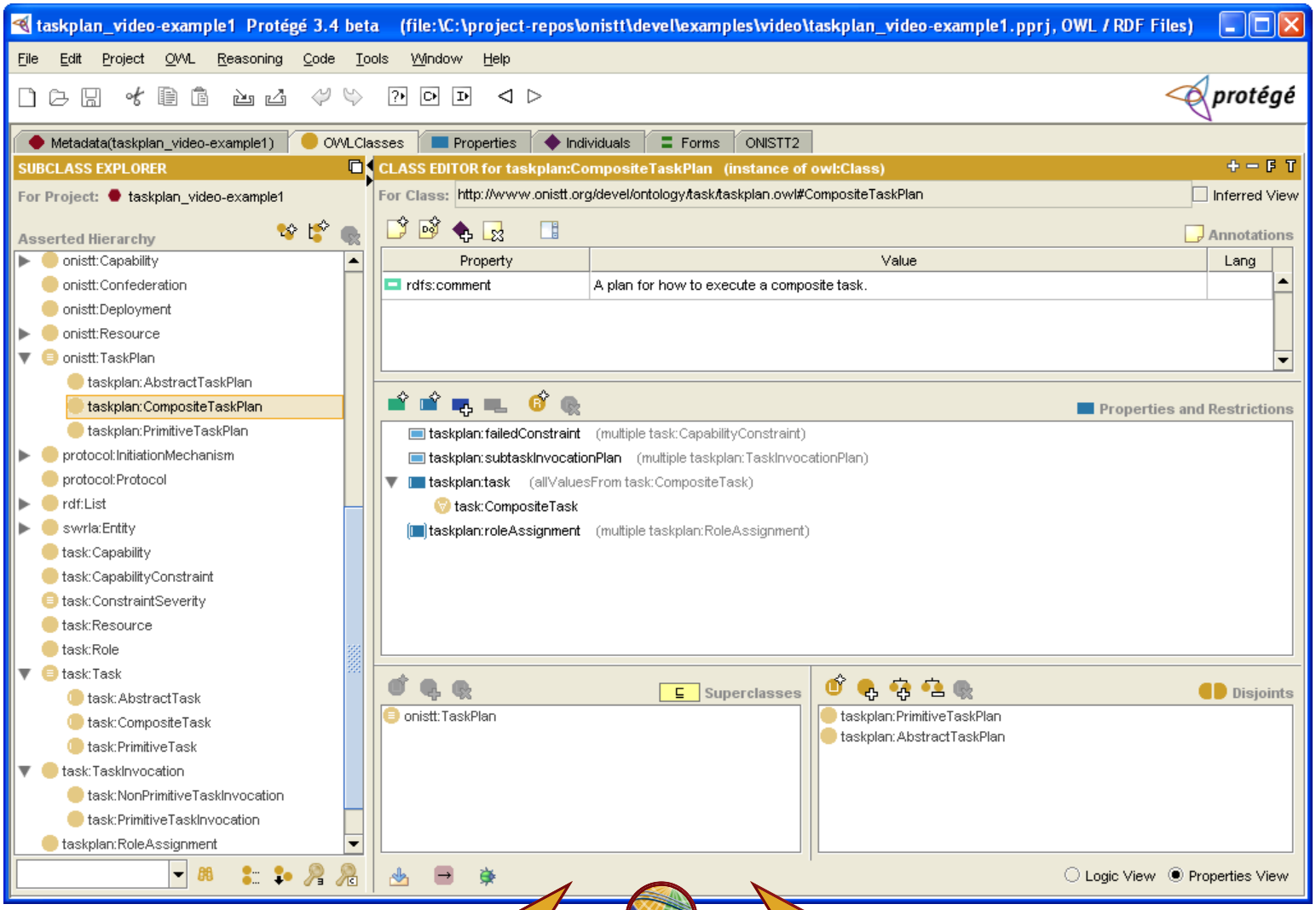


National Aeronautics and Space Administration

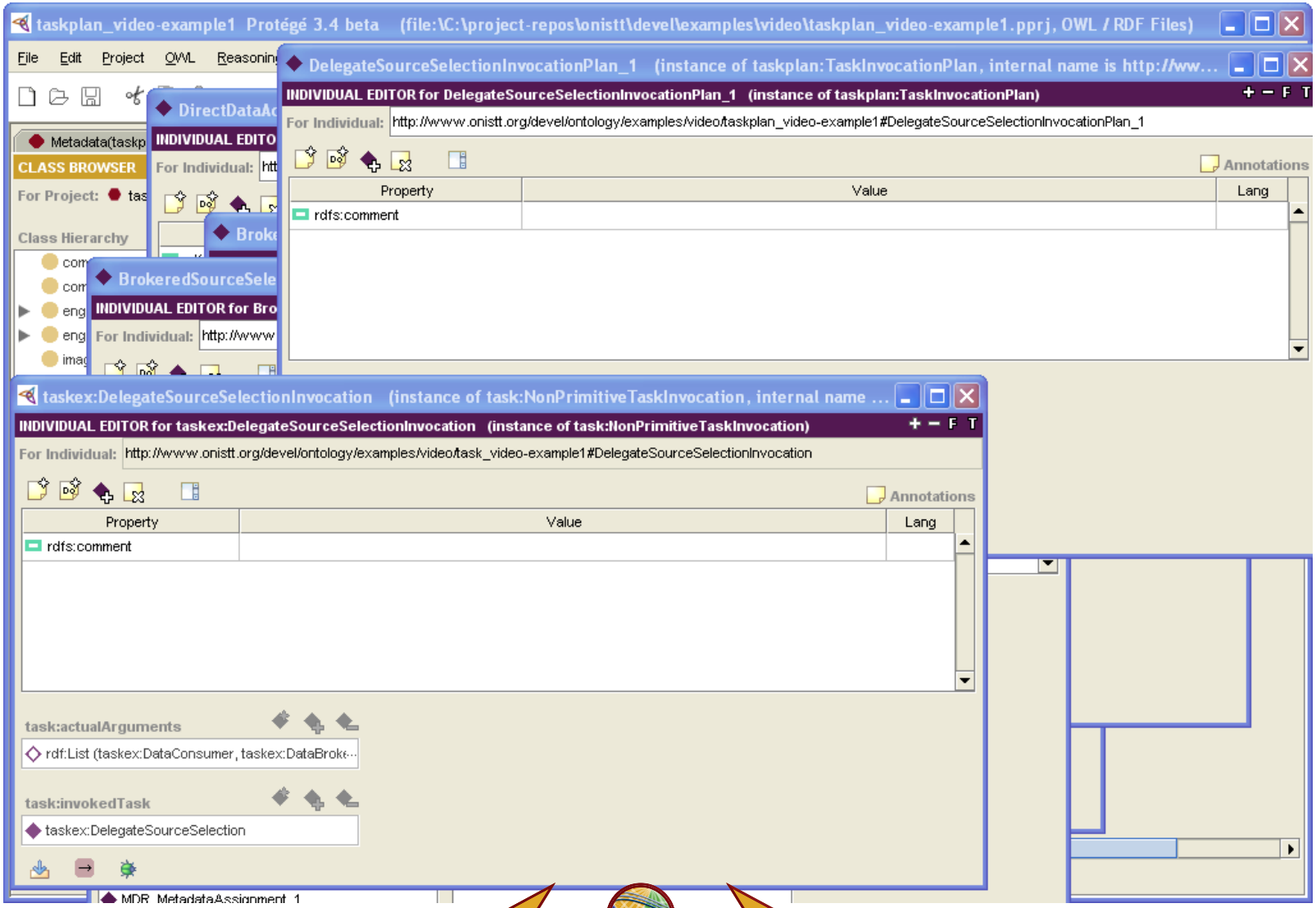
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

sandpiper software

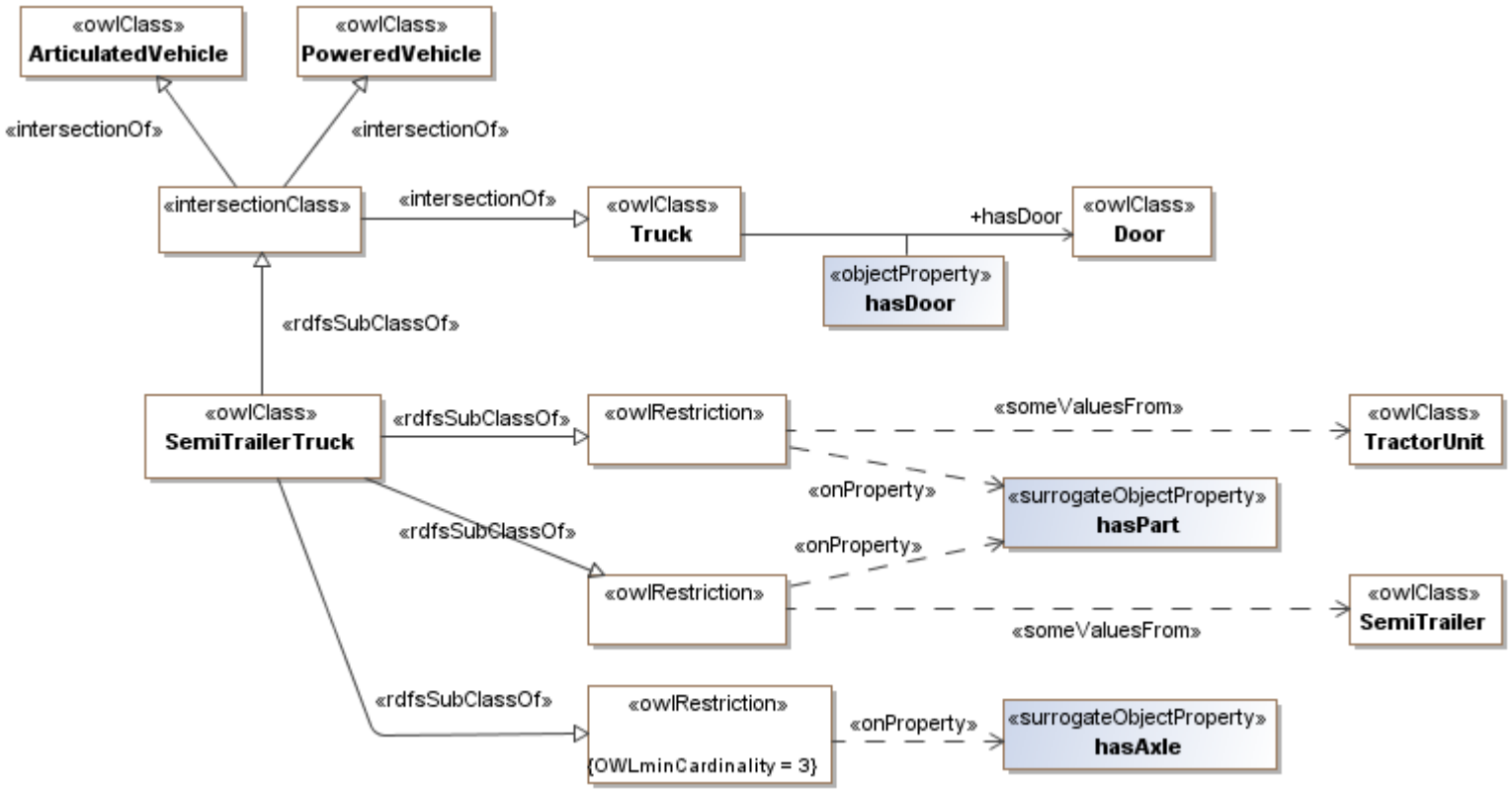
As complexity increases, it can be difficult to follow relationships …



And it's worse with individuals



UML's well-known graphical notation is more accessible to many



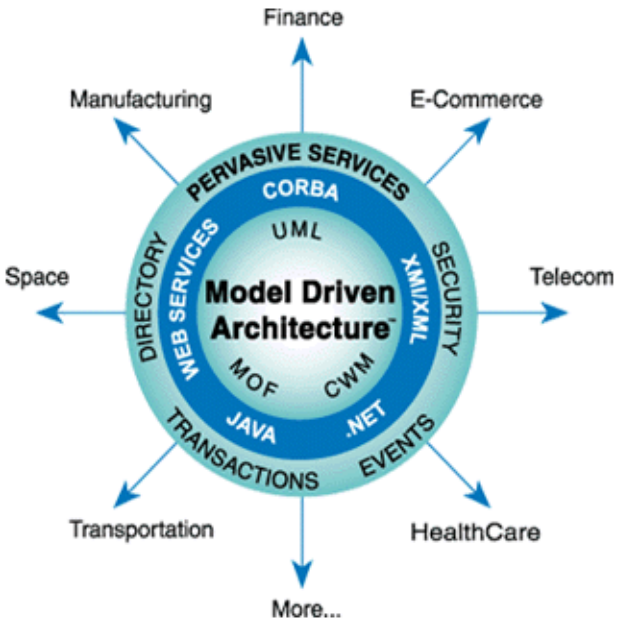
A little more background …

- UML provides a graphical notation, but OWL & UML are very different languages
  - Language mapping from OWL to UML only covers a fraction of the UML language – to logical (class) diagrams
  - Key distinctions:
    - Properties in OWL are first-class citizens, second class in UML (meaning, it's difficult to map OWL properties directly to UML properties or associations)
    - UML supports n-ary relations whereas in OWL, properties are binary
    - OWL uses true set theoretic concepts (intersection, union, complement, etc.), where UML is less formal

This is overly simplified – the mapping is not straightforward, but the benefits of having a graphical notation are acknowledged in the W3C OWL 2 community.

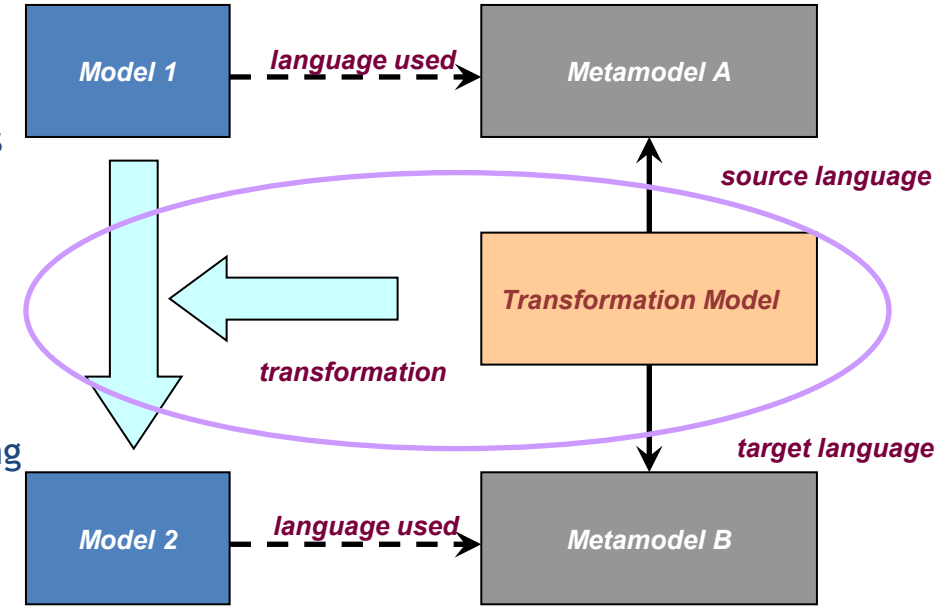
Model Driven Architecture® (MDA®)

- Insulates business applications from technology evolution, for
  - Increased portability and platform independence
  - Cross-platform interoperability
  - Domain-relevant specificity
- Consists of standards and best practices across a range of software engineering disciplines
  - The Unified Modeling Language (UML®)
  - The Meta-Object Facility (MOF™)
  - The Common Warehouse Metamodel (CWM™)
- MOF defines the metadata architecture for MDA
  - Database schema, UML and ER models, business and manufacturing process models, business rules, API definitions, configuration and deployment descriptors
  - Supports automation of physical management and integration of enterprise metadata
  - MOF models of metadata (of the abstract syntax of the representation language) are called *metamodels*



MOF-Based Metadata Management

- MOF tools use metamodels to generate code that manages metadata, as XML documents, CORBA objects, Java objects
- Generated code includes access mechanisms, APIs to
  - Read and manipulate
  - Serialize/transform
  - Abstract the details based on access patterns
- Related standards:
  - XML Metadata Interchange (XMI®)
  - CORBA Metadata Interface (CMI)
  - Java Metadata Interface (JMI)
- Metamodels are defined for
  - Relational and hierarchical database modeling
  - Online analytical processing (OLAP)
  - Business process definition, business rules specification
  - XML, UML, and CORBA IDL

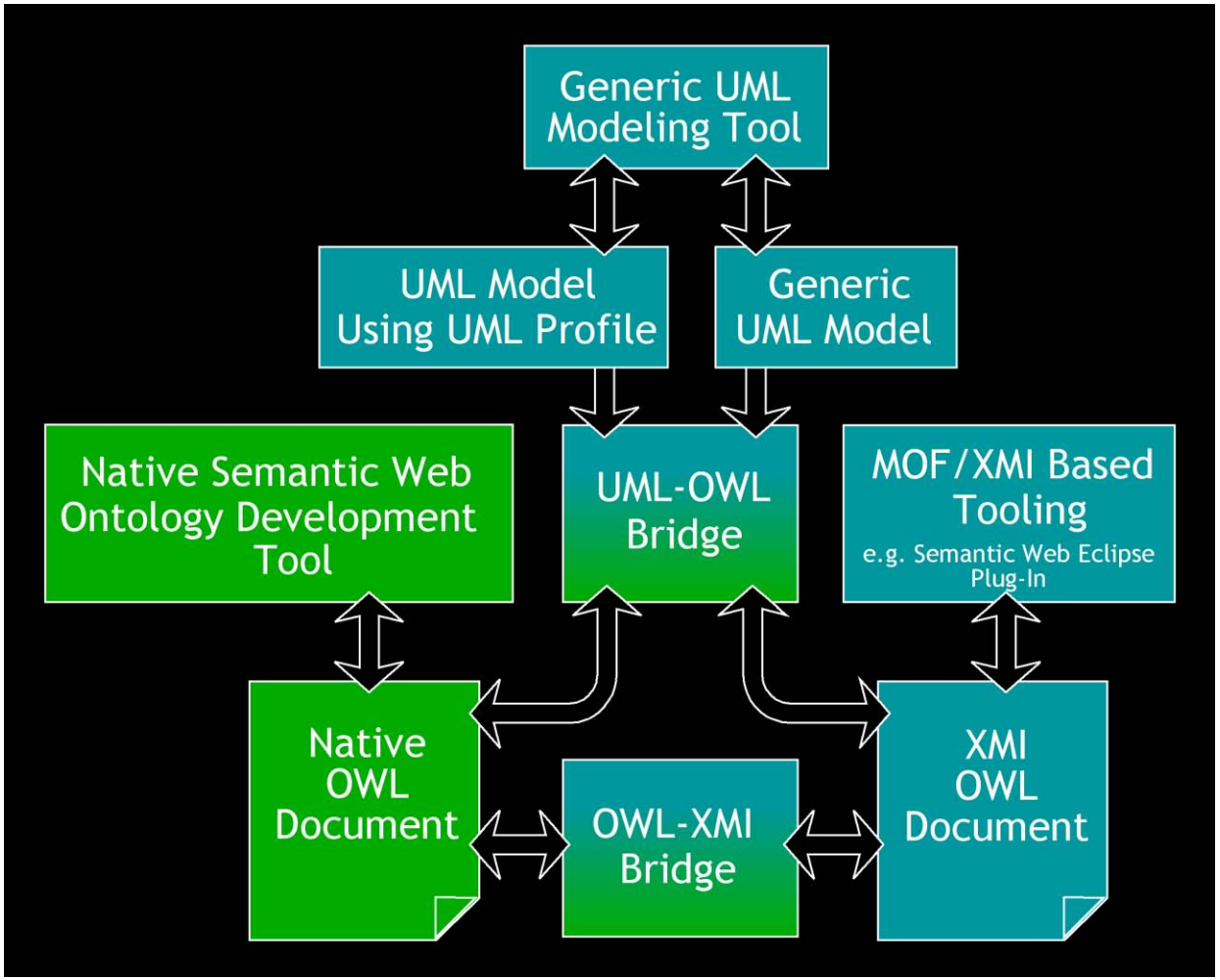


MOF and KR Together

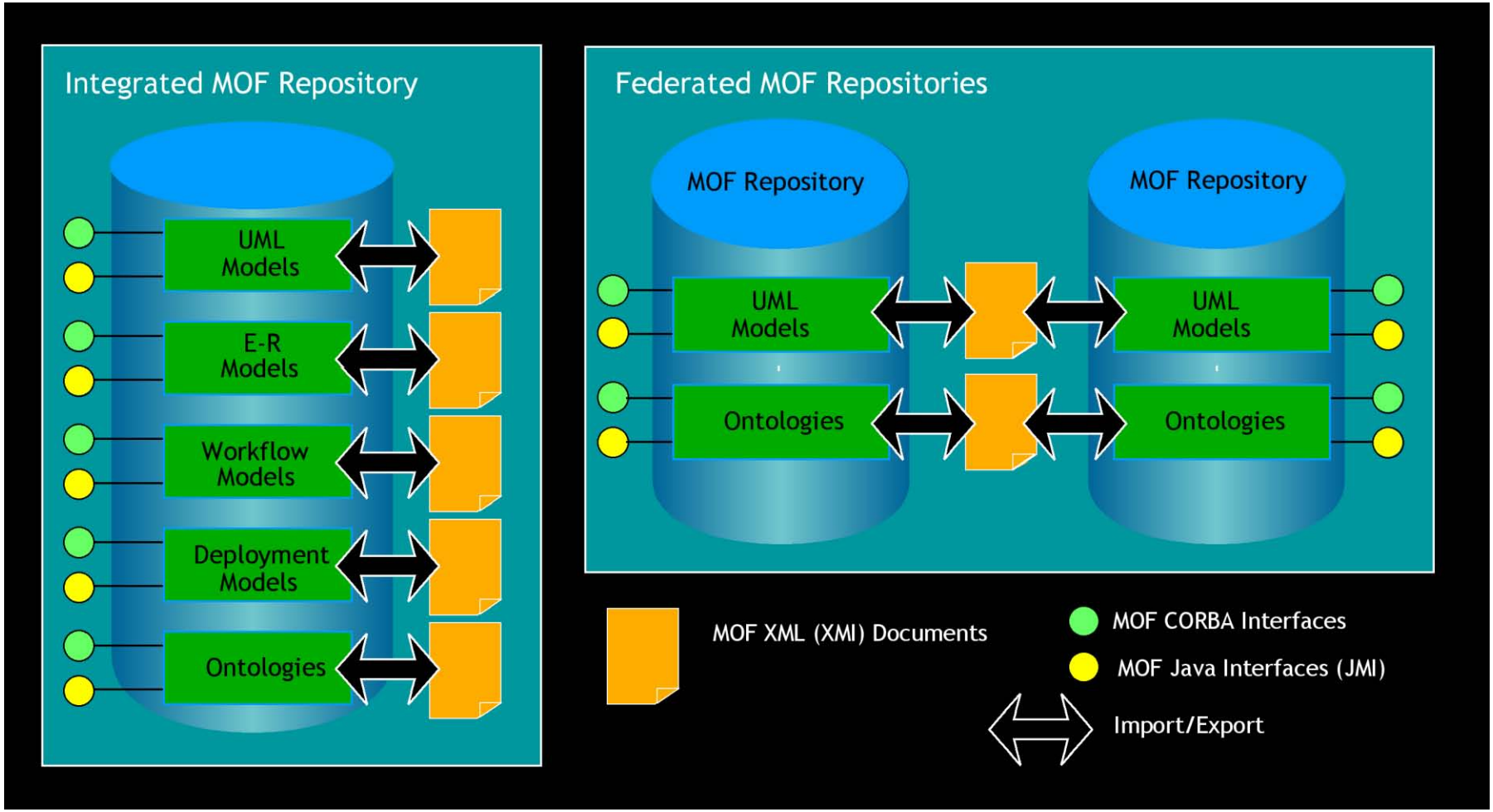
- MOF technology streamlines the *mechanics* of managing models as XML documents, Java objects, CORBA objects
- Knowledge Representation supports *reasoning* about resources
  - Supports semantic alignment among differing vocabularies and nomenclatures
  - Enables consistency checking and model validation, business rule analysis
  - Allows us to ask questions over multiple resources that we could not answer previously
  - Enables policy-driven applications to leverage existing knowledge and policies to solve business problems
    - Detect inconsistent financial transactions
    - Support business policy enforcement
    - Facilitate next generation network management and security applications

  while integrating with existing RDBMS and OLAP data stores
- MOF provides no help with reasoning
- KR is not focused on the mechanics of managing models or metadata
- Complementary technologies – despite some overlap

Bridging KR and MDA



Metadata Management Scenarios

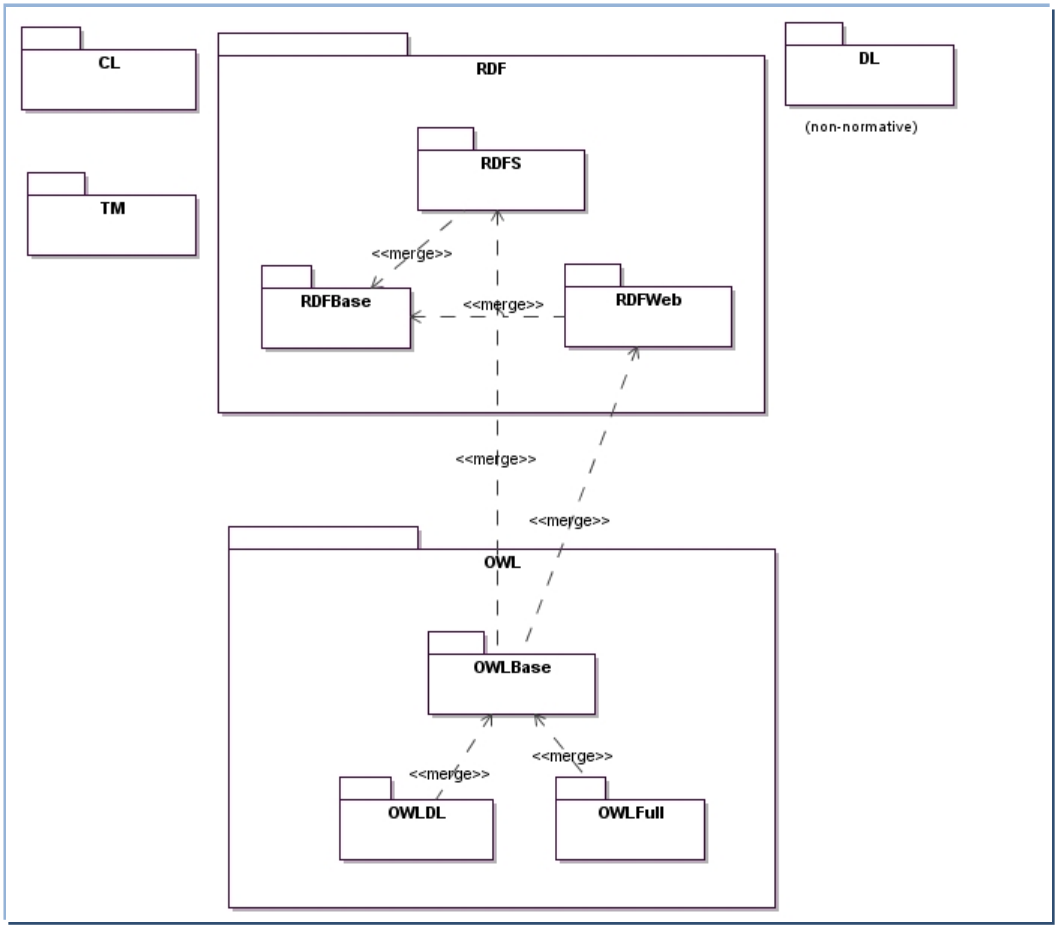


Ontology Definition Metamodel (ODM)

- ODM is Object Management Group's standard for model driven ontology development (adopted in October 2006, finalized in May 2009) – available at http://www.omg.org/spec/ODM/1.0/
- A family of metamodels & profiles that enable model interchange, ontology development in UML 2
- Grounded in formal logic enabling reasoning engines to understand, validate, and apply ontologies developed using the ODM
- Mappings to other OMG standards are either in work or under consideration, including
  - Information Management Metamodel (IMM) for exchange of ER, logical & physical database models, use of database schema for ontology development
  - SysML for exchange of systems engineering models, use of SysML models as a basis for ontology development
  - SoaML for exchange of service models, use of SoaML models as starting points for richer service description development
  - Production Rule Representation (PRR), Semantics for Business Vocabularies and Rules (SBVR) for rule interchange

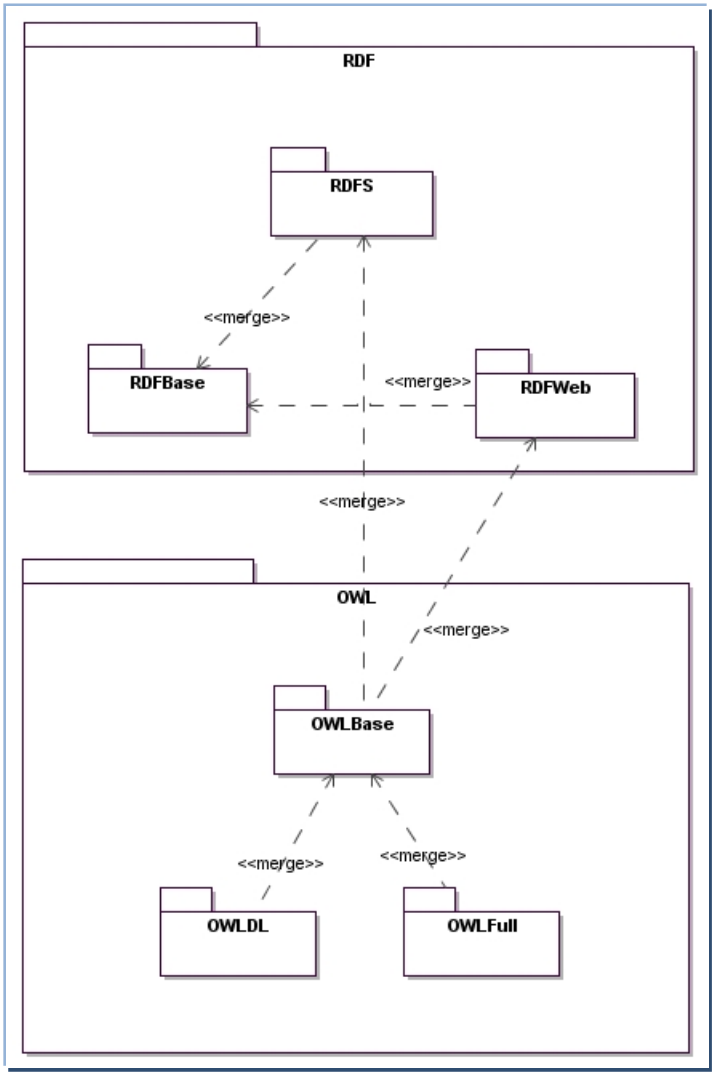
Ontology Definition Metamodel (ODM)

- Five EMOF platform independent metamodels (PIMs), four normative
- Mappings (MOF QVT)
- UML2 Profiles
  - RDFS & OWL
  - TM
- Collateral
  - XMI
  - Java APIs
  - Proof-of-concepts
- Conformance
  - RDFS & OWL
  - Multiple Options
  - TM, CL Optional
  - Informative Mappings



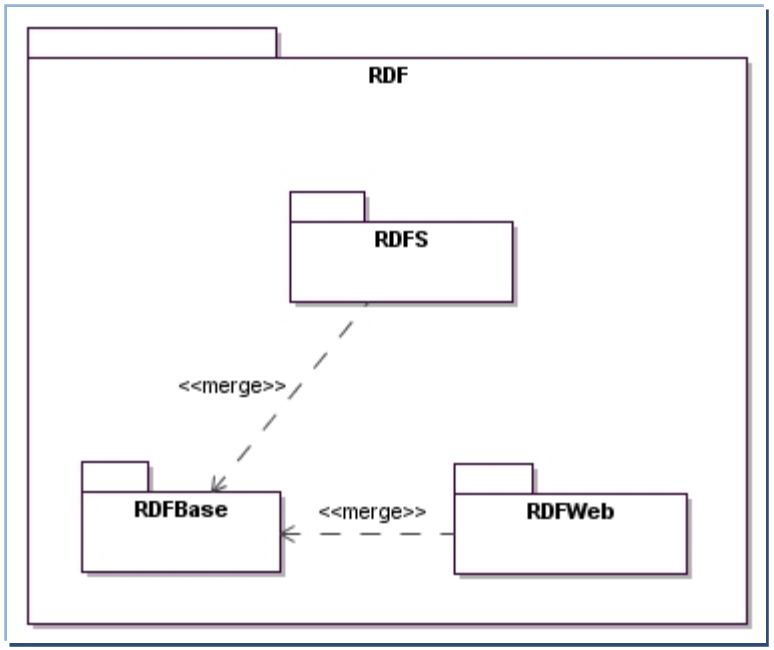
Semantic Web Language Metamodels

- Focus is on abstract syntax of the Resource Description Framework (RDF), RDF Schema, & the Web Ontology Language (OWL)
- Components build on one another, but RDF can be used either standalone or as the basis for OWL ontology development
- Both OWL DL & OWL Full dialects are supported; OWL 2 profile-specific applications can use a subset of constructs from the OWL DL metamodel

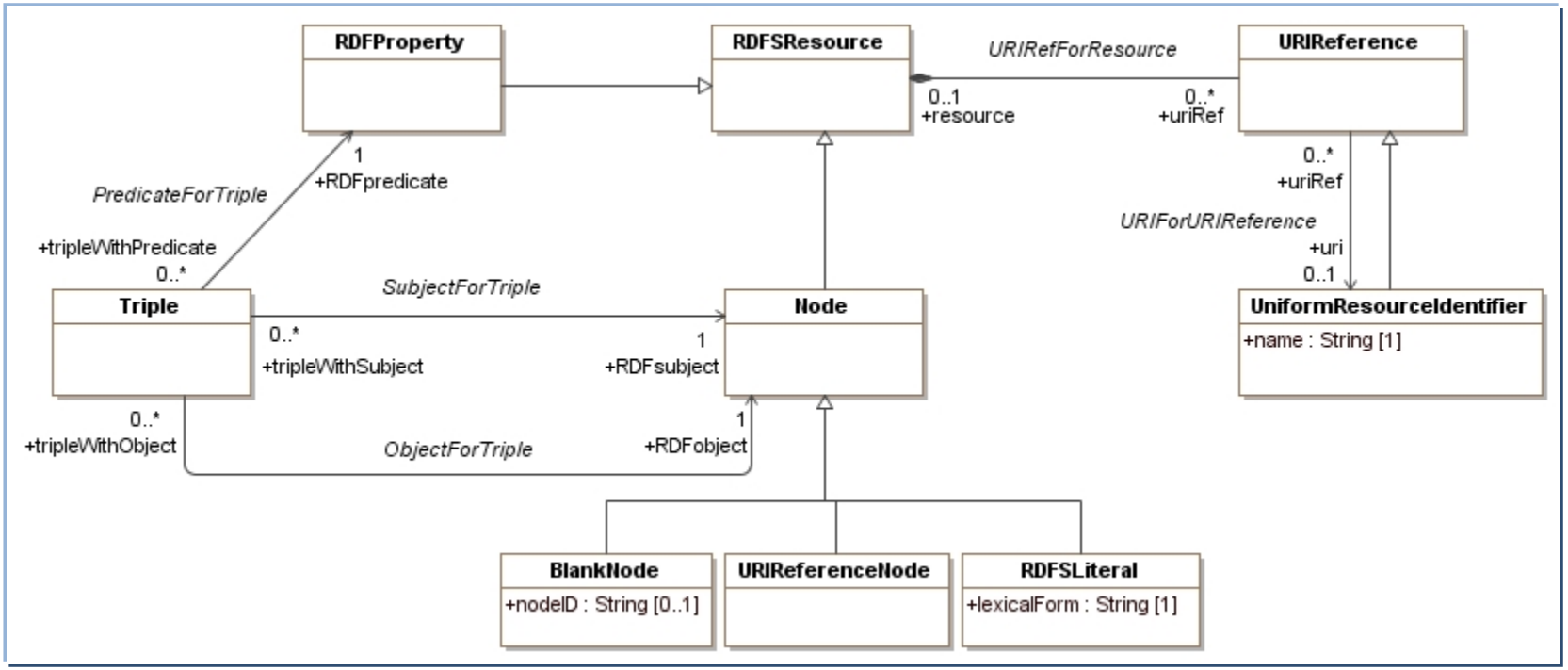


Resource Description Framework (RDF) Metamodel Overview

- RDFBase – primary package
  - Reflects basic abstract syntax from RDF Concepts
  - Minimal implementation requirements, *e.g.*, for RDF triple/quad store
- RDFS – adds vocabulary related to RDF Schema, few additional RDF features
- RDFWeb – fits the model to the Web via document model, required for RDF/XML syntax, among others

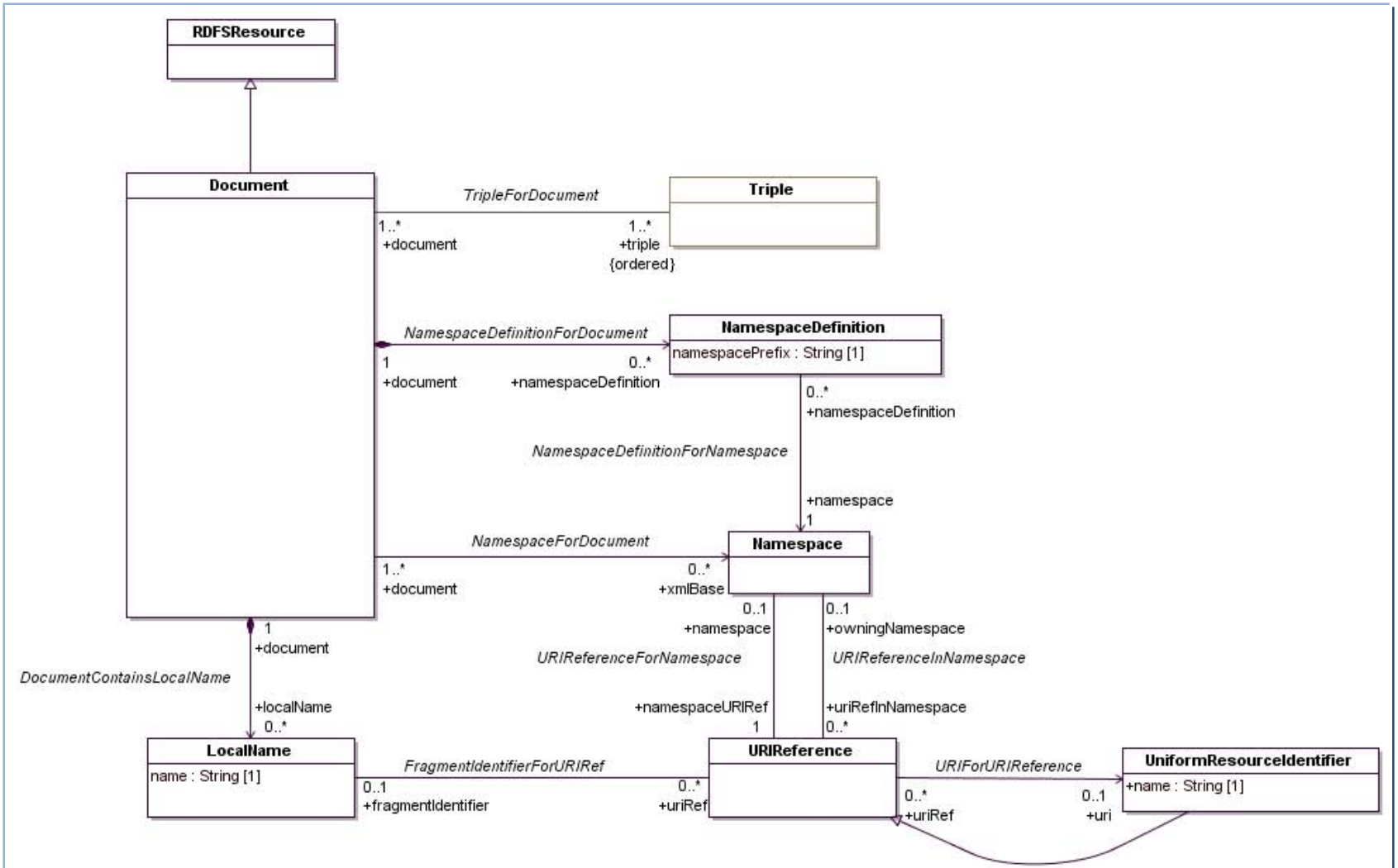


RDFBase Package – Graph Data Model

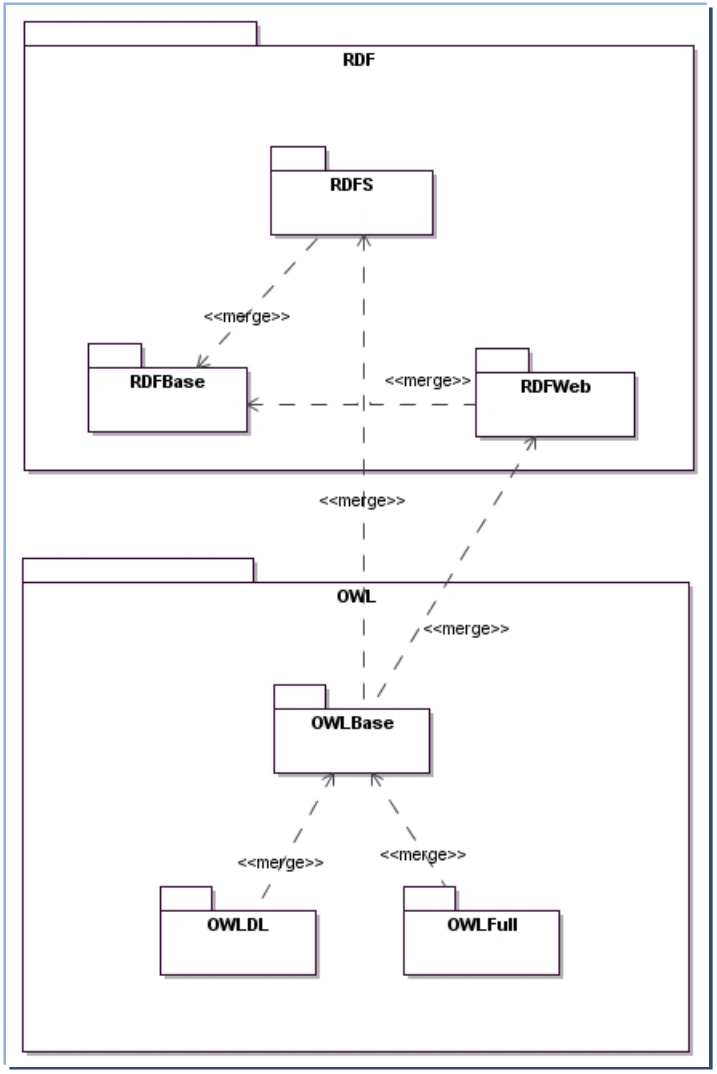


- Supports triple model from RDF (s, p, o), blank node identifiers, essentially RDF basics
- Limited coverage to RDF Concepts document rather than along namespace boundaries, which didn't work from a UML perspective

RDFWeb Package – Documents



Web Ontology Language (OWL) Metamodel Overview



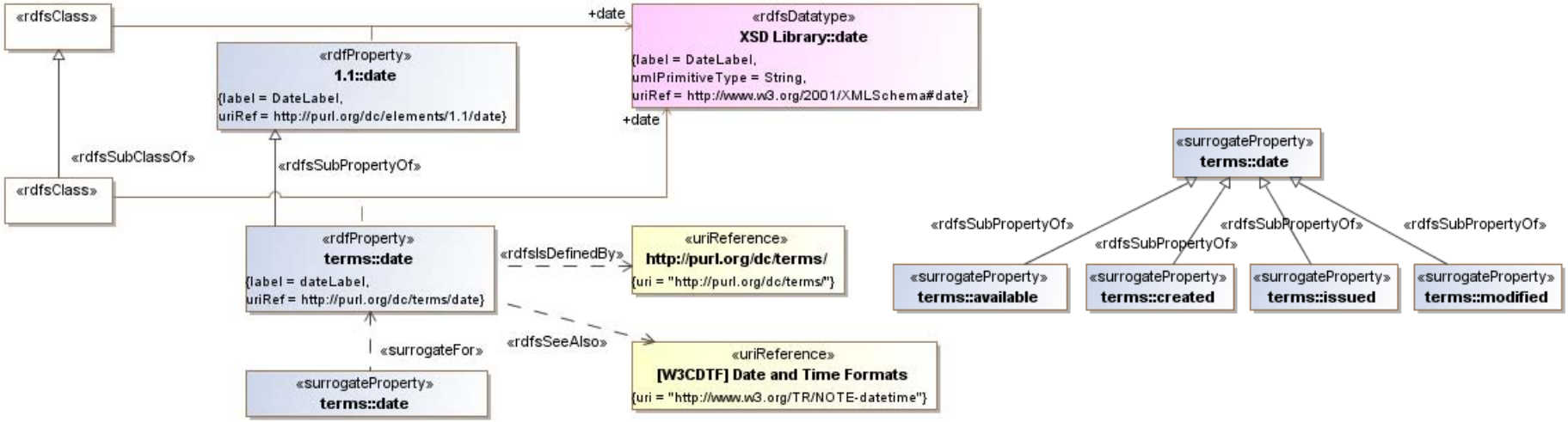
- OWL metamodel components include:
  - OWLBase, covering all common abstract syntax & constraints
  - OWLDL – containing OWL DL constraints
  - OWLFull – containing OWL Full constraints
- Non-normative models for OWL, including changes to property representation & intersection classes for OWL Full, to address MOF multiple classification, are posted to the OMG web site

The UML Profile for RDF & OWL

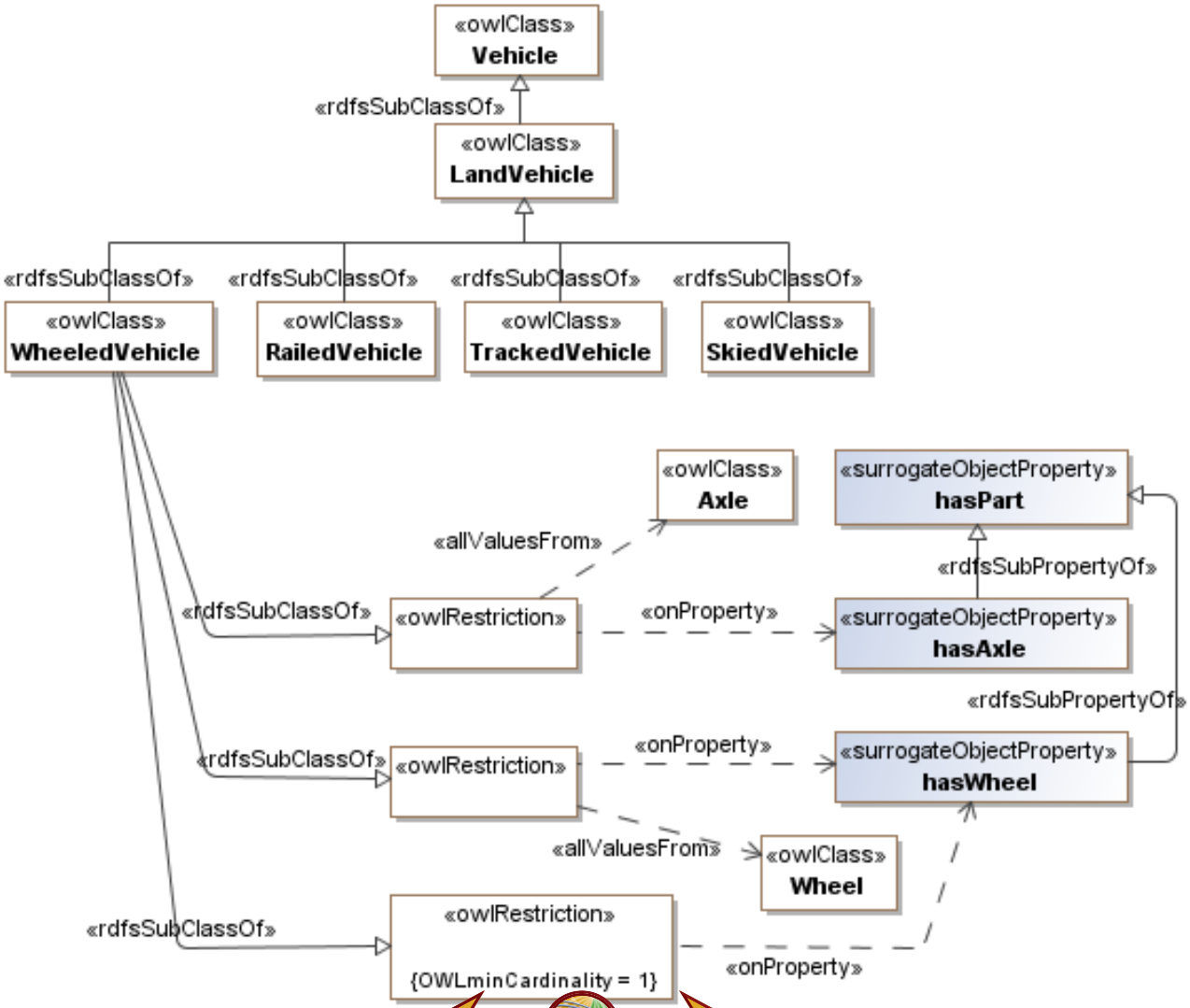
- Intended to be highly intuitive for UML users
- Reuses UML constructs when they have the same semantics as OWL
- When this is not possible, stereotypes UML constructs that are consistent and as close as possible to OWL semantics
- Uses standard UML 2 notation
- In the few cases where this is not possible, follows the clarifications and elaborations of stereotype notation defined in UML 2.1
- Leverages the model library included in Appendix A for a number of constructs, for example statements, rdf:value, container and list elements, as well as built-in properties

Key Features of the RDF Profile

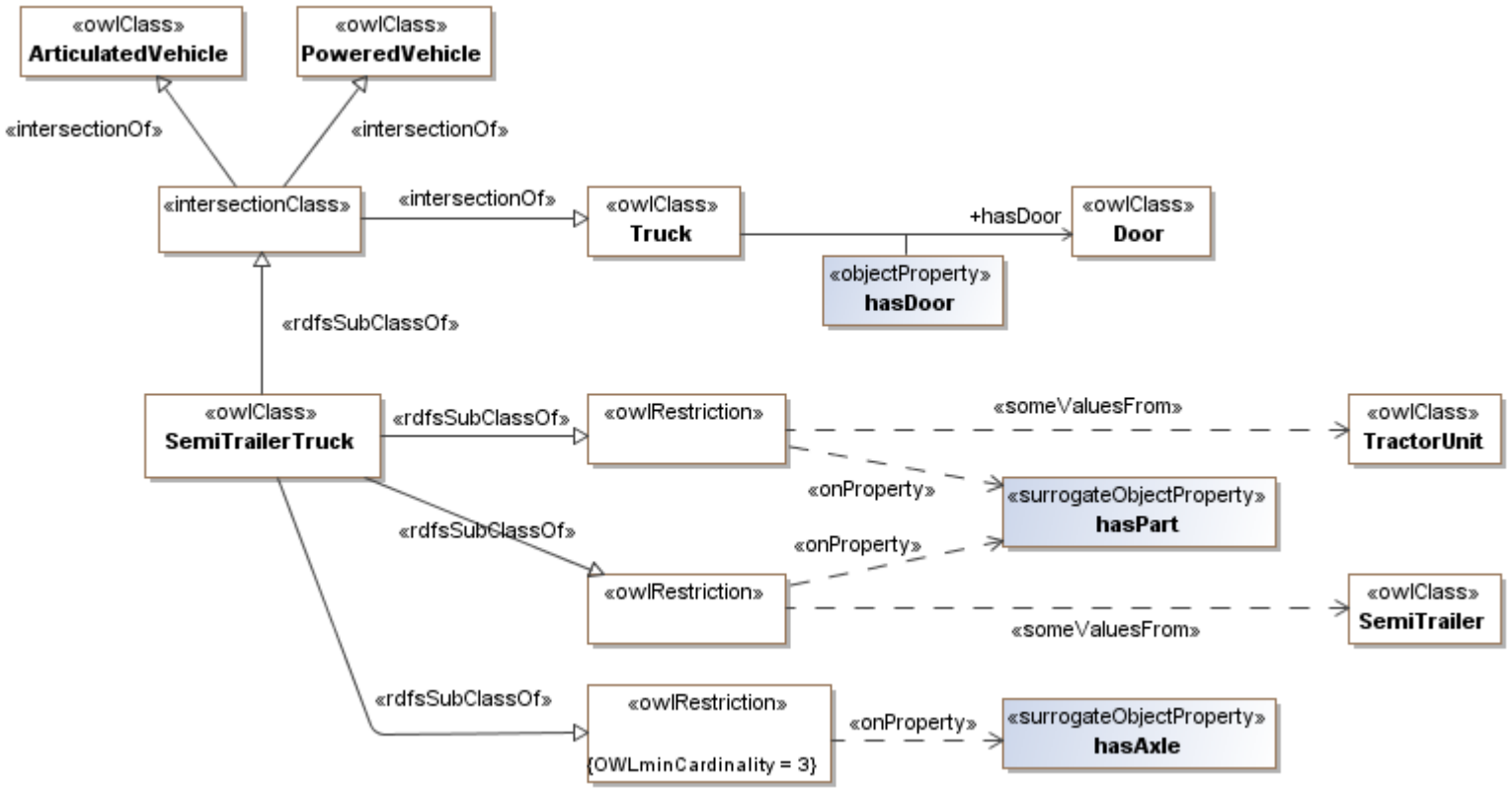
- rdf:resource is modeled as UML::InstanceSpecification
- Introduction of «reifies» stereotype of UML::Dependency to allow such instance specifications to reify classes, properties, individuals, statements, etc.
- rdf:Property is modeled as UML::AssociationClass, UML::Association, *and* UML::Property, to provide greatest possible flexibility
- Several possible representations of various aspects of rdf:Property



OWL Value, Cardinality Restrictions



OWL Intersection, Domain & Range



Relationship to ISO Standards

- CL Metamodel is identical to the UML diagrams in ISO 24707
- High degree of synergy between ODM and Topic Maps ISO 13250 working group
- Current work in ISO JTC1 SC32 to update ISO 11179 (Metadata Registration) references ODM; also addressing alignment with SKOS (Simple Knowledge Organization System) and Dublin Core
- All ODM metamodels are referenced and used in ISO CD 19763 (MMF – Metamodel Framework, Model Registry specification)
- Mappings from multiple components of IMM (*e.g.*, ER, ISO Express, W3C XML Schema, etc.) are planned

Planned Extensions

- W3C is moving the ball forward on a number of relevant fronts: RDF Query (SPARQL), Rules (RIF), OWL2
- Planned revision of ODM will support OWL 2 (in work)
- Ontology PSIG roadmap includes MOF revisions to support multiple classification (SMOF)
- RFP published in Minneapolis (June 2010) to support APIs for knowledge base access
- Extensions under consideration include mappings to
  - SysML
  - Production Rule Representation (PRR) specification
  - IMM Metamodels (ER, XML Schema …)
  - "SoaML meets Semantic Technologies"
- OMG BMI DTF Semantics for Business Vocabularies & Rules (SBVR)
  - logical grounding in Common Logic / ODM CL Metamodel
  - direct mapping to OWL
  - Date Time vocabulary under development as test case

Benefits of UML-Based Ontology Tools

- Well-known graphical notation for collaboration, sharing, and maintenance of complex ontologies and other models
- Interoperable with other modeling languages, including Entity-Relationship (ER), business process, and data modeling paradigms – allows users to leverage existing artifacts as a basis for ontology development
- Native development framework supports ontology reuse in downstream rule, software, and service development
- Eclipse-based architecture enables seamless integration of knowledge bases, reasoners, transformation and other services
- A large and growing community of technologists who are familiar with the UML notation and tools

Additional Metadata Standards for Definitions & Registries

- NASA & JPL Taxonomies for document-related asset management, navigation extend
  - Dublin Core Metadata Terms (DCMI, http://www.dublincore.org/)
  - Simple Knowledge Organization System (W3C, http://www.w3.org/TR/skos-reference/ – now candidate recommendation)
- DISA approach to metadata for other asset types will reuse & extend these vocabularies where applicable
- Registry metadata will extend ISO 11179-3 Metadata Registry, ISO 19763 Model Registry standards as appropriate
- Current approach for planetary science data store (PDS) uses ISO 11179 Edition 2, may be updated to support emerging Edition 3

Part 3: Integrating Ontologies & Systems Engineering via SysML

The Challenge

- JPL is developing formal OWL ontologies for flight project development (technical and programmatic)
  - To normalize terminology for human communication
  - To assist and support data exchange among information systems
    - Including, but not limited to, SysML modeling tools
- We want to see the concepts and properties from our ontologies in our SysML modeling tools
  - To express precise semantics in SysML
  - To sustain our consensus terminology through regular use

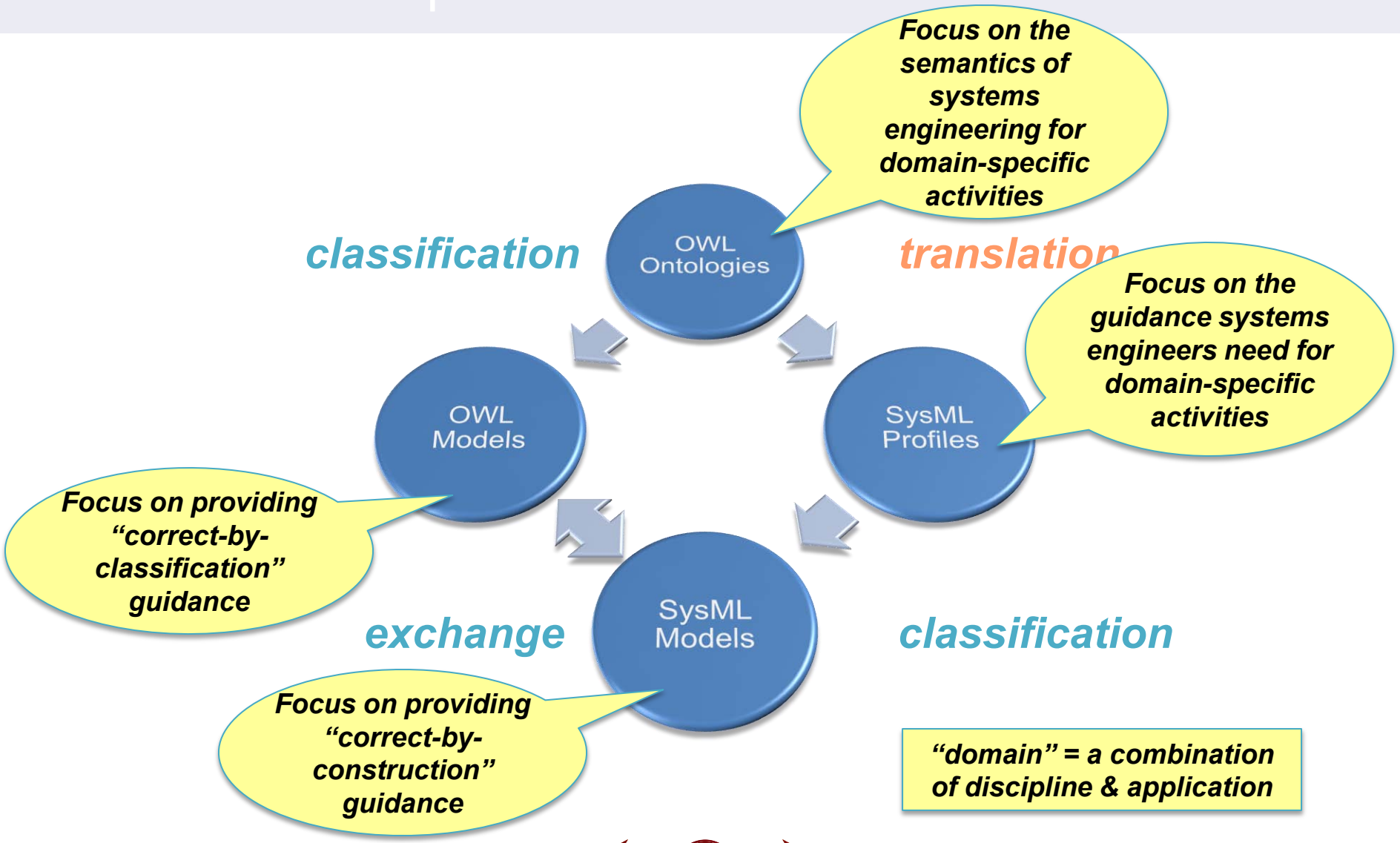
Therefore….

- We need (at least) to translate OWL ontologies into SysML profiles

Someday….

- We want full bidirectional interchange of models—including instances

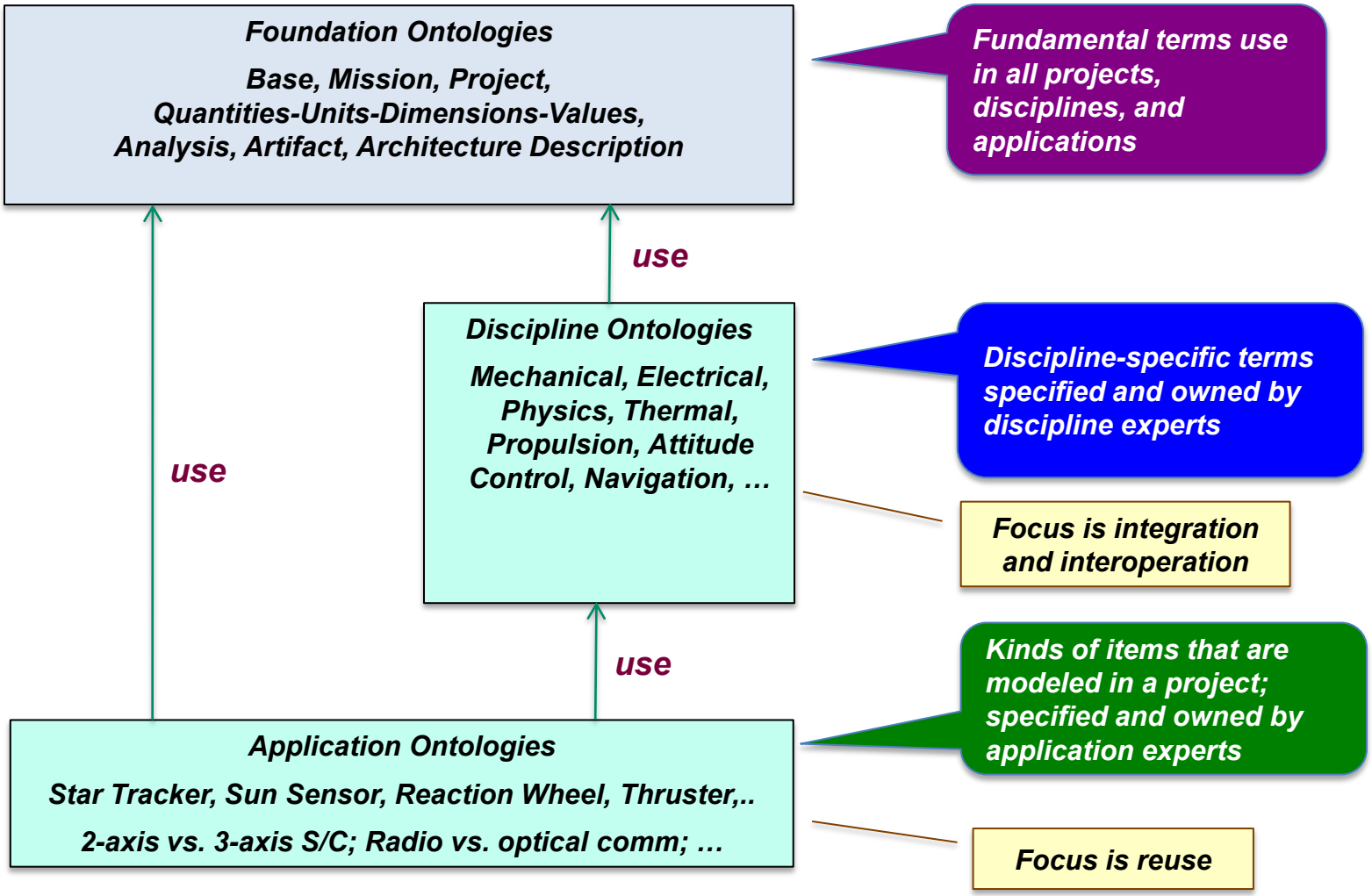
The Challenge



A Quick Look at JPL OWL Ontologies

- Foundation Ontologies
  - Establish broad concepts and properties in systems engineering and space flight
  - Provides the basis for aligning & integrating discipline-specific views & application-specific problem domains
- Discipline Ontologies
  - Establish concept and property definitions for discipline-specific views, particularly those with widespread applicability
  - E.g., all spacecraft subsystems have mass properties
  - Shared viewpoints encourage model reuse
- Application Ontologies
  - Establish concept and property definitions for application-specific problem domains
  - E.g, Spacecraft, Telecom Subsystem, Transponder, etc.
  - Recurring problems encourage model reuse

Ontology Organization



Example Foundation Concepts

- Component
  - Object that performs one or more functions and presents zero or more interfaces that define its connections to the outside world
  - Examples: launch vehicle, spacecraft, telecom subsystem, flight software, attitude control software, and mission operations team
- Interface
  - A set of mechanical, electrical, signal, or other properties that describe some aspect of a component's connection to or interaction with another component
  - Examples: spacecraft to launch vehicle, launch vehicle to spacecraft, battery terminals

Example Foundation Concepts

- Requirement
  - An assertion about a Component, Function, or Interface that must be true for every acceptable realization of that element
  - Examples:
    - The spacecraft bus main structure shall be aluminum.
    - The spacecraft shall provide 300 W to instruments.
    - The mission shall conform to CCSDS telecom standards.
- Work Element
  - Discrete unit of project authority, cost, schedule, and activity
  - Node in the project *Work Breakdown Structure*

A Simple Mapping Approach

- Map every OWL Class to a stereotype extending SysML::Block
  - E.g., mission:Component → «Component»
- Map every OWL Data Property to a SysML value property
  - E.g., physics:mass → mass : Float
- Map every OWL Object Property to a SysML reference part
  - E.g., mission:performs → «performs»

- Would this work in general? It could work…
- Would it be practical? It would ignore non-Block concepts already in SysML
  - A simple, mechanistic approach fails to acknowledge the ontological commitments implicitly made in SysML

- Let's take a closer look

Aligning Ontological Commitments

- Function
  - SysML strives to be methodology-neutral; thus a function can be modeled using several behavioral constructs
  - Our Function is a specialization of UML::Activity
  - We retain the name *Function* to conform to local usage and to distinguish distinct Activity types (e.g., Process)
- Interface
  - UML's Interface concept is intended for *declaring* a contract but the parties involved in that contract are implicit
  - Our interface concept extends SysML::Block for explicitly *specifying* a contract including the involved parties
- Requirement
  - Requirement modeling is well supported in SysML
  - Our Requirement specializes SysML::Requirement to benefit from current SysML practice and differentiate conceptual requirements from other kinds of requirements

Work Element

- A Work Breakdown Structure is, conceptually, a tree of responsibility and authority
  - Each node (Work Element) has the responsibility to produce certain deliverables
  - Each Work Element has authority to make design decisions, expend resources, contract for its own deliverables, etc.
  - An element should appear in a model only by decision of a unique and explicitly identified authority
- Some notions attached to a Work Element:
  - Naming: A Work Element names objects within its design authority
  - Access Control: A Work Element controls who can read/write model elements within its design authority
  - Delegation: A Work Element authorizes other Work Elements
- UML::Package matches the ontological commitments intrinsic in the concept of Work Element

Example Concept Mappings

OWL Concept	Ontological Commitments	specializes	extends
Component	performs Function, presents Interface	UML::Class	SysML::Block
Function	performed by Component	UML::Activity	—
Interface	presented by Component, mates with Interface	UML::Class	SysML::Block
Requirement	specifies Component, Function, Interface	UML::Class	SysML::Requirement
Work Element	authorizes multiple elements exclusively	UML::Package	—

Annotation-Driven Transformation

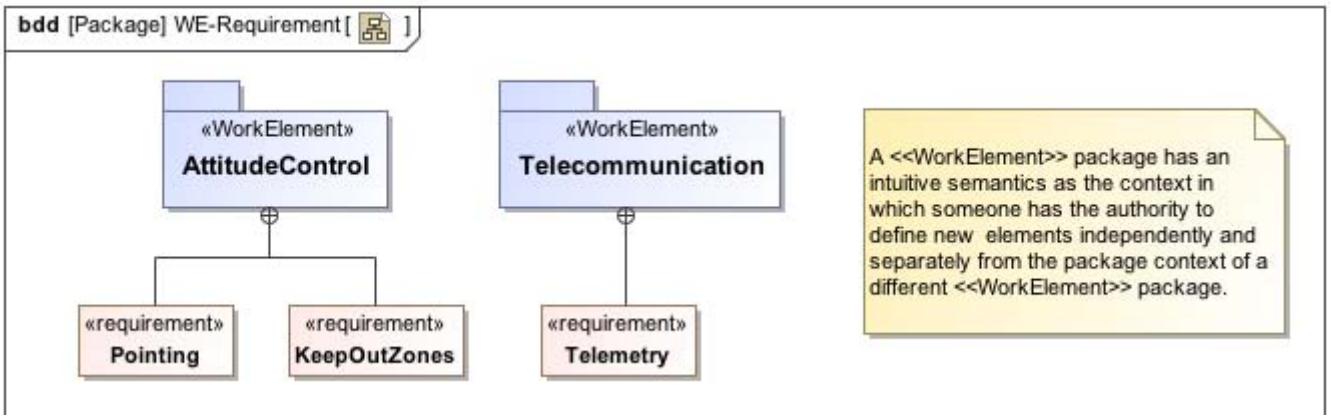
- The conceptual mapping approach suggests a strategy for automated transformation of ontologies into profiles
- Define two OWL Annotation Properties
  - specializesMetaclass
  - extendsStereotype
- Annotate OWL Class Declarations with these properties
- Transform by this algorithm:
  - Parse OWL ontology
  - For each class declaration
    - Create a matching SysML stereotype
    - Apply attributes as specified by annotation properties
  - Emit SysML profile
- This works for classes
- Object properties in the conceptual ontology make the mapping strategy more complex….

Object Property Mapping

- Some Mapping Options
  - Specialize an existing meta-association (e.g., containment)
  - Extend an existing meta-relationship (e.g. «depend»)
  - Create a reference property
  - Create an association block
- Which one is right depends (again) on ontological commitments
  - Which one best matches the intended meaning?
  - Which one results in a profile that is conceptually intuitive?
- Practical issues
  - Meta-associations can't be specialized in a profile
    - Implemented in tool-specific customization
  - Mapping options probably require some tool customization
  - Constraints on association end ownership in a profile
  - Mapping object property inverses
  - Some OWL2 restrictions are better mapped as OCL constraints

Work Elements and Authorization

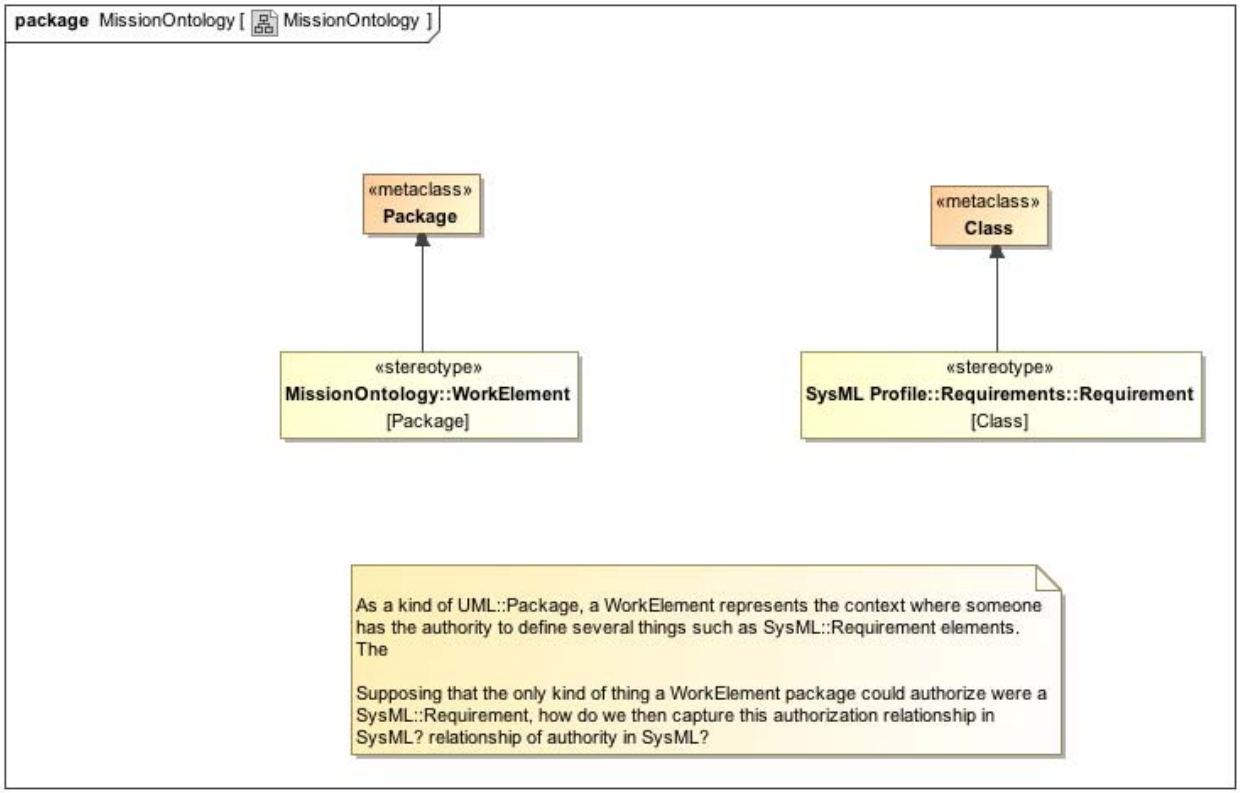
- We want to convey that a Requirement is authorized by a Work Element
- Let e_w be a Work Element; let *r* be a Requirement
- We'd like these statements to be equivalent:
  - e_w is the innermost Work Element containing *r*
  - e_w authorizes *r*



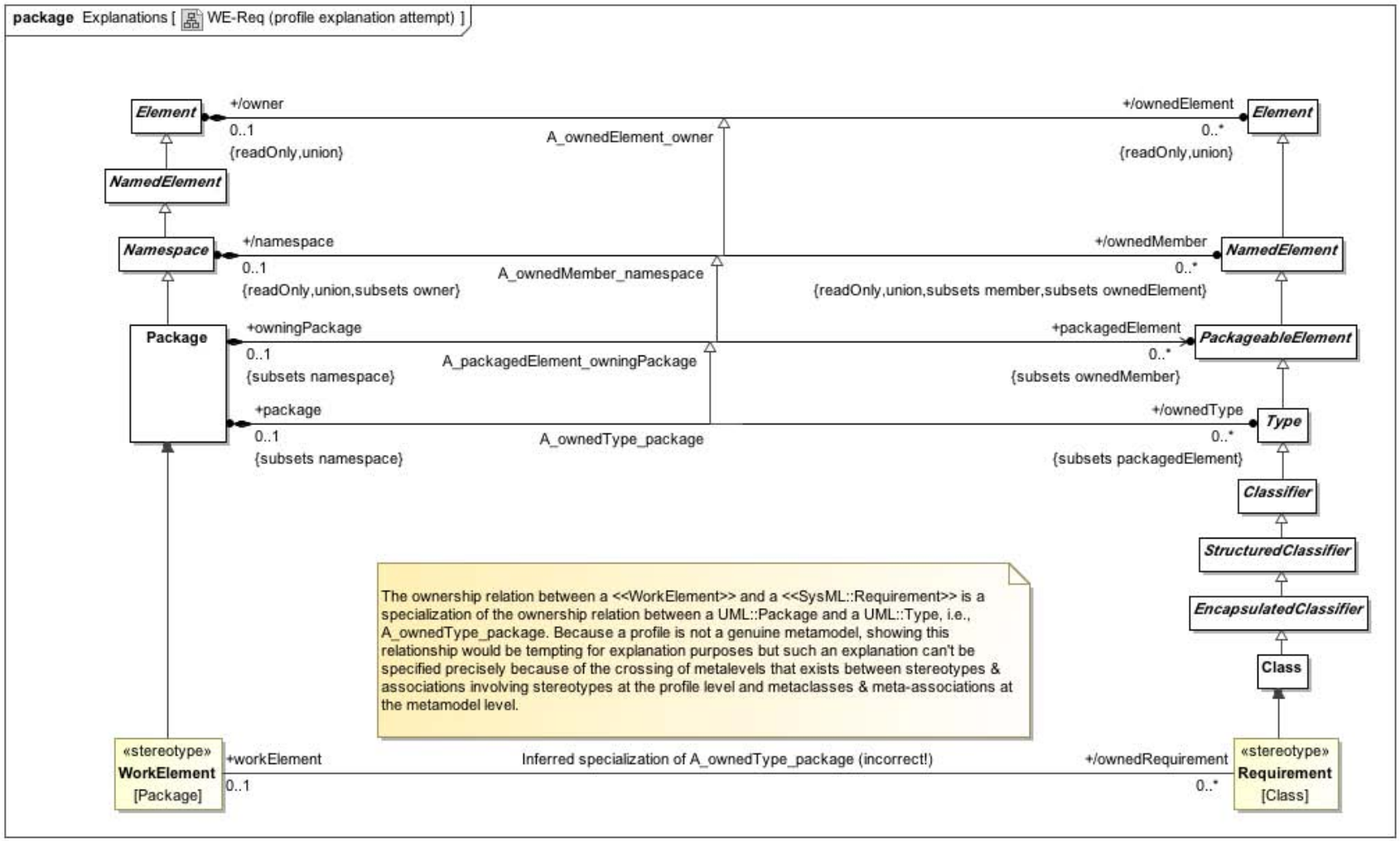
- Example:
  - Telecomunication *authorizes* Telementry

Inside the Profile

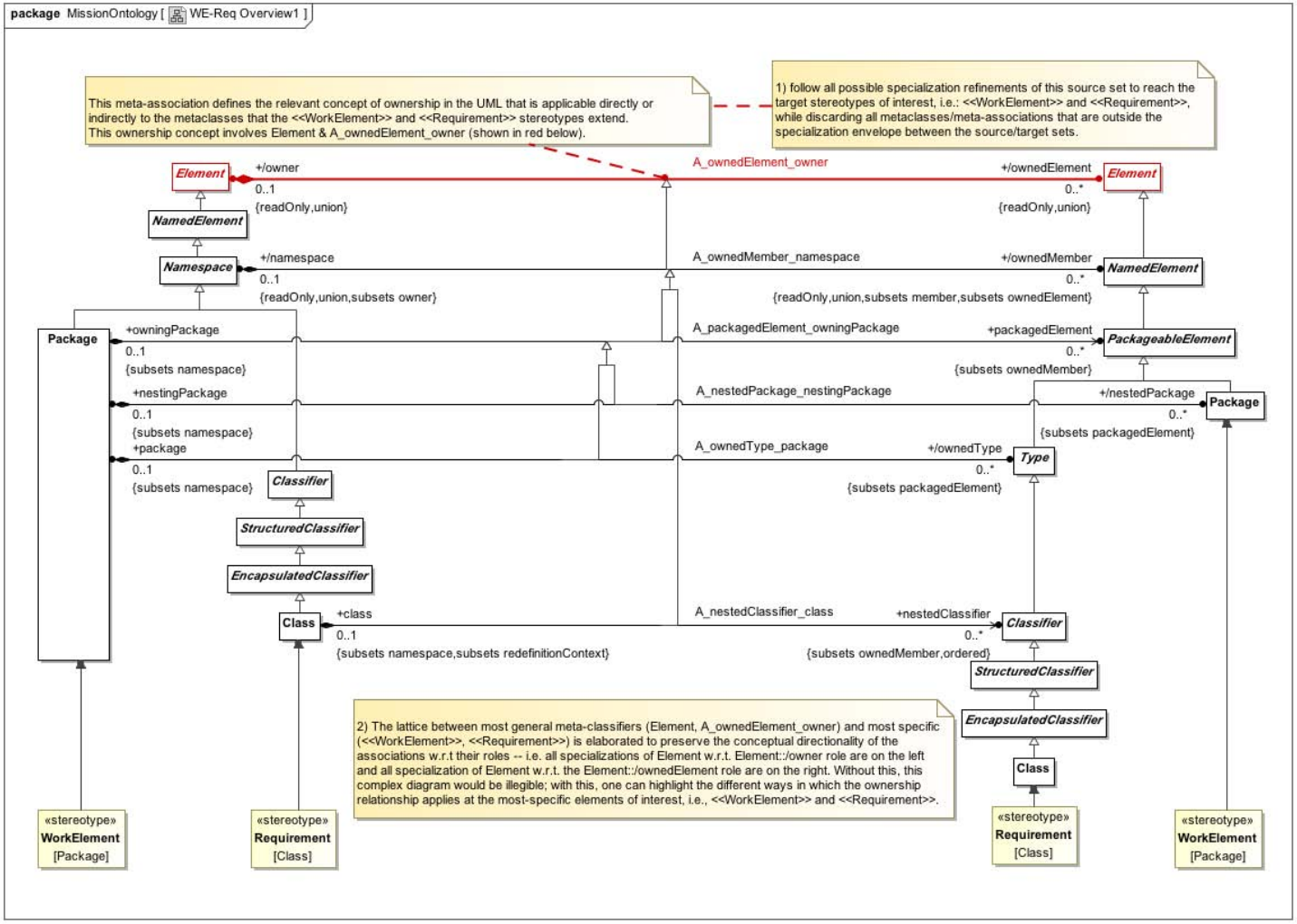
- Recall that Work Element specializes Package; Requirement specializes Class



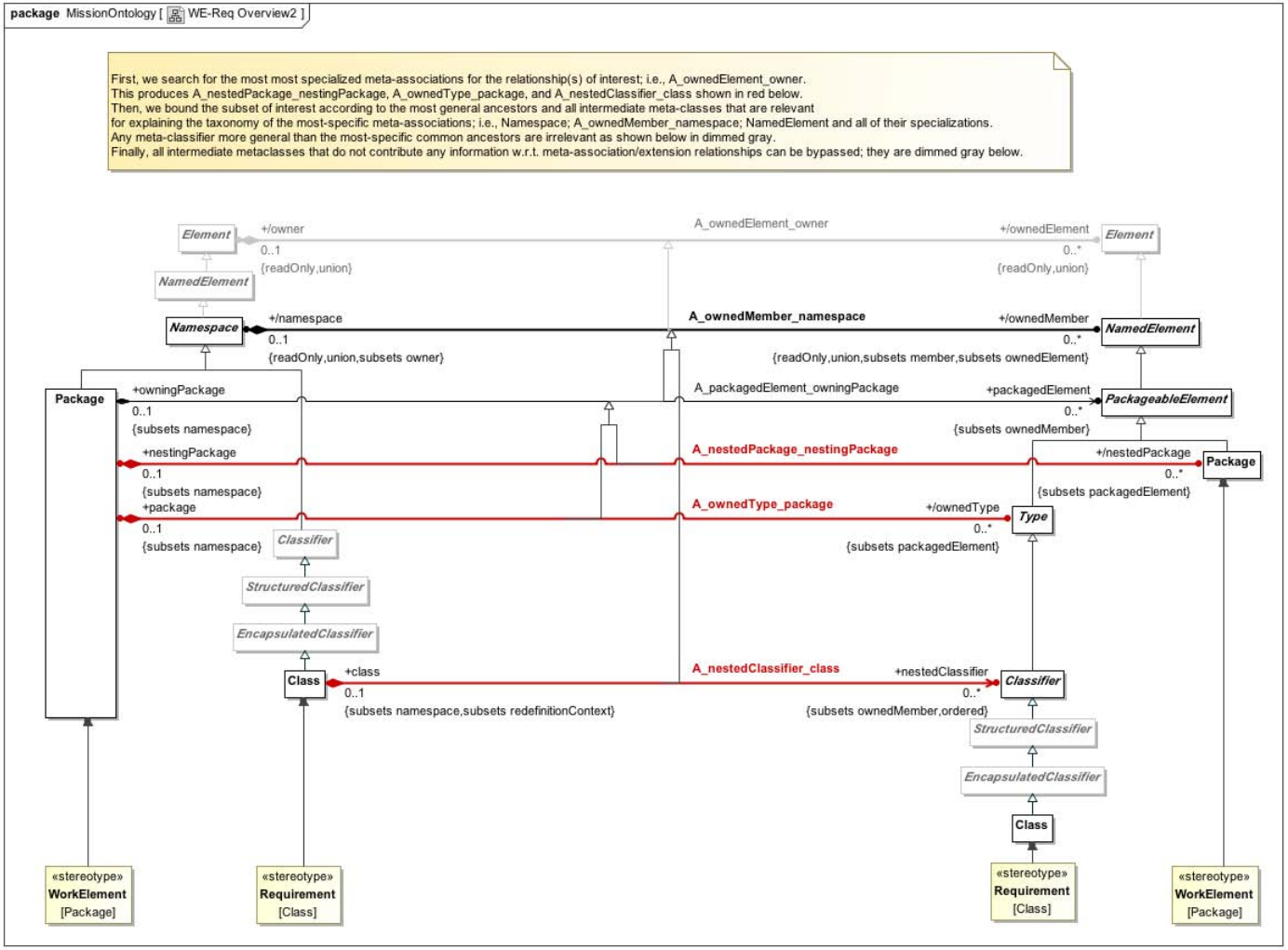
Issue: No Meta-Association Specialization



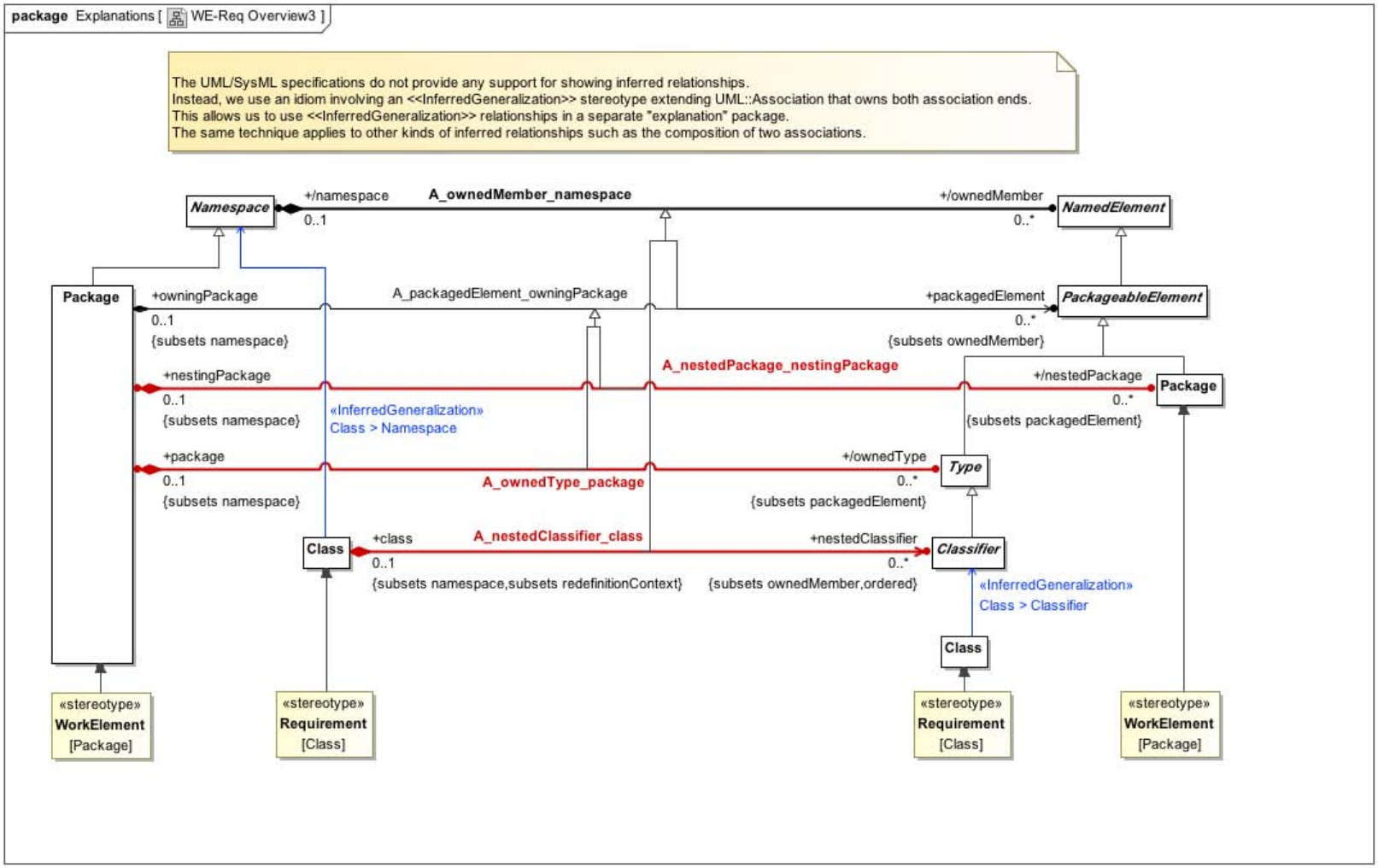
Association Strategy Step 1



Association Strategy Step 2



Association Strategy Step 3

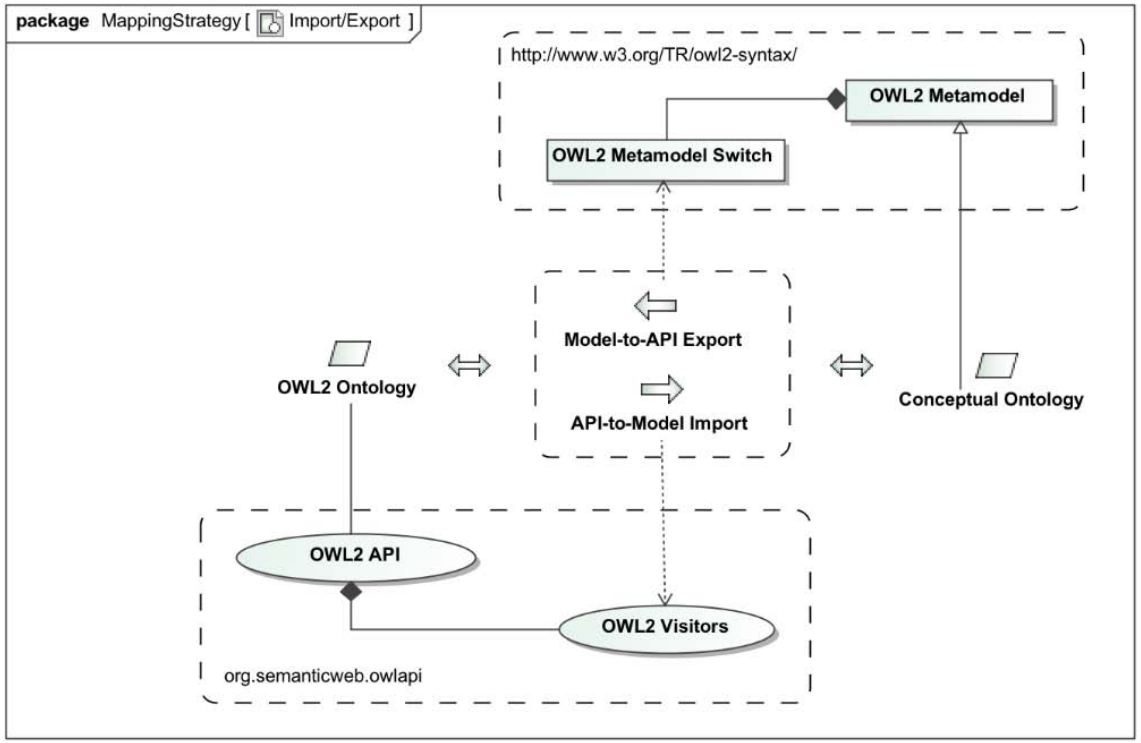


Same Profiling Strategy, New Choices

- Annotate OWL object properties to inform the generator whether to
  - Specialize an existing meta-association
  - Extend an existing meta-relationship
  - Create a reference property
  - Create an association block
- Annotations are also required for
  - Determining association end ownership
  - Relating inverse properties
- Getting this right requires
  - Understanding our own ontological commitments
  - Understanding the UML and SysML metamodels

Implementation Overview: Integrating Ontologies in UML/SysML

- Import/export conversion between ontologies serialized in W3C syntax vs. represented as instance models of the OWL2 metamodel



OWL2 metamodel extracted from the W3C spec

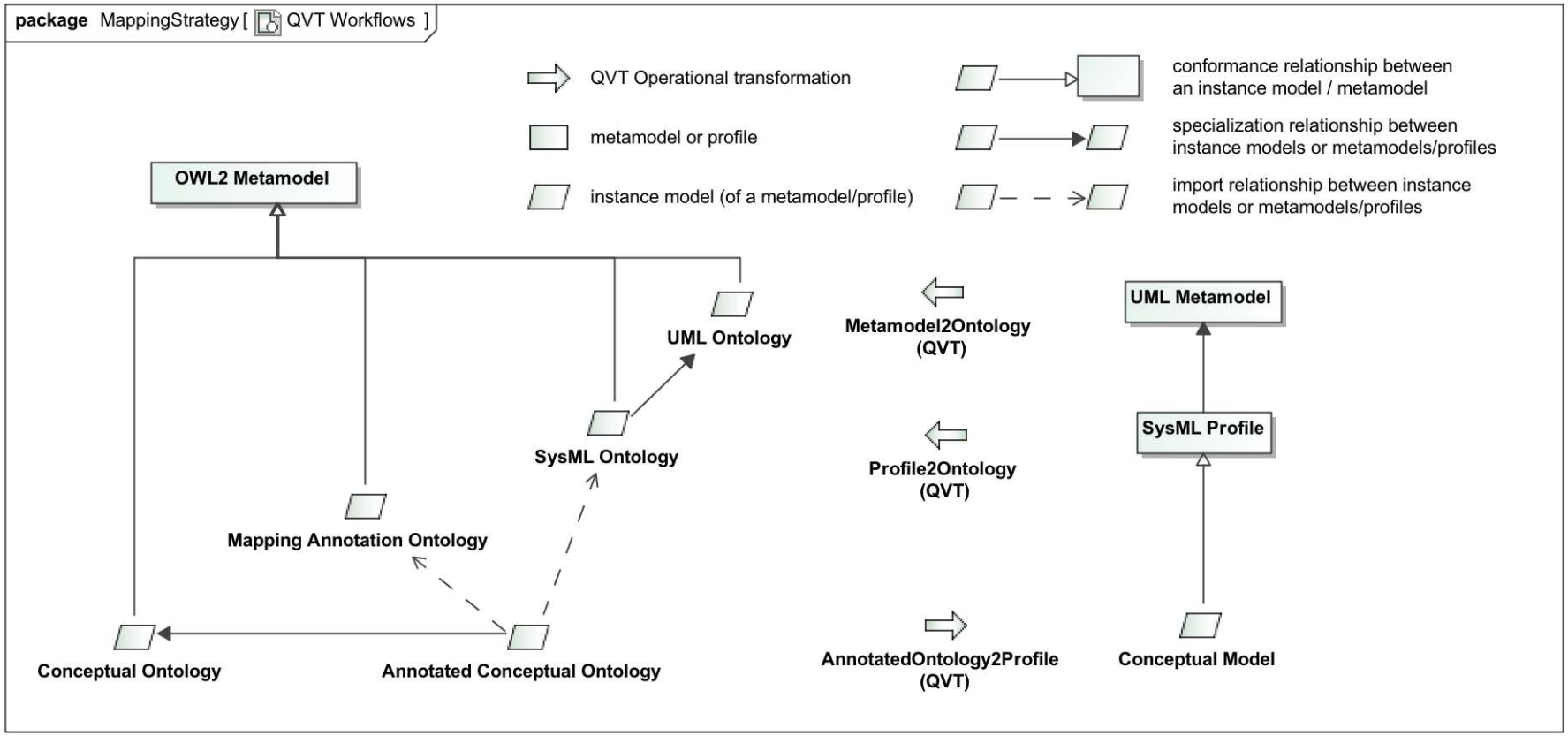
Generated EMF-based OWL2 metamodel API

Importer constructs an OWL2 model (instance of the OWL2 metamodel) by visiting it

Exporter uses the OWL2 metamodel switch to guide its construction via the OWL2 semantic web factory

Implementation Overview: QVT Workflows

- Workflows implemented with Eclipse Helios' M2M QVTo



Status and Future Work

- Current prototypes:
  - OWL2 Semantic Web / OWL2 MM import/export
  - Mapping Annotation Ontology
    - Supports mapping of class concepts
  - Metamodel2Ontology QVTo transformation
  - AnnotatedOntology2Profile QVTo transformation
    - Supports current Annotation Mapping Ontology
- Future Work
  - Mapping Annotation Ontology
    - Patterns for mapping object & data properties
  - AnnotatedOntology2Profile QVTo transformation
    - Add support for object/data property mapping
  - Live synchronization of SysML tools & OWL2 repositories
    - Transform the AnnotatedOntology2Profile trace as a PIM-level specification into PSM-level synchronization logic for SysML tools & OWL2 repositories (e.g., change listeners)
  - Alignment with evolving Ontology Definition Metamodel