OWL2 & SysML

Nicolas Rouquette
(Joint work with Steven Jenkins)
Jet Propulsion Laboratory
California Institute of Technology

September 2010

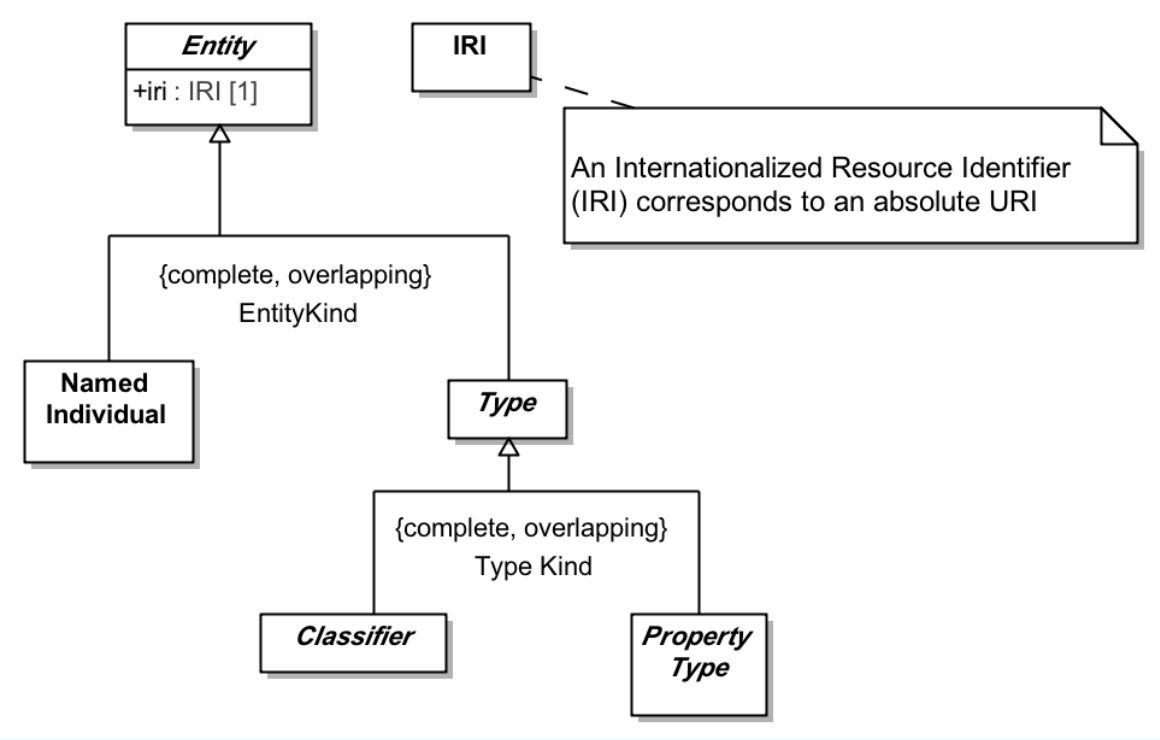
© 2010 California Institute of Technology.
Government sponsorship acknowledged.

Overview

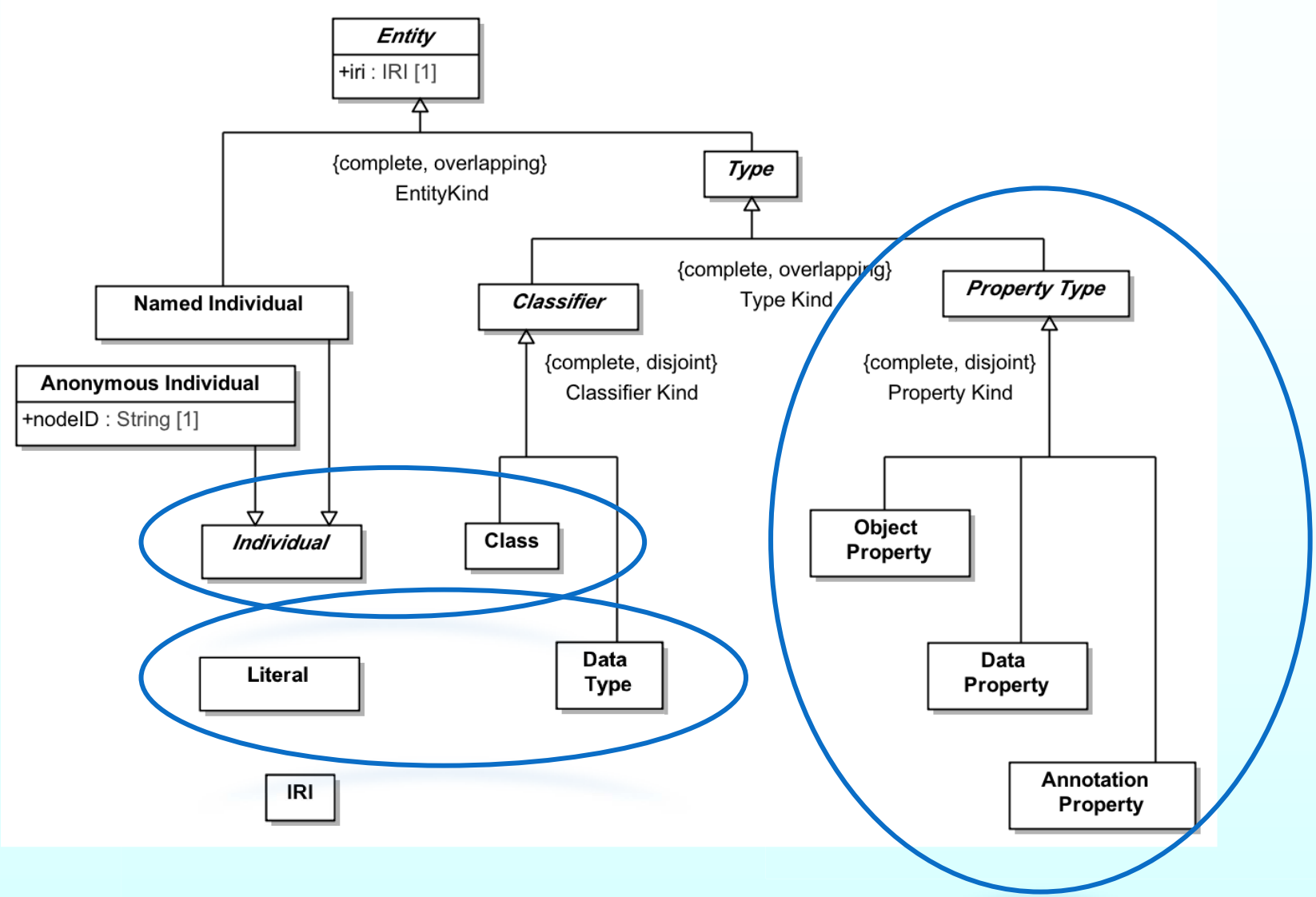
- OWL2 Concepts
- Classification in OWL2 vs. UML2
- Mismatches between OWL2 & UML2
  - Classifier, Instance, Property
- Simplified UML/SysML
- Overview of the simplified OWL2/UML2 mapping
- Essential vs. Complicated MOF/OWL2
- QUDV Example
- Why?

OWL2 – A kernel perspective

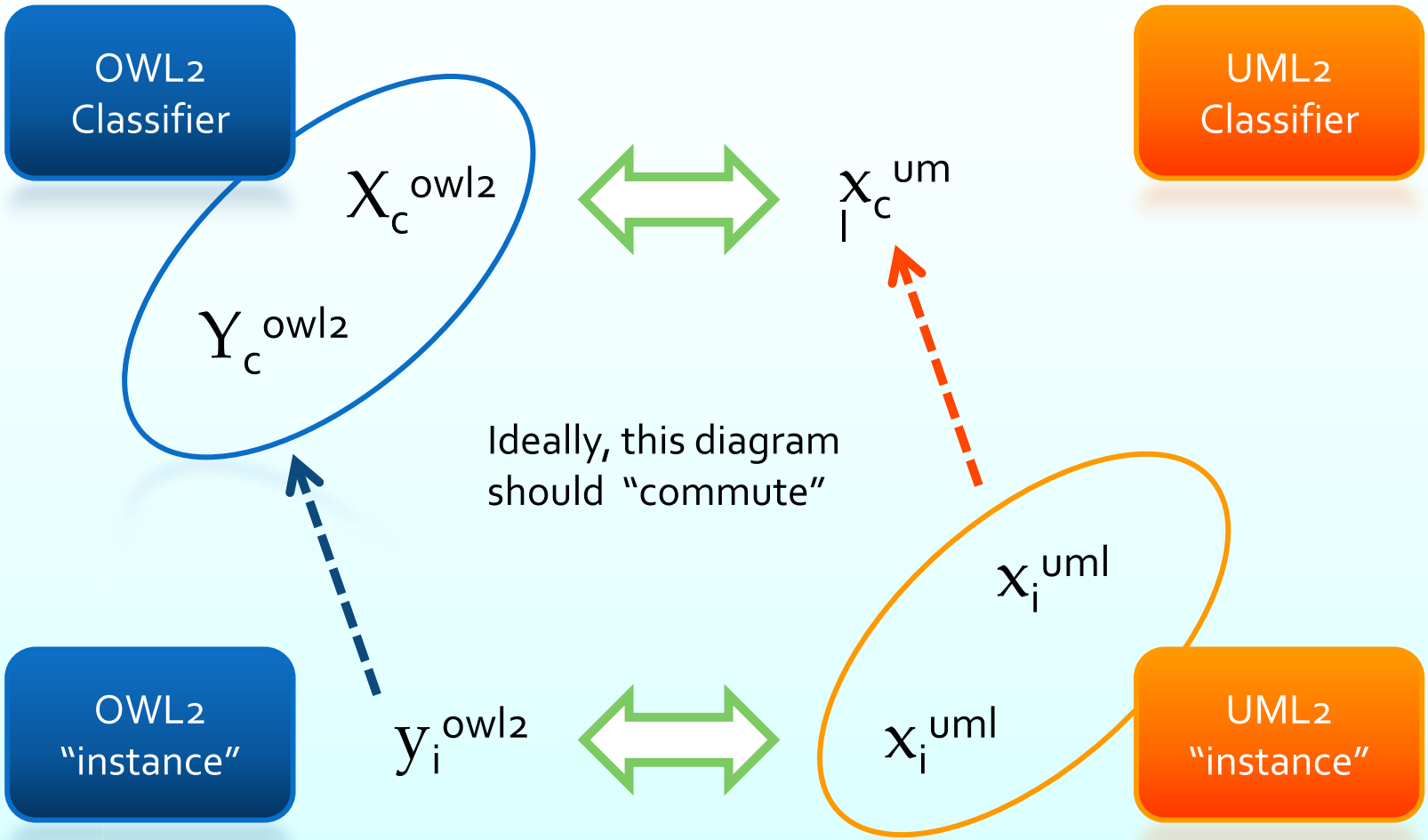
- The UML kernel is the basis for everything in MOF, UML, …
- What is OWL2's equivalent of UML2's kernel?



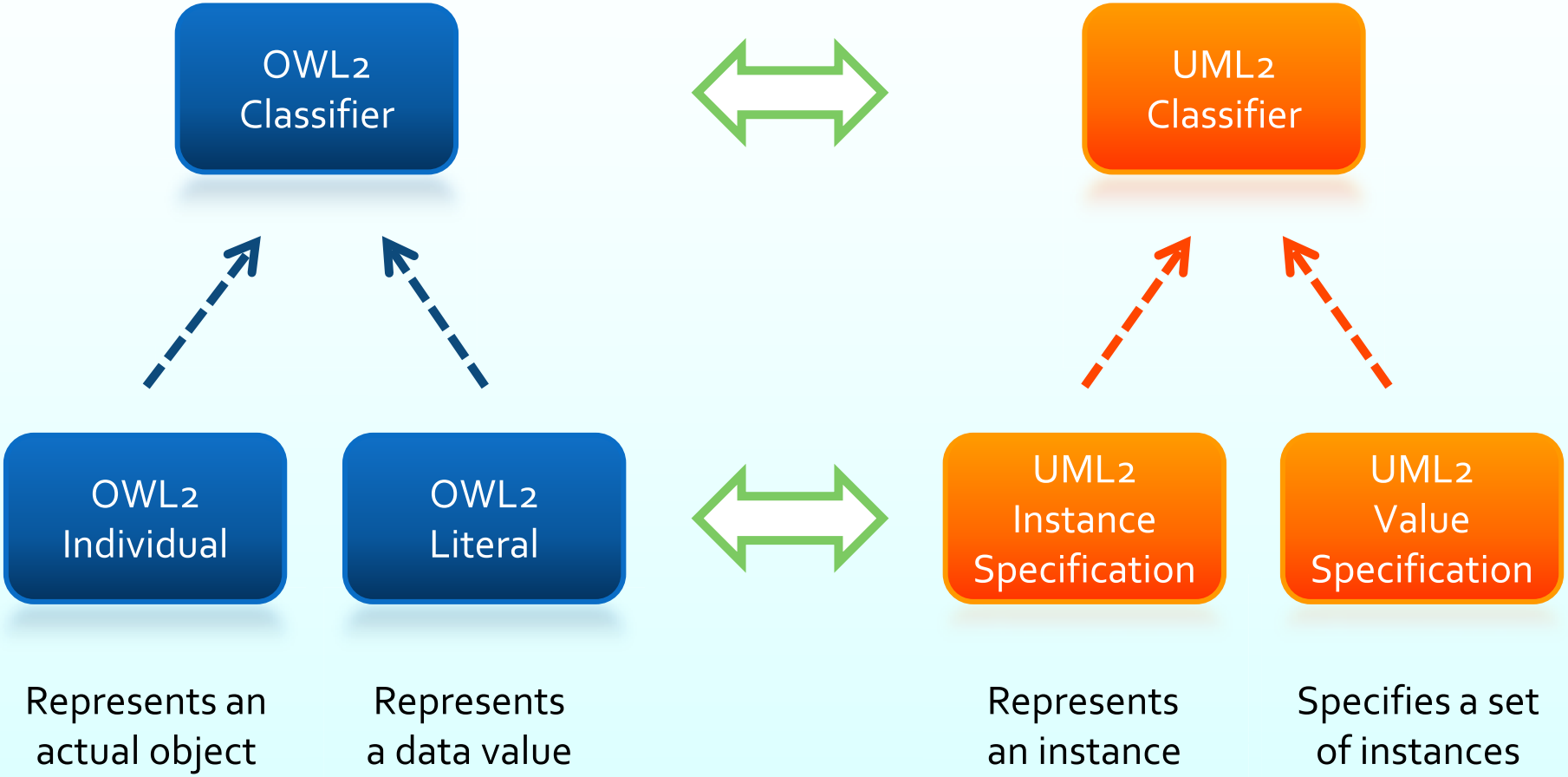
OWL2 Kernel Concepts



OWL2 & UML2 have classifiers…



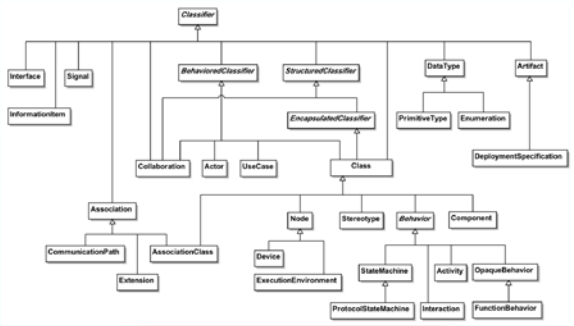
Classification: OWL2 vs. UML



Classification: OWL2 vs. UML

OWL2 Class

OWL2 Data Type



OWL2 Individual

OWL2 Literal

UML2 Instance Specification

UML2 Value Specification

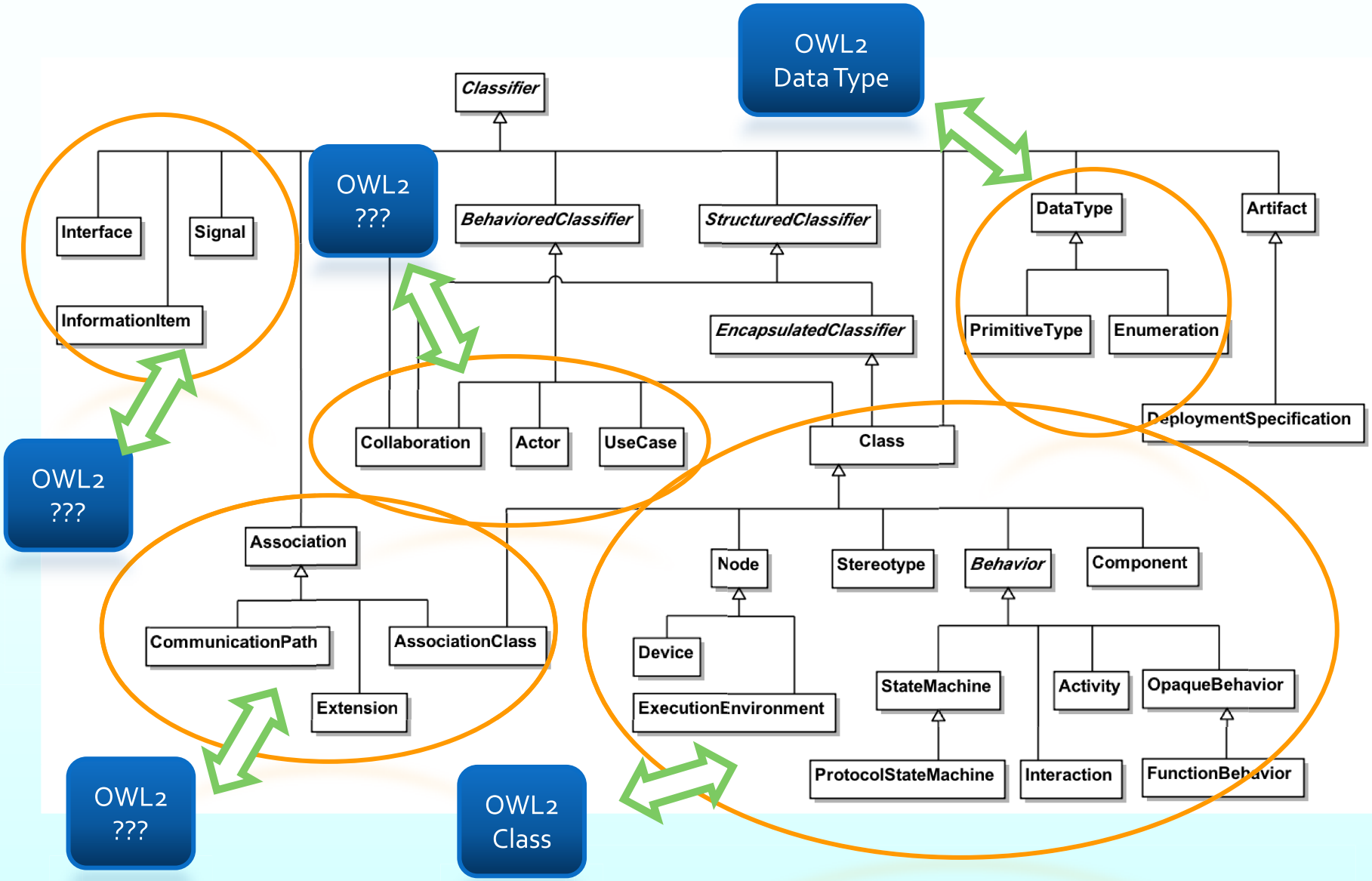
Represents an actual object

Represents a data value

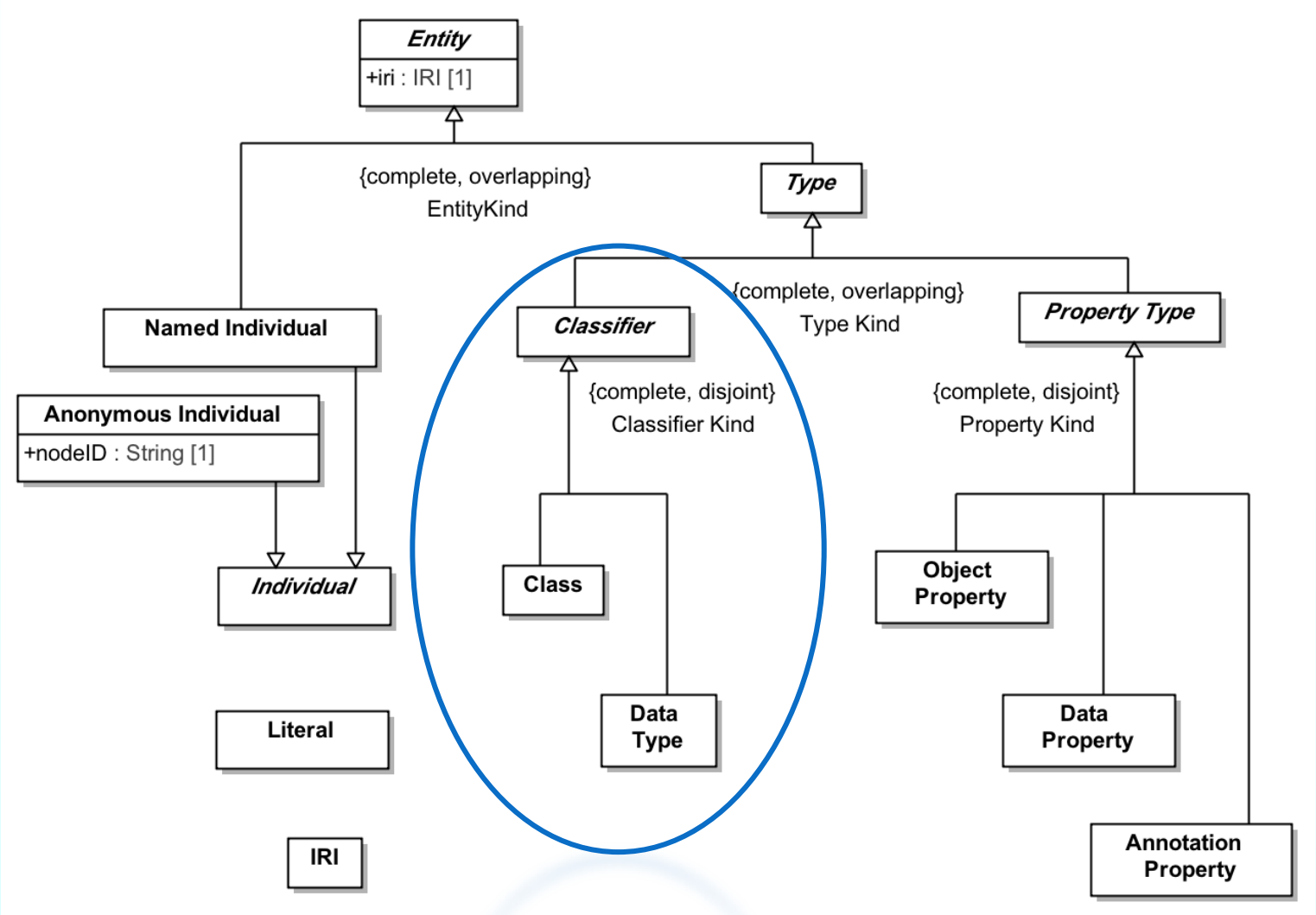
Represents an instance

Specifies a set of instances

Mapping UML2 Classifiers to OWL2



Classifier Mismatch

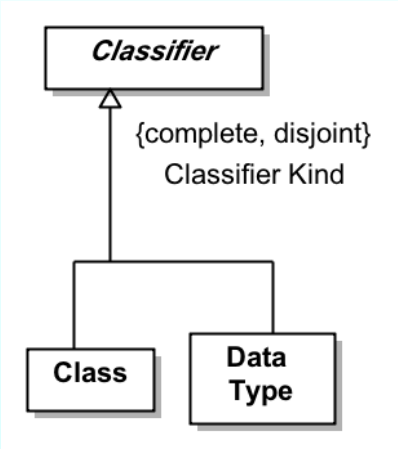


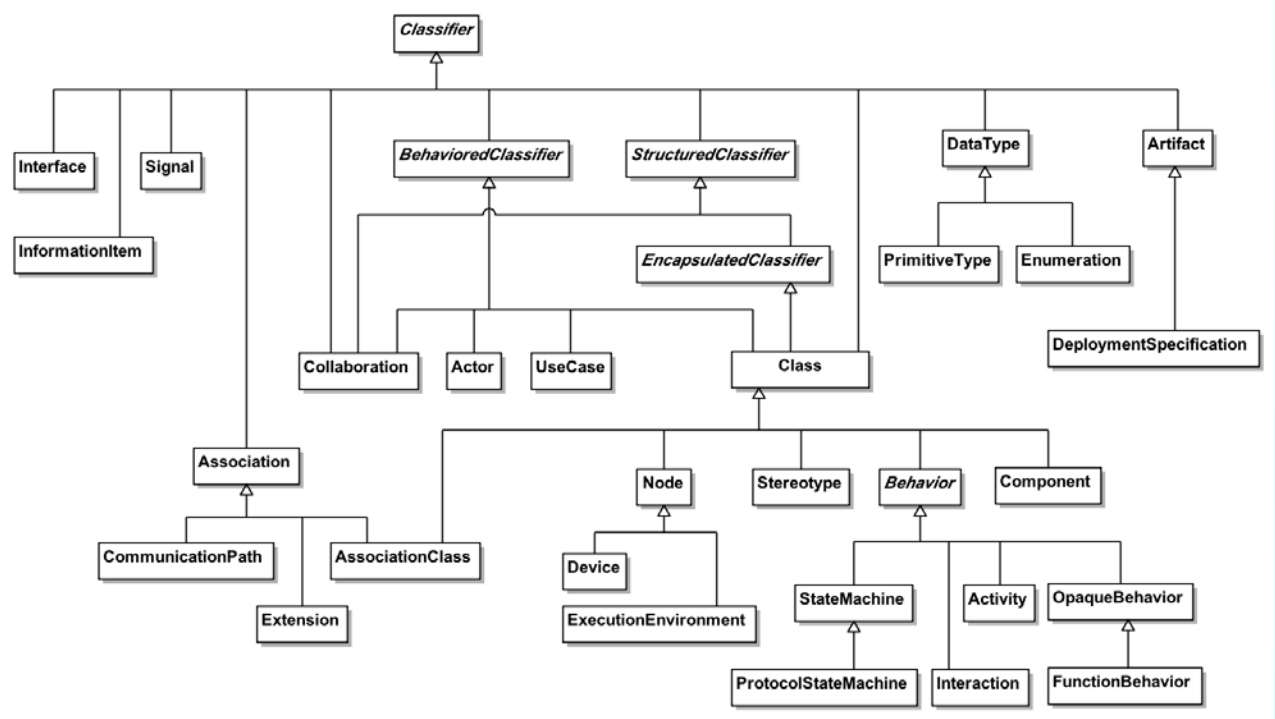
Classifier Mismatch

OWL2
Classifier

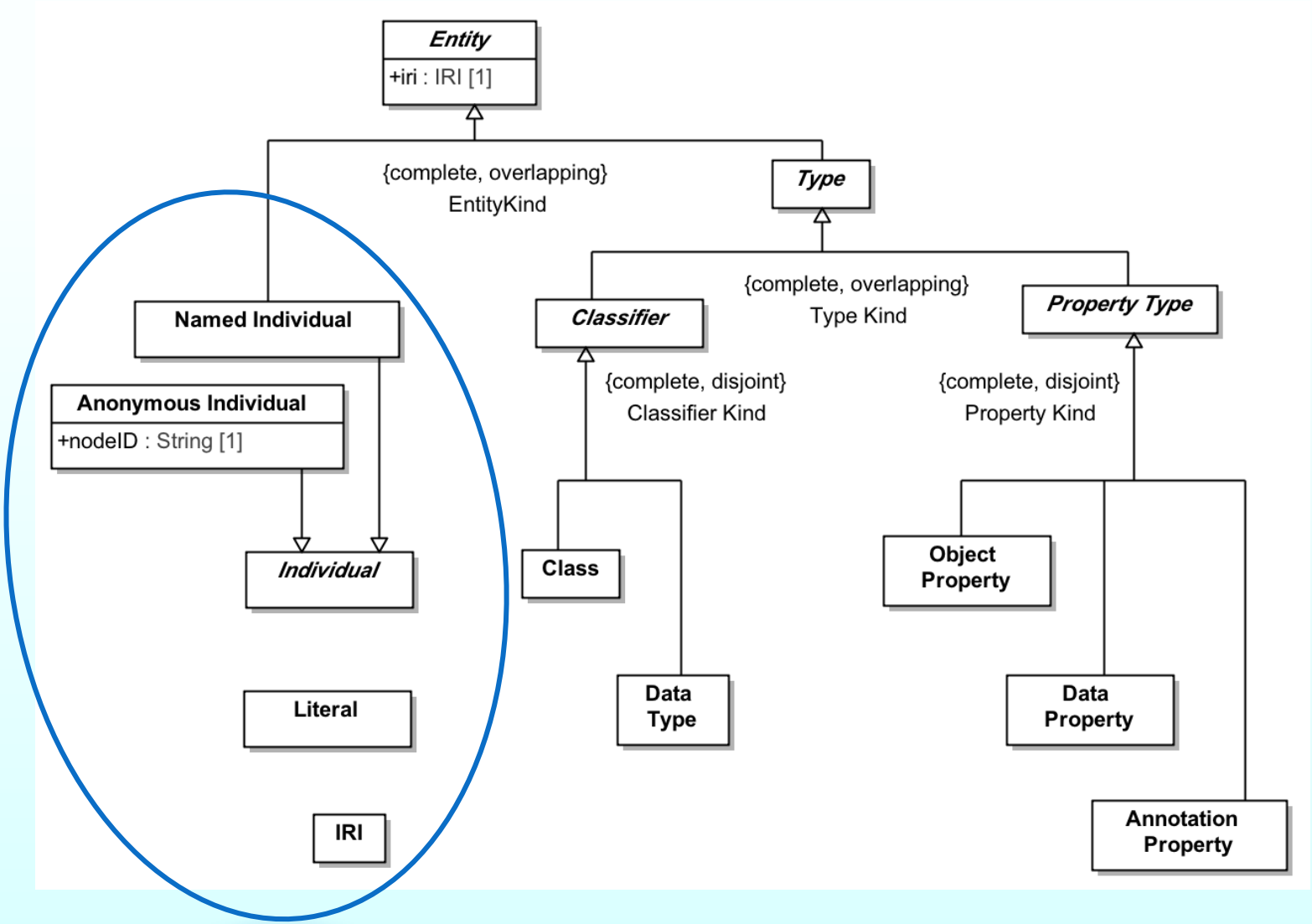
OWL2 & UML2
CLASSIFIERS ARE
HOMONYMS!

UML2
Classifier





Instance Mismatch



Instance Mismatch

OWL2 Individual

Represents an actual object

OWL2 Literal

Represents a data value

OWL2 IRI

Identifies an Entity (named individual or type)

50% OF UML2 INSTANCES ARE SPECIFICATION GHOSTS!

UML2 Instance Specification

UML2 Link (?)

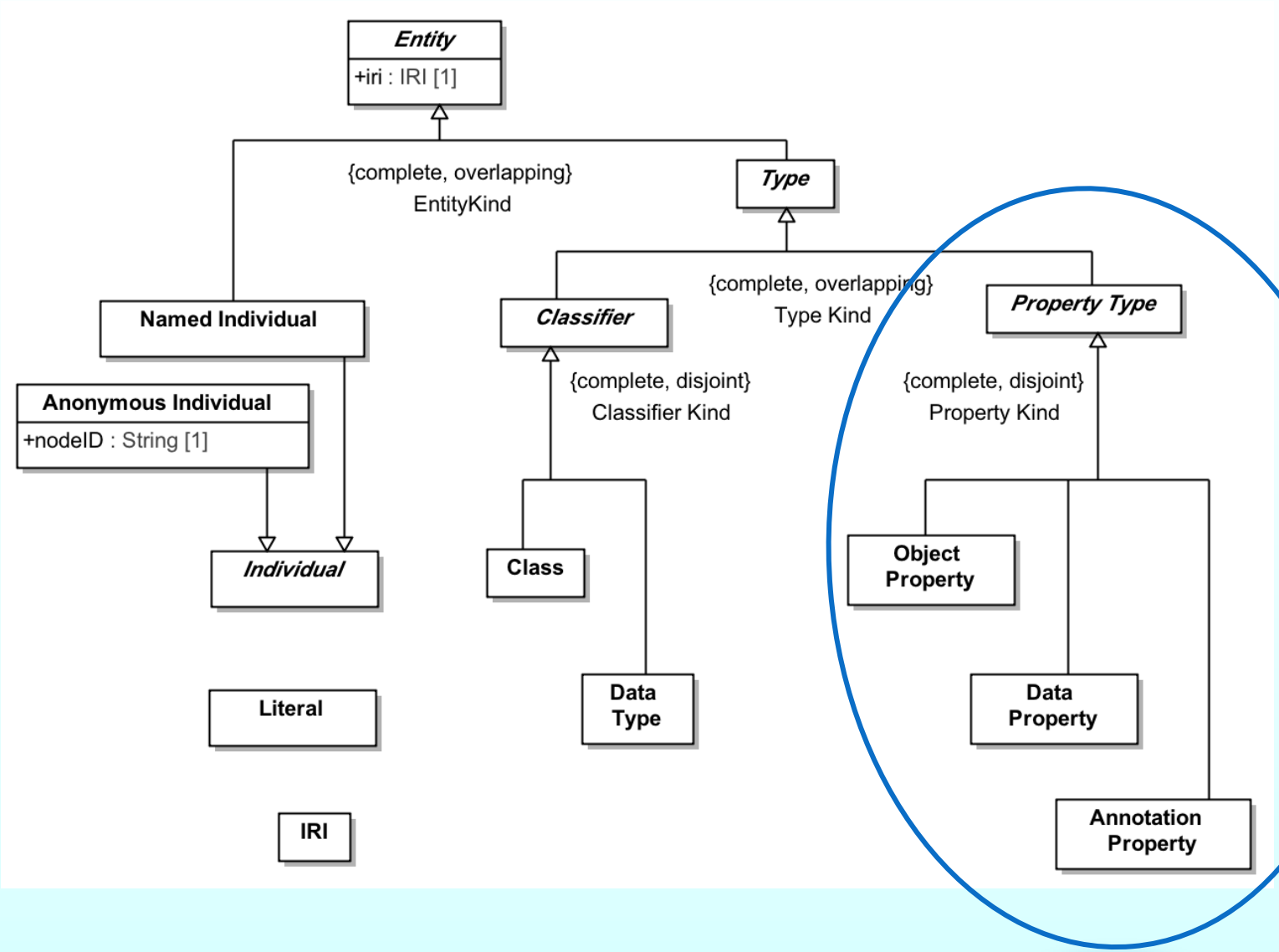
UML2 Value Specification

UML2 Stereotype Instance (?)

An instance specification is a model element that represents an instance in a modeled system.

A value specification is the specification of a (possibly empty) set of instances, including both objects and data values.

Property Mismatch



Property Mismatch

OWL2 Object Property

Connects pairs of individuals

OWL2 Data Property

Connects individuals with literals

OWL2 Annotation Property

Connects individuals with individuals or literals

WITH LINK A SPECIFICATION GHOST, ASSOCIATIONS ARE SEMANTIC VAPOR!

UML2 Property

A property is a structural feature of a classifier; it may represent an attribute, an association end or both

UML2 Association

Specifies a semantic relationship that can occur between typed instances

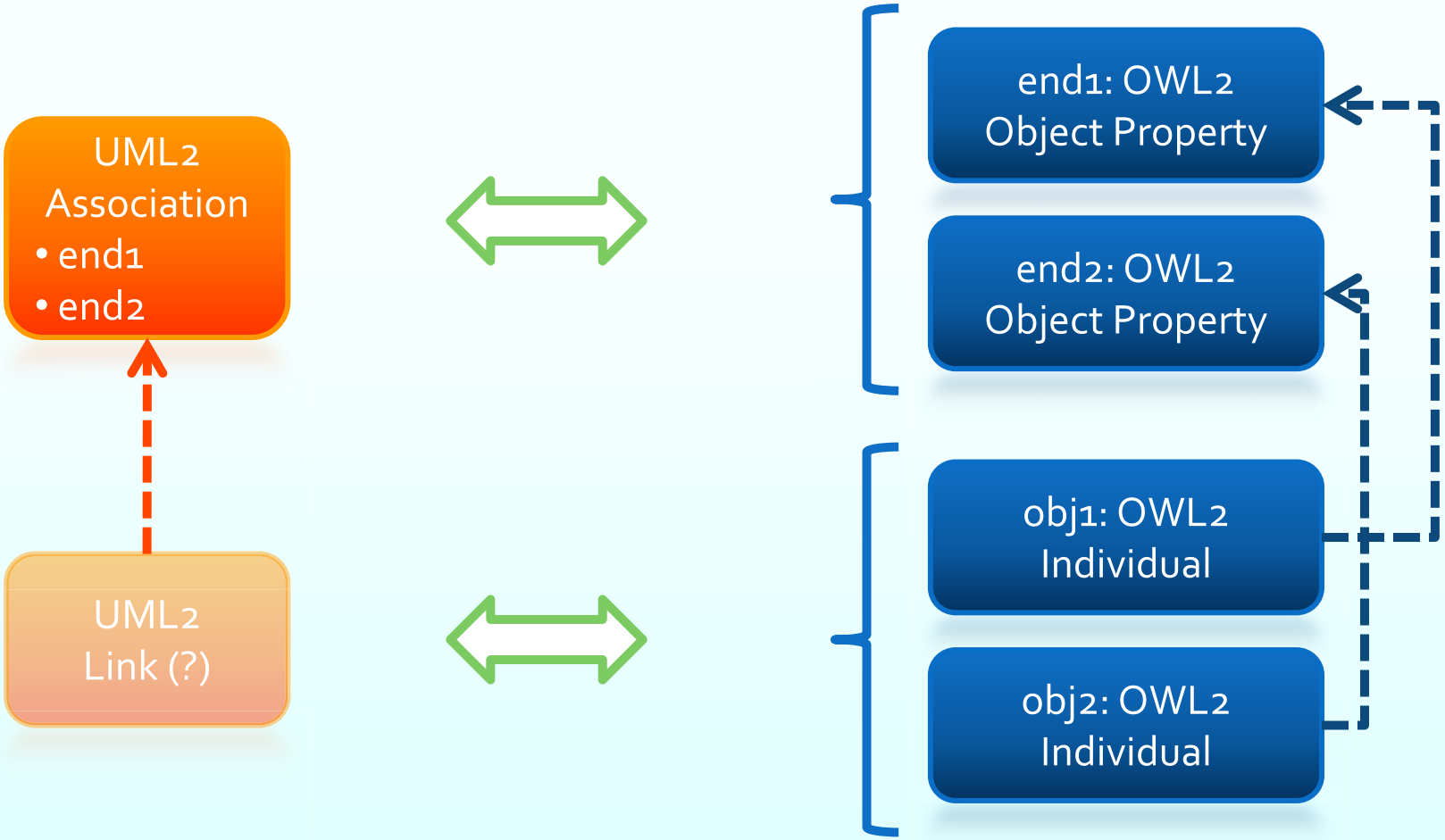
UML2 Link (?)

An instance of an association is a link: a tuple with one value for each end of the association, where each value is an instance of the type of the end

Key Strategy:

Simplify UML/SysML

ODM 1.0



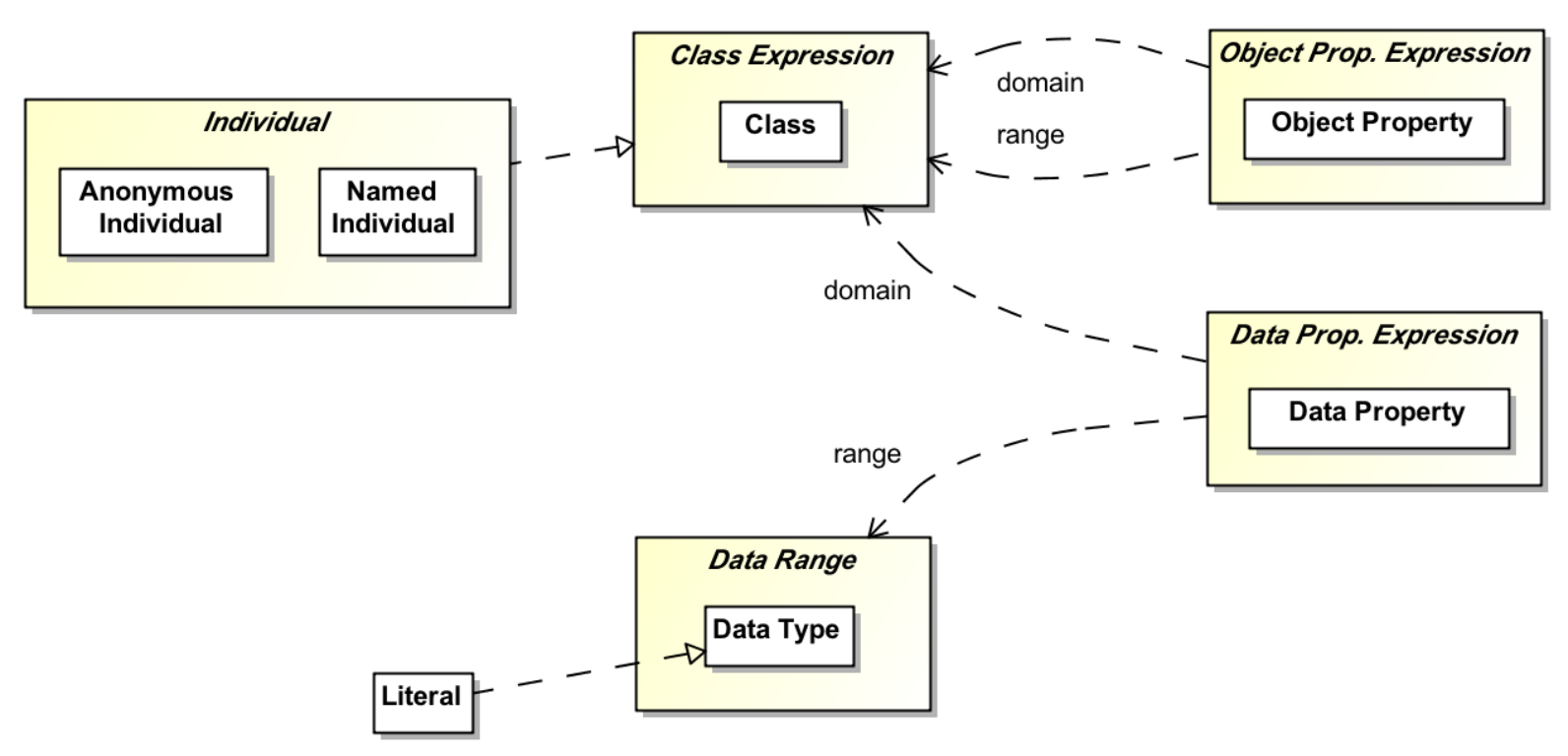
WITH LINK A SPECIFICATION GHOST,
THIS CONSTRUCTION REQUIRES NON-STANDARD TOOL SUPPORT

Key Simplification Insight

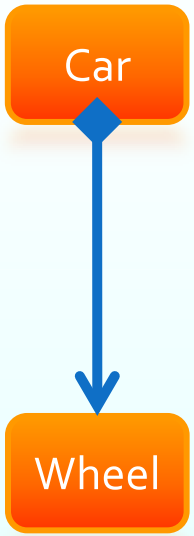
- 1 metaclass is responsible for most of the mismatches

Specifies a semantic relationship that can occur between typed instances

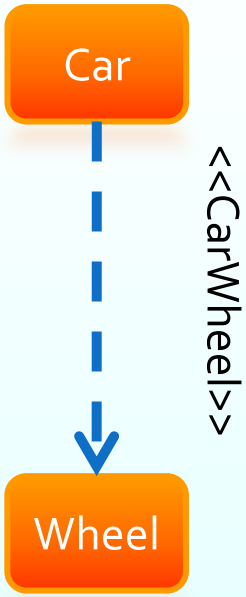
UML2 Association



OWL2 / SysML (Without Associations)

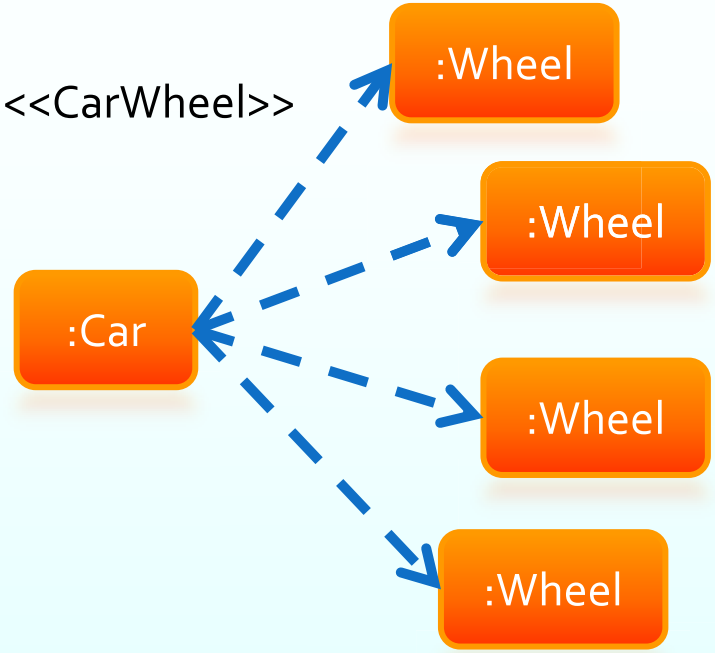


Conventional way
(with associations)



Dependency
Stereotype

The "association" type



Stereotype
Application

The "association" tuple

Association-Less is Easier

- No multiplicities
  - Avoids the problems of trying to specifying a value for a collection property
- No property subsets / redefinitions
  - It's replaced by property (type) specialization
- No instance specifications
  - All "instances" are singleton classes
  - The characteristics of an "instance" is specified via static properties with a read-only default value

How does it work?

Data Type

Literal

Class

Individual

Object Property

Data Property

Annotation Property

Data Type

Literal

Element

Stereotyped Element

Relationship

Property

Comment

ALL VARIATIONS = TYPE SPECIALIZATION

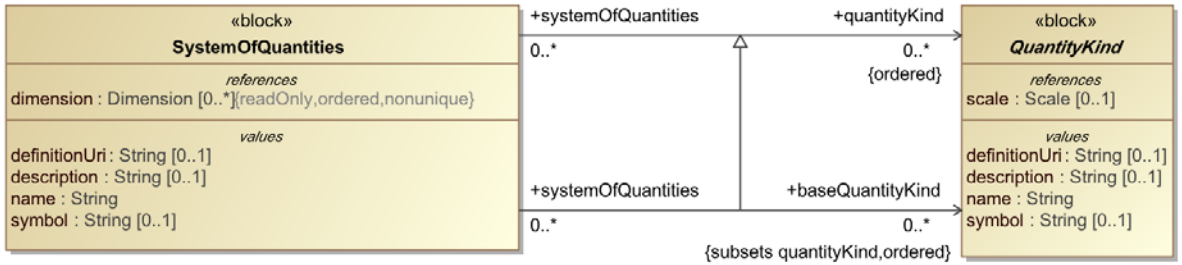
NO ASSOCIATIONS! NO INSTANCES!

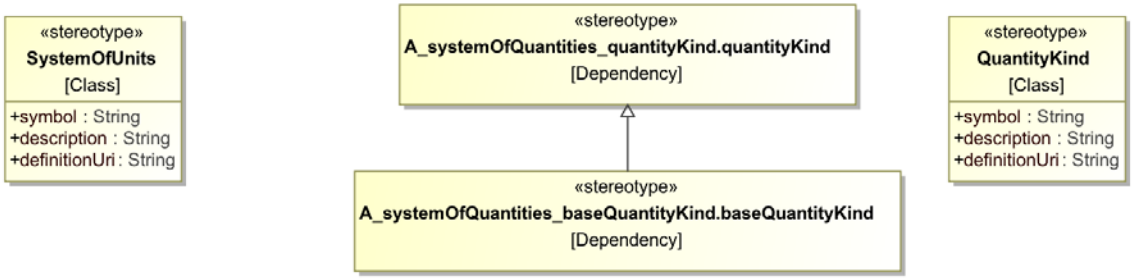
Essential vs. Complete OWL2

- EMOF / EOWL2 modeling level
  - Simplified UML2 (no associations, no instances, ...)
- CMOF / COWL2 modeling level
  - Complicated UML2 (w/ associations, w/ instances ...)
- UML2 / SysML2 + ... => EOWL2 / COWL2
  - In this direction, we reverse engineer a simplified view of UML2, SysML2, and conceptual models (e.g., QUDV)
- EOWL2 => Essential UML Profiles
  - Essential UML Profiles are focused on usability

Example from QUDV

See SysML1.2 Annex C.5





QUDV-cmof ontology contains axioms to make the QUDV complete (i.e,, complicated) w.r.t. the QUDV conceptual model

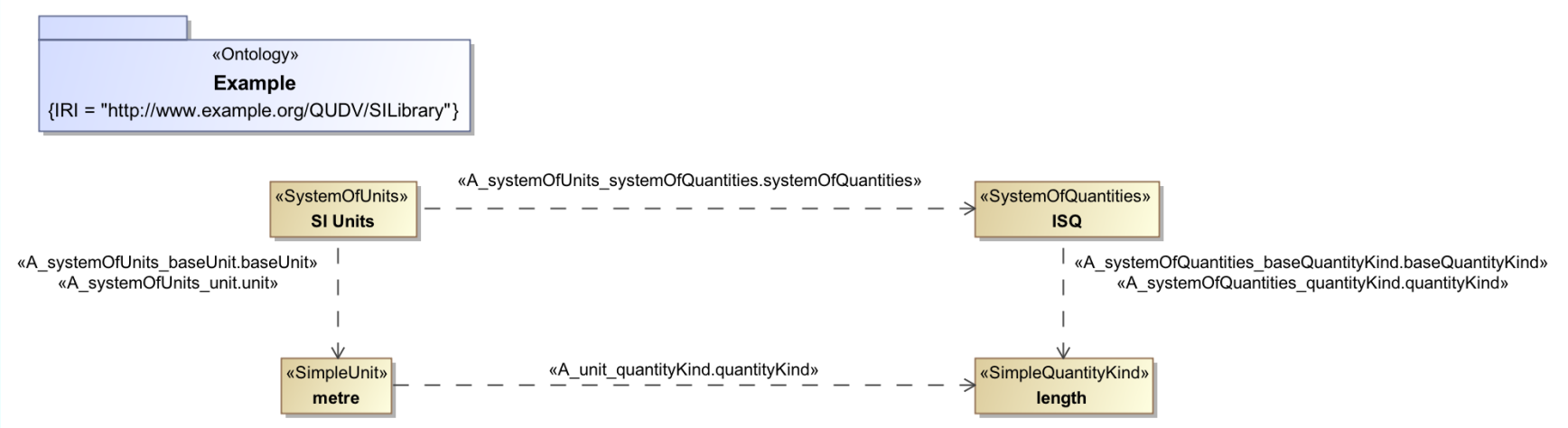
ObjectProperty: QUDV-cmof:A_systemOfUnits_systemOfQuantities.systemOfUnits

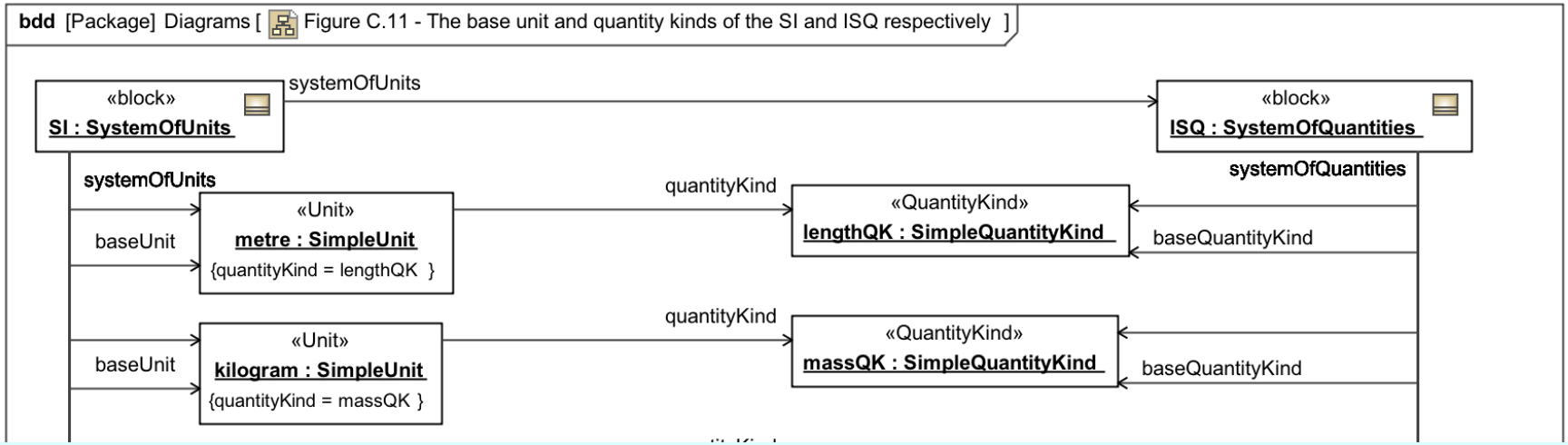
Domain:
<http://eis.jpl.nasa.gov/imce/ontologies/imce.jpl.nasa.gov/QUDV/QUDV#SystemOfQuantities>
Range:
<http://eis.jpl.nasa.gov/imce/ontologies/imce.jpl.nasa.gov/QUDV/QUDV#SystemOfUnits>
InverseOf:
<http://eis.jpl.nasa.gov/imce/ontologies/imce.jpl.nasa.gov/QUDV/QUDV#A_systemOfUnits_systemOfQuantities.systemOfQuantities>

ObjectProperty: QUDV-cmof:A_systemOfQuantities_baseQuantityKind.systemOfQuantities

Domain:
<http://eis.jpl.nasa.gov/imce/ontologies/imce.jpl.nasa.gov/QUDV/QUDV#QuantityKind>
Range:
<http://eis.jpl.nasa.gov/imce/ontologies/imce.jpl.nasa.gov/QUDV/QUDV#SystemOfQuantities>
InverseOf:
<http://eis.jpl.nasa.gov/imce/ontologies/imce.jpl.nasa.gov/QUDV/QUDV#A_systemOfQuantities_baseQuantityKind.baseQuantityKind>

Essential UML Profile example (QUDV SI Library)





Why?

- We can formalize UML2- - - - in OWL2 fairly easily
- We can convert between UML2 & OWL2 easily
- We can use OWL2 for our repository
  - XMI is essential for **normative, OMG-grade** artifacts
- QVT's traces for mapping OWL2 ⇔ SysML gives us the correspondences we need to generate tool-specific OWL2 import/export & much more!
- Anything we can formalize in OWL2 we can turn into a user-friendly profile extending SysML
- It works (regular mappings are simpler to implement)
- It's a pragmatic compromise between:
  - Supporting JPL projects with current OMG specifications & tools
  - A step towards a semantic formalism for reasoning about SysML, not UML2
    (UML2 is quite large ~ 275 metaclasses; SysML1.2 profile is smaller ~ 50 stereotypes)

Summary of OWL2 + SysML: SysML Modeling Becomes Easier

With current UML2.4 + SysML1.2:

- Association ambiguity
  - SysML: association-owned end is irrelevant but it is relevant to UML underneath
- Limited Support for Instances
  - Instance specifications are "flat" – no hierarchy
  - No concept of an instance as distinct from a commitment to instantiating a metaclass
  - An instance specification blurs the distinction between instance & its state of affairs

With an OWL2 foundation underneath:

- Dependencies as associations
  - Derived properties & tool customizations gives users the appearance of associations
- Instance = stereotyped-singleton
  - Singleton elements can be arbitrarily nested
  - Stereotype application is an optional aspect of modeling an instance as singleton element
  - An UML::InstanceSpecification is ideally suited for modeling a state of affairs of a singleton instance