Integrated Model Centric Engineering

OML Platform

CHARLES ELACHI MISSION CONTROL CENTER
MISSION CONTROL
DATA CONTROLLER
DEEP SPACE NETWORK
The Center of the Universe
MADRID 65

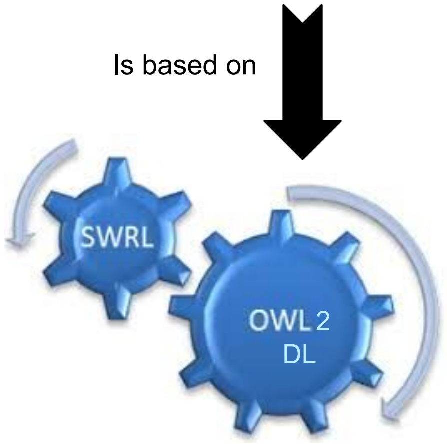
Maged Elaasar
Software Architect

Jet Propulsion Laboratory
California Institute of Technology

Ontological Modeling Language

- OML is a language for
  - defining semantic vocabularies for interrelated domains
  - bundling them in a methodological way to enable
  - describing and analyzing complex systems

- OML is inspired by the W3C semantic web standards: OWL2-DL & SWRL
  - OML abstract syntax maps to patterns expressed in a subset of those standards
  - OML concrete syntaxes include a textual grammar and a graphical notation
  - OML semantics is based on Description Logic (DL)

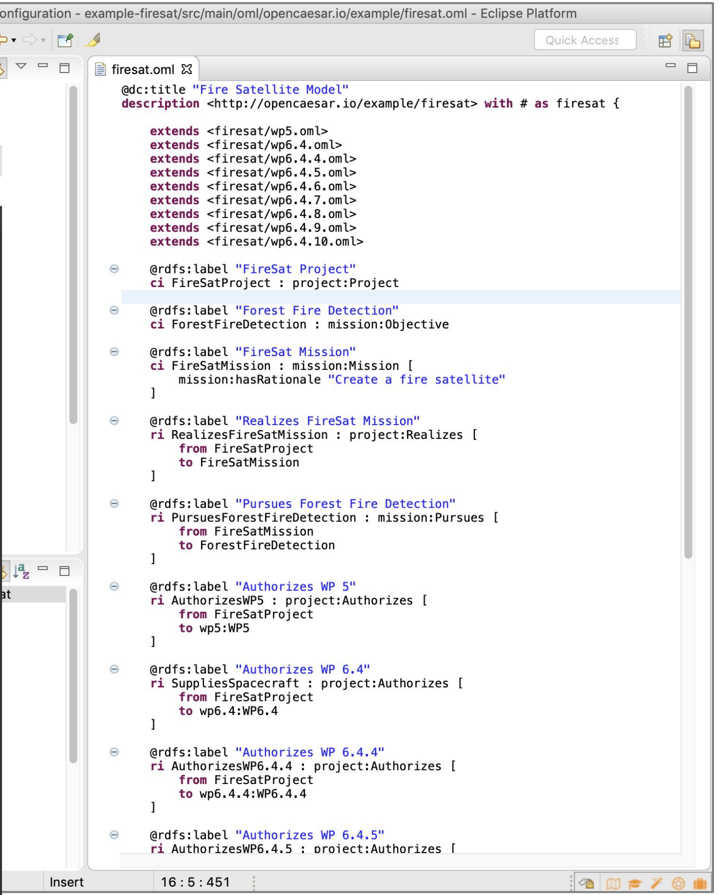


Implementation

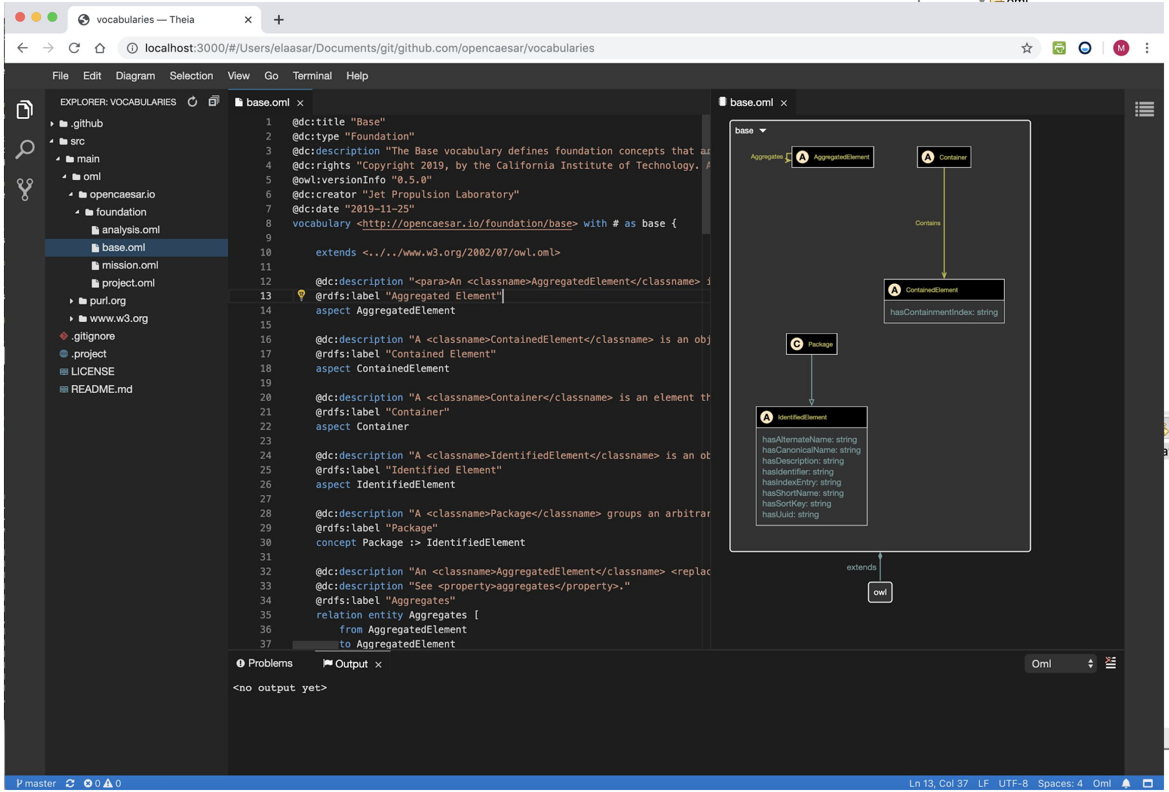
- **Abstract syntax** (metamodel) developed using Ecore
- **Java APIs** developed using Ecore and Xtend
- **Textual syntax** (grammar) developed using Xtext
- **Graphical syntax** (notation) developed (*partially*) using Sprotty
- **OML libraries** are packaged as Maven artifacts, p2 update site, & LSP server
- **OML Workbenches** include: desktop (Eclipse, VS Code, Theia), web (Theia)
- **OML interchange formats** include: textual grammar, XMI and a Zip
- **OML repositories** include GitHub or any other file/object repositories

OML Workbenches

Eclipse Workbench: Rosetta (desktop)


runtime-New_configuration - example-firesat/src/main/oml/opencaesar.io/example/firesat.oml - Eclipse Platform
Quick Access
Project Explorer
example-firesat
src
main
oml
firesat.oml
Insert
16 : 5 : 451

```
@dc:title "Fire Satellite Model"
description <http://opencaesar.io/example/firesat> with # as firesat {

    extends <firesat/wp5.oml>
    extends <firesat/wp6.4.oml>
    extends <firesat/wp6.4.4.oml>
    extends <firesat/wp6.4.5.oml>
    extends <firesat/wp6.4.6.oml>
    extends <firesat/wp6.4.7.oml>
    extends <firesat/wp6.4.8.oml>
    extends <firesat/wp6.4.9.oml>
    extends <firesat/wp6.4.10.oml>

    @rdfs:label "FireSat Project"
    ci FireSatProject : project:Project

    @rdfs:label "Forest Fire Detection"
    ci ForestFireDetection : mission:Objective

    @rdfs:label "FireSat Mission"
    ci FireSatMission : mission:Mission [
        mission:hasRationale "Create a fire satellite"
    ]

    @rdfs:label "Realizes FireSat Mission"
    ri RealizesFireSatMission : project:Realizes [
        from FireSatProject
        to FireSatMission
    ]

    @rdfs:label "Pursues Forest Fire Detection"
    ri PursuesForestFireDetection : mission:Pursues [
        from FireSatMission
        to ForestFireDetection
    ]

    @rdfs:label "Authorizes WP 5"
    ri AuthorizesWP5 : project:Authorizes [
        from FireSatProject
        to wp5:WP5
    ]

    @rdfs:label "Authorizes WP 6.4"
    ri SuppliesSpacecraft : project:Authorizes [
        from FireSatProject
        to wp6.4:WP6.4
    ]

    @rdfs:label "Authorizes WP 6.4.4"
    ri AuthorizesWP6.4.4 : project:Authorizes [
        from FireSatProject
        to wp6.4.4:WP6.4.4
    ]

    @rdfs:label "Authorizes WP 6.4.5"
    ri AuthorizesWP6.4.5 : project:Authorizes [
```



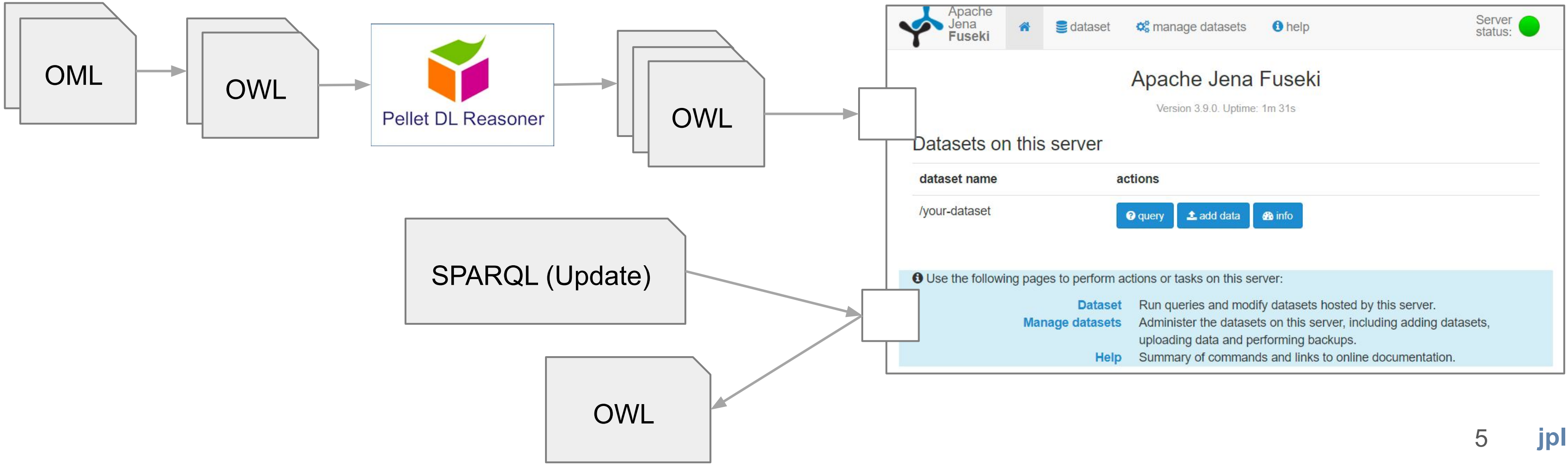
```
@dc:title "Base"
@dc:type "Foundation"
@dc:description "The Base vocabulary defines foundation concepts that ar
@dc:rights "Copyright 2019, by the California Institute of Technology. A
@owl:versionInfo "0.5.0"
@dc:creator "Jet Propulsion Laboratory"
@dc:date "2019-11-25"
vocabulary <http://opencaesar.io/foundation/base> with # as base {

    extends <../../www.w3.org/2002/07/owl.oml>

    @dc:description "<para>An <classname>AggregatedElement</classname> i
    @rdfs:label "Aggregated Element"
    aspect AggregatedElement

    @dc:description "A <classname>ContainedElement</classname> is an obj
    @rdfs:label "Contained Element"
    aspect ContainedElement

    @dc:description "A <classname>Container</classname> is an element th
    @rdfs:label "Container"
    aspect Container

    @dc:description "A <classname>IdentifiedElement</classname> is an ob
    @rdfs:label "Identified Element"
    aspect IdentifiedElement

    @dc:description "A <classname>Package</classname> groups an arbitrar
    @rdfs:label "Package"
    concept Package :> IdentifiedElement

    @dc:description "An <classname>AggregatedElement</classname> <replac
    @dc:description "See <property>aggregates</property>."
    @rdfs:label "Aggregates"
    relation entity Aggregates [
        from AggregatedElement
        to AggregatedElement
```

```
<no output yet>
```

VSCode Workbench: Luxor (desktop and web)

Analysis with Semantic Web Tools

- OML allows analysis with semantic web tools
  - OML supports reasoning (satisfiability analysis, consistency checking) with OWL2-DL reasoners
  - OML supports queries with SPARQL and transformations with SPARQL Update
  - OML supports uploading models to a (triple store) database and querying them using RESTful endpoints
  - OML allows simplification of queries thanks to built-in DL inference semantics and custom-defined inference rules



Apache Jena Fuseki

Version 3.9.0. Uptime: 1m 31s

Datasets on this server

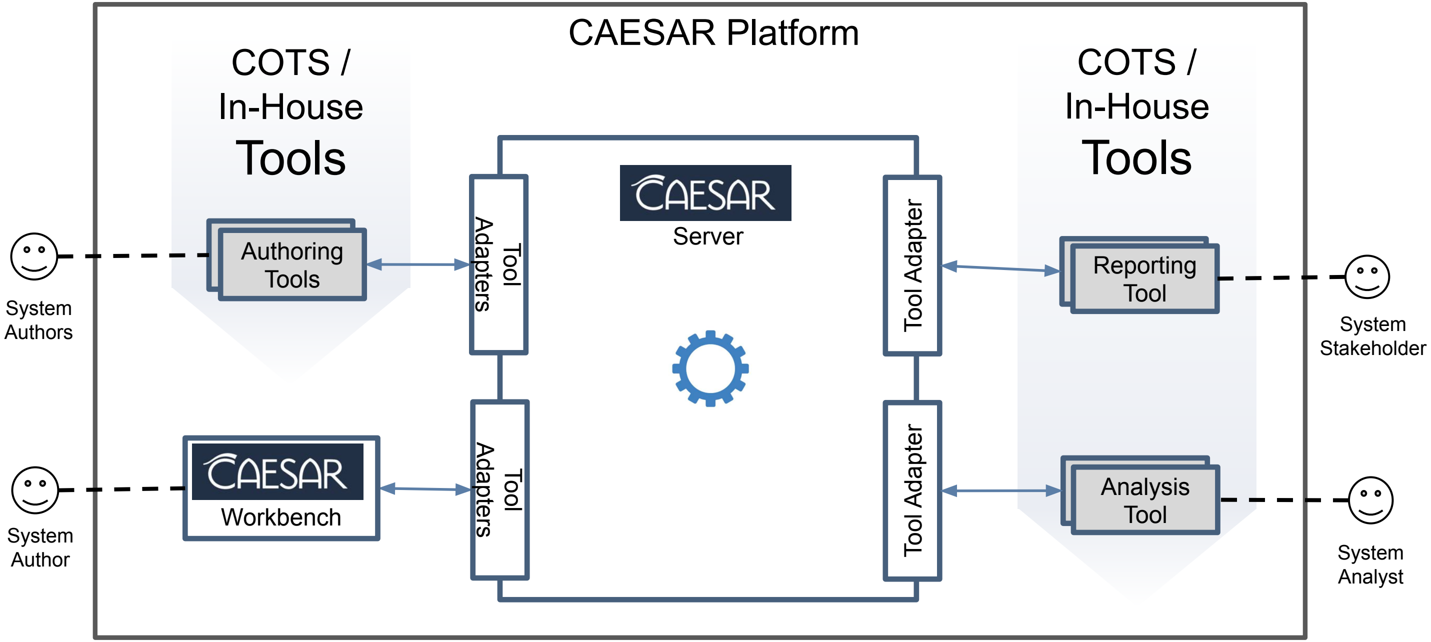
dataset name	actions
/your-dataset	query add data info

Use the following pages to perform actions or tasks on this server:

Dataset Run queries and modify datasets hosted by this server.
Manage datasets Administer the datasets on this server, including adding datasets, uploading data and performing backups.
Help Summary of commands and links to online documentation.

Use Case 1

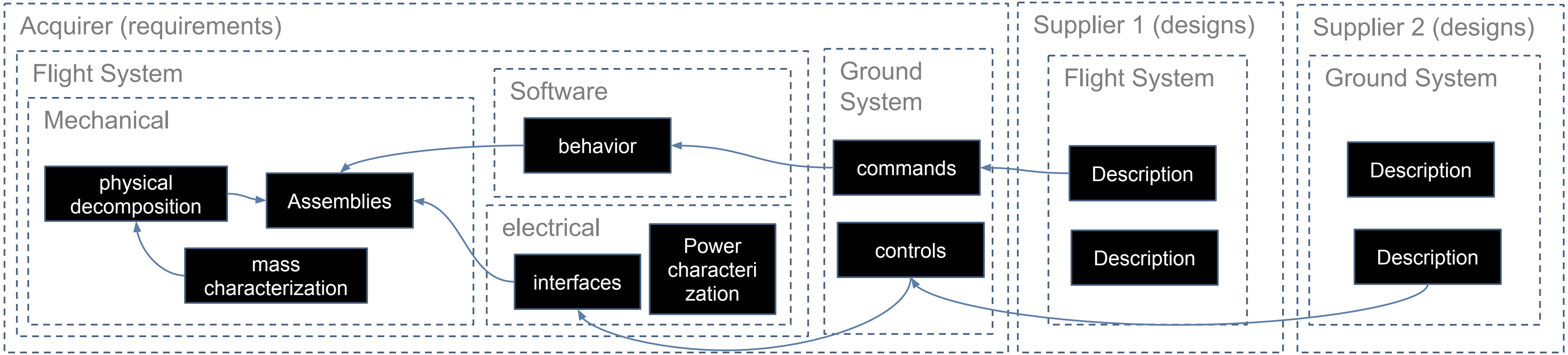
- The first use case for OML is to use it as an tool-neutral methodology-based integration formalism for systems models that are described in disparate repositories, formalisms and tools



Author → Configure / Integrate / Deliver → Analyze / Report

Use Case 2

- The second use case for OML is to use it as a primary system modeling language
  - Allows modeling both vocabularies and user models with the same language
  - Allows modeling with open-world semantics or closed-world semantics
  - Allows componentized, extensible and federated description of vocabularies and user models



OML Specification

- The OML specification is available on:
- https://opencaesar.github.io/oml/

- Covers only abstract syntax and textual syntax (other sections are forthcoming)

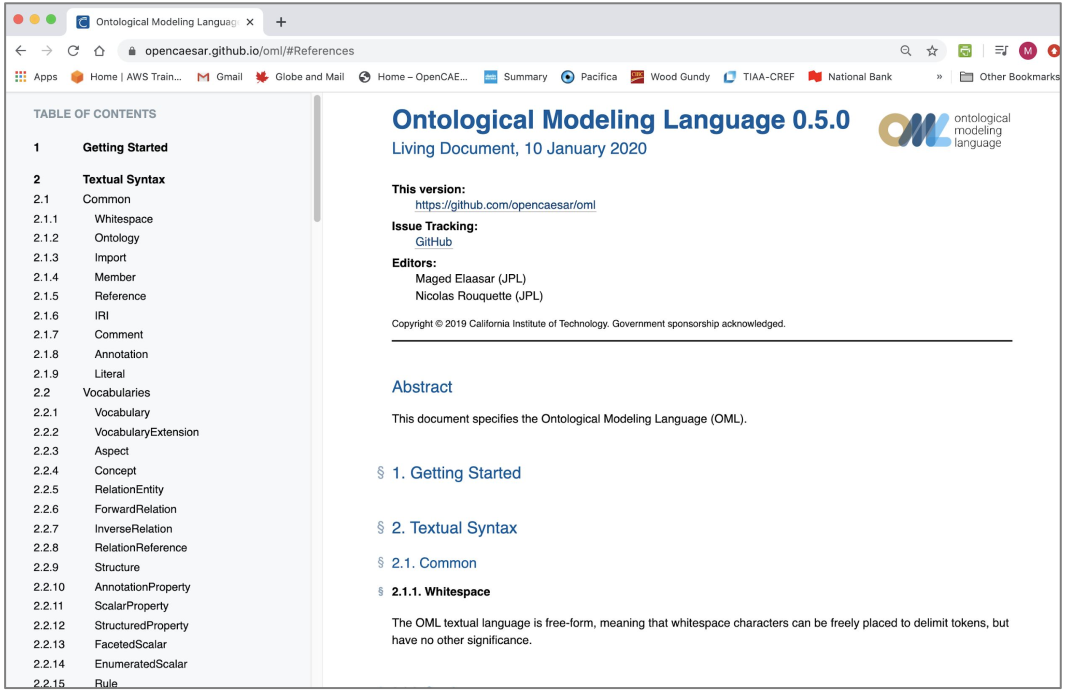


TABLE OF CONTENTS

1 **Getting Started**

2 **Textual Syntax**

2.1 Common

2.1.1 Whitespace

2.1.2 Ontology

2.1.3 Import

2.1.4 Member

2.1.5 Reference

2.1.6 IRI

2.1.7 Comment

2.1.8 Annotation

2.1.9 Literal

2.2 Vocabularies

2.2.1 Vocabulary

2.2.2 VocabularyExtension

2.2.3 Aspect

2.2.4 Concept

2.2.5 RelationEntity

2.2.6 ForwardRelation

2.2.7 InverseRelation

2.2.8 RelationReference

2.2.9 Structure

2.2.10 AnnotationProperty

2.2.11 ScalarProperty

2.2.12 StructuredProperty

2.2.13 FacetedScalar

2.2.14 EnumeratedScalar

2.2.15 Rule

Ontological Modeling Language 0.5.0

Living Document, 10 January 2020

This version:
https://github.com/opencaesar/oml

Issue Tracking:
GitHub

Editors:
Maged Elaasar (JPL)
Nicolas Rouquette (JPL)

Abstract

This document specifies the Ontological Modeling Language (OML).

§ 1. Getting Started

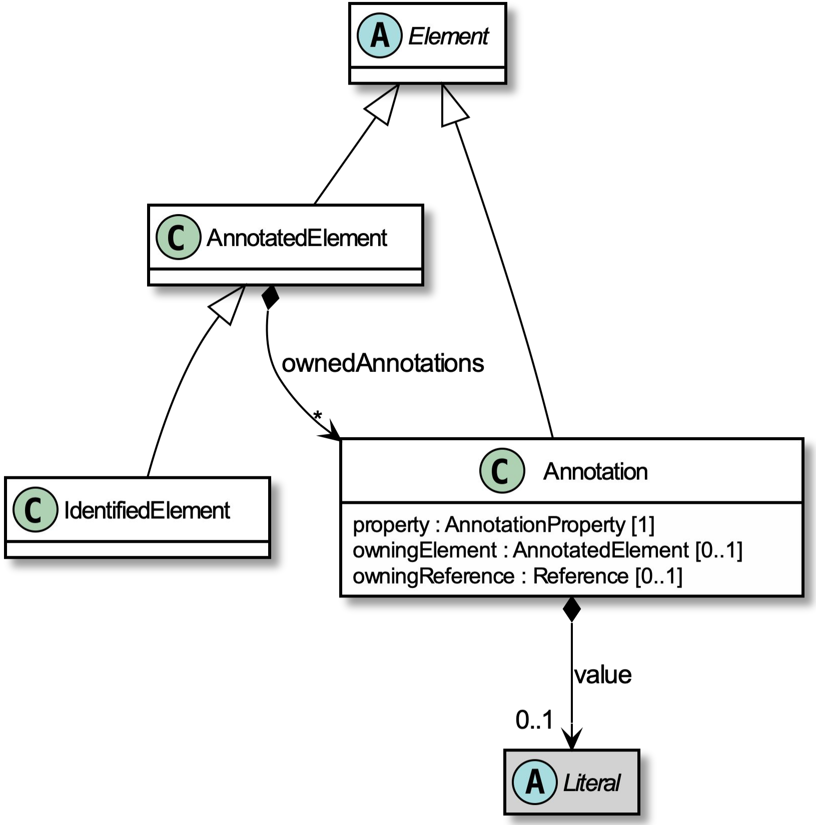
§ 2. Textual Syntax

§ 2.1. Common

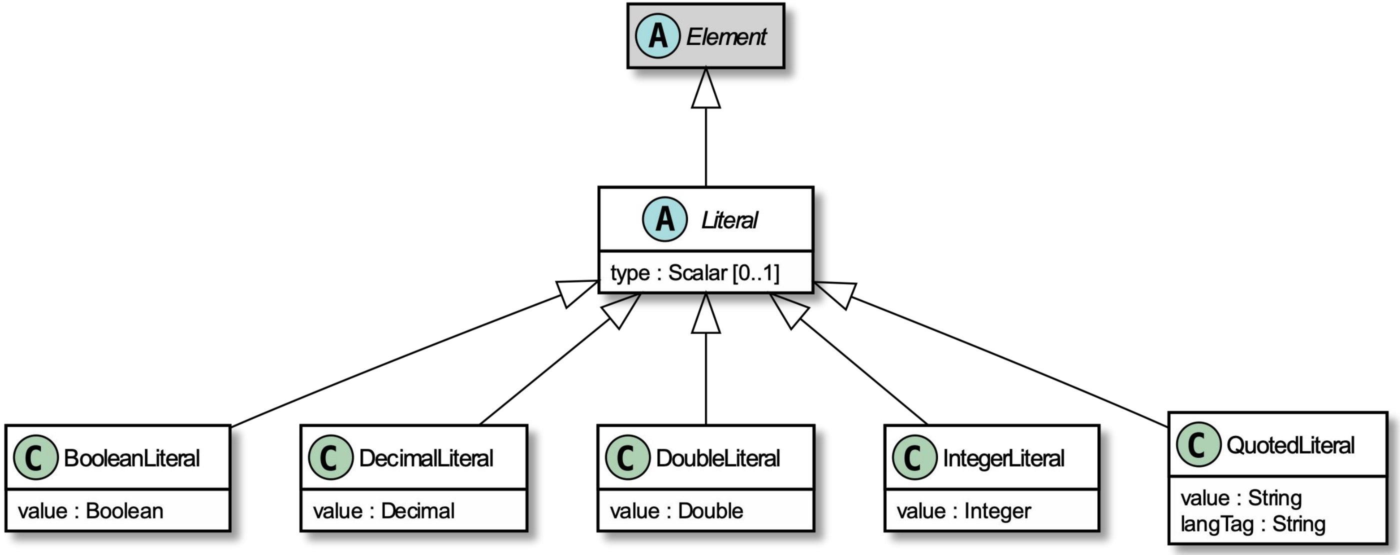
§ 2.1.1. Whitespace

The OML textual language is free-form, meaning that whitespace characters can be freely placed to delimit tokens, but have no other significance.

Elements

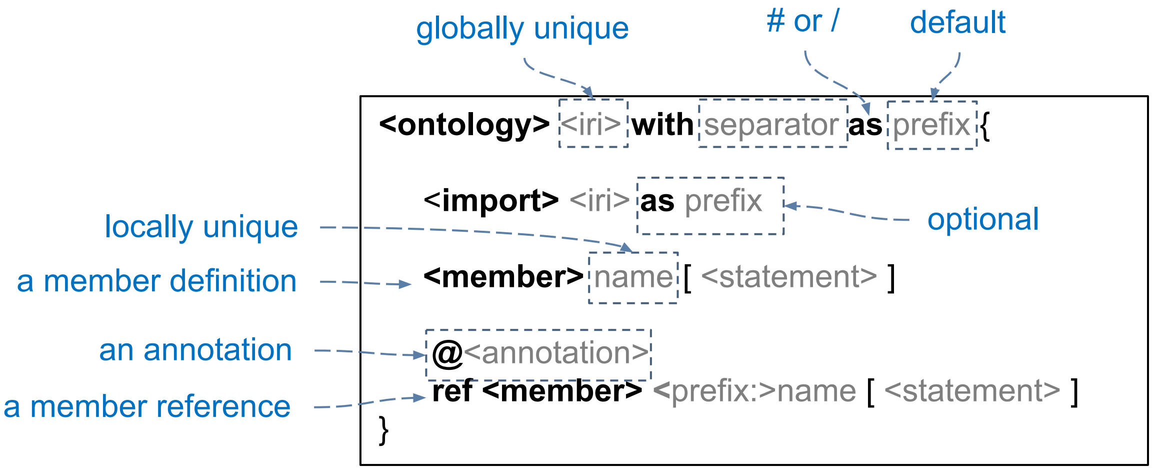

A
Element
C
AnnotatedElement
ownedAnnotations
*
C
IdentifiedElement
C
Annotation
property : AnnotationProperty [1]
owningElement : AnnotatedElement [0..1]
owningReference : Reference [0..1]
value
0..1
A
Literal

Literals


A Element
A Literal
type : Scalar [0..1]
C BooleanLiteral
value : Boolean
C DecimalLiteral
value : Decimal
C DoubleLiteral
value : Double
C IntegerLiteral
value : Integer
C QuotedLiteral
value : String
langTag : String

Ontologies

- An ontology has a namespace IRI, separator, and prefix
- An ontology can import (the content of) other ontologies
- An ontology can have statements about its own or imported members
- An ontology, its members, and its statements can have non-semantic annotations



```
<ontology> <iri> with separator as prefix {

    <import> <iri> as prefix

    <member> name [ <statement> ]

    @<annotation>
    ref <member> <prefix:>name [ <statement> ]
}
```

globally unique

or /

default

optional

locally unique

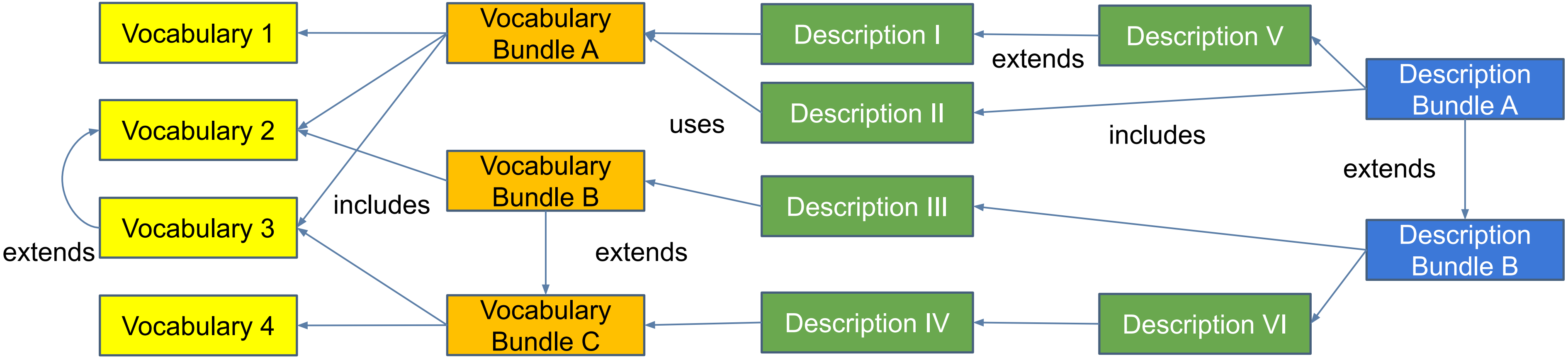
a member definition

an annotation

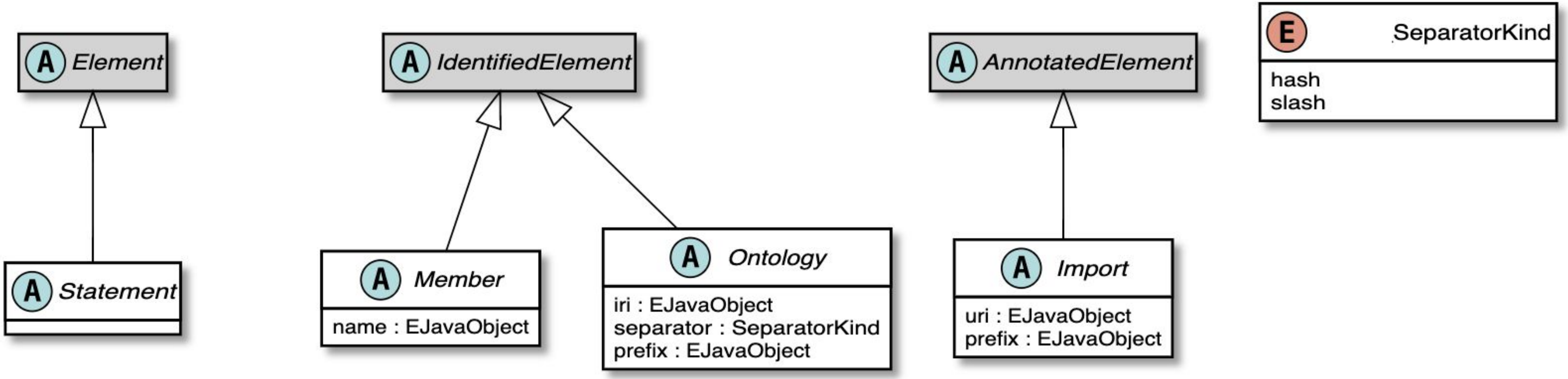
a member reference

Four Kinds of Ontologies

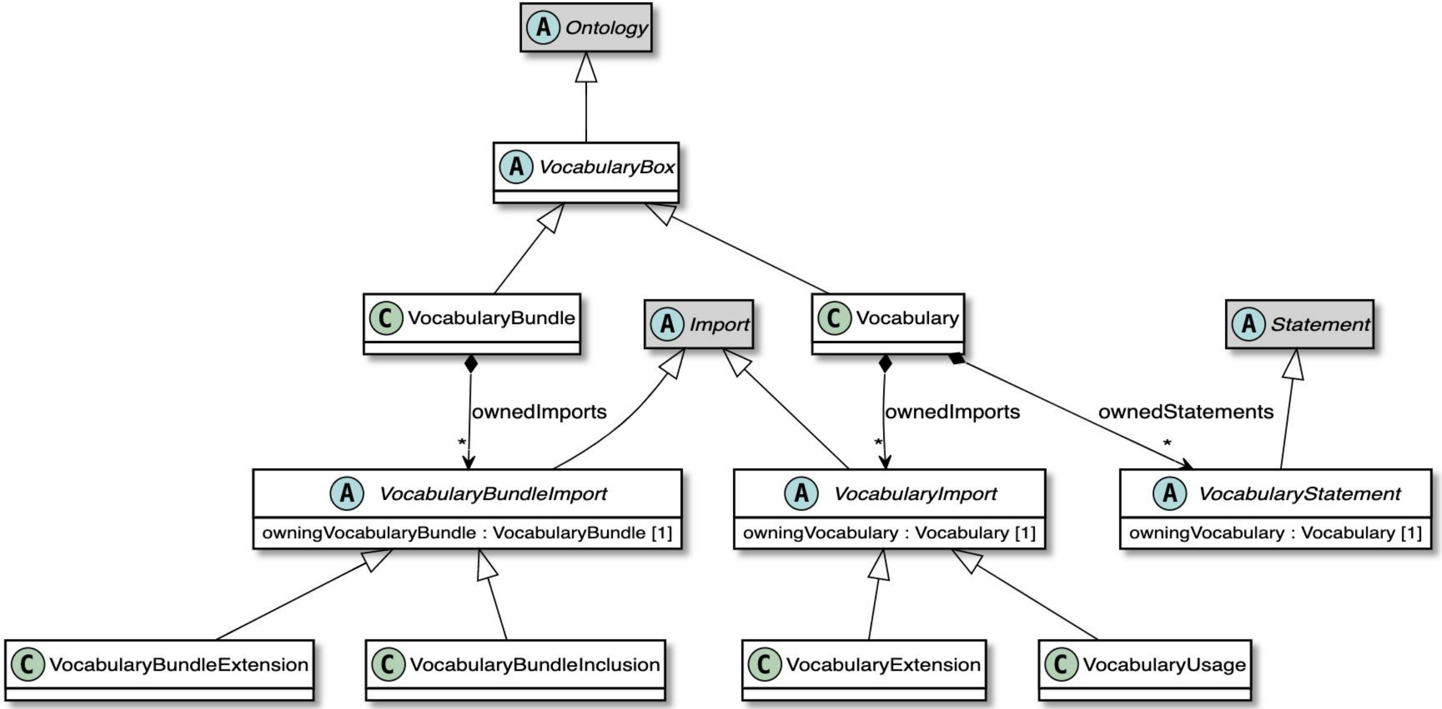
- **Vocabularies:** ontological terms with open-world semantics
- **Vocabulary Bundles:** aggregations of vocabularies with closed-world semantics
- **Descriptions:** systems described using ontological vocabularies / bundles
- **Description Bundles:** aggregations of descriptions to reason on together



Ontologies and Imports



Vocabularies

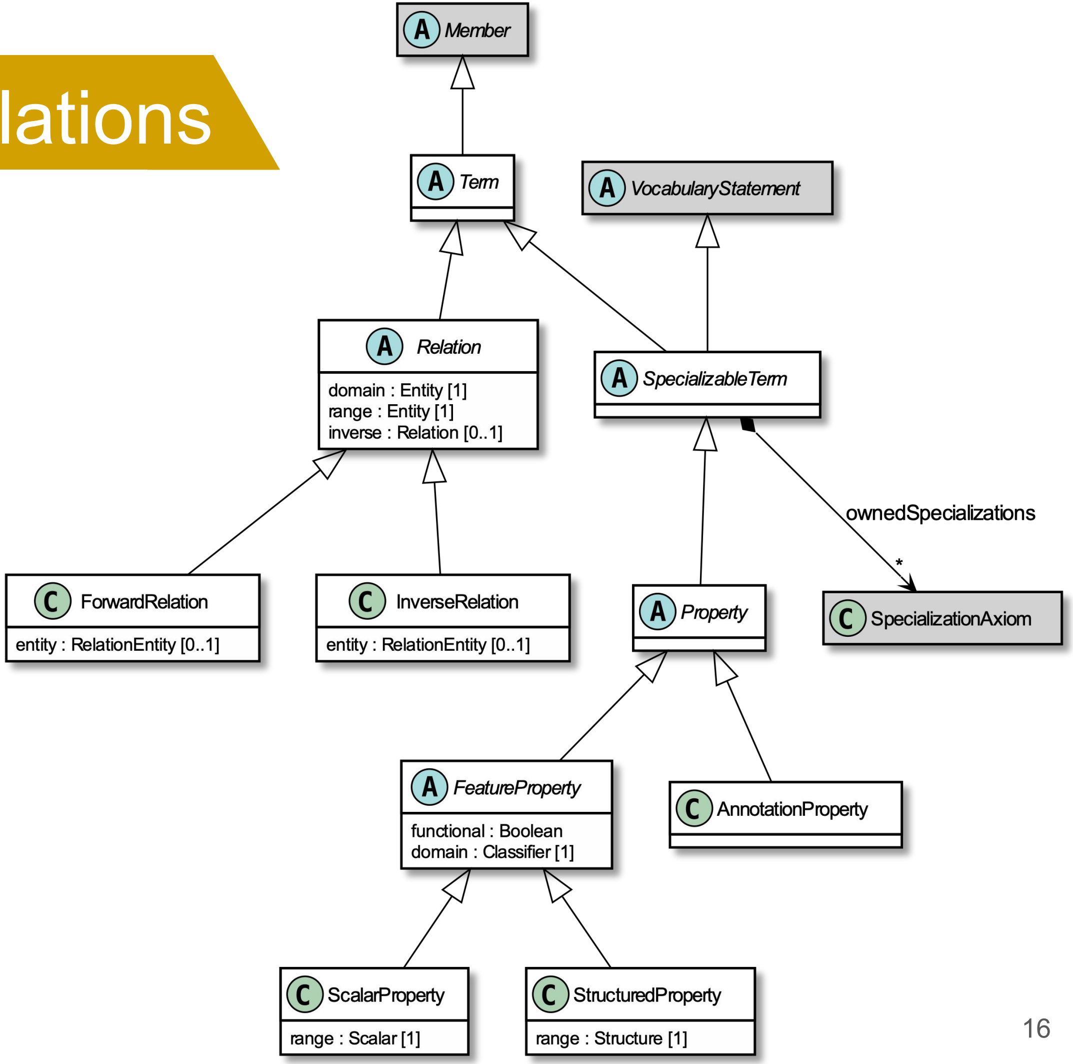


Terms and Rules

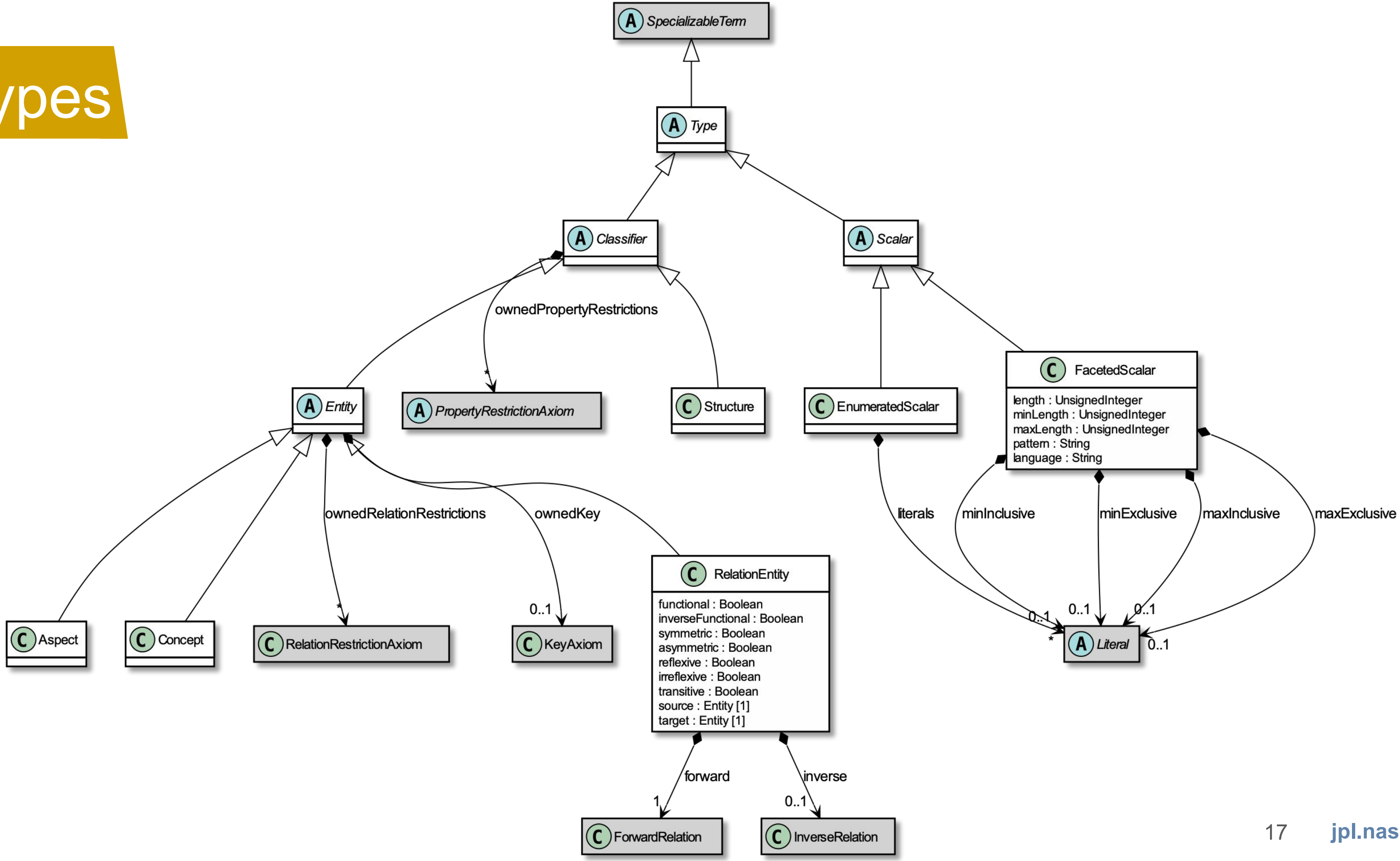
- Vocabulary members are terms and rules of a given business domain or concern
    - Term
        - Specializable Term
            - Type
                - Classifier
                    - Entity
                        - **Aspect**
                        - **Concept**
                        - **Relation Entity**
                    - **Structure**
                - Scalar
                    - **Faceted Scalar**
                    - **Enumerated Scalar**
            - Property
                - Feature Property
                    - **Scalar Property**
                    - **Structured Property**
                - **Annotation Property**
        - Relation
            - **Forward Relation**
            - **Inverse Relation**
    - **Rule**

```
vocabulary <http://opencaesar.io/mission> with # as mission {
    extends <http://www.w3.org/2001/XMLSchema>
    aspect NamedElement
    aspect PerformingElement :> NamedElement
    aspect PerformedElement :> NamedElement
    concept Component :> PerformingElement
    concept Function :> PerformedElement
    relation entity Performs [
        from PerformingElement
        to PerformedElement
        forward performs
        inverse isPerformedBy
        functional
    ]
    scalar property name [
        domain PerformingElement
        range xsd:string
    ]
    rule R1 [ Leader(X)  & performs(X □ Y) => leads(X □ Y) ]
}
```

Properties and Relations

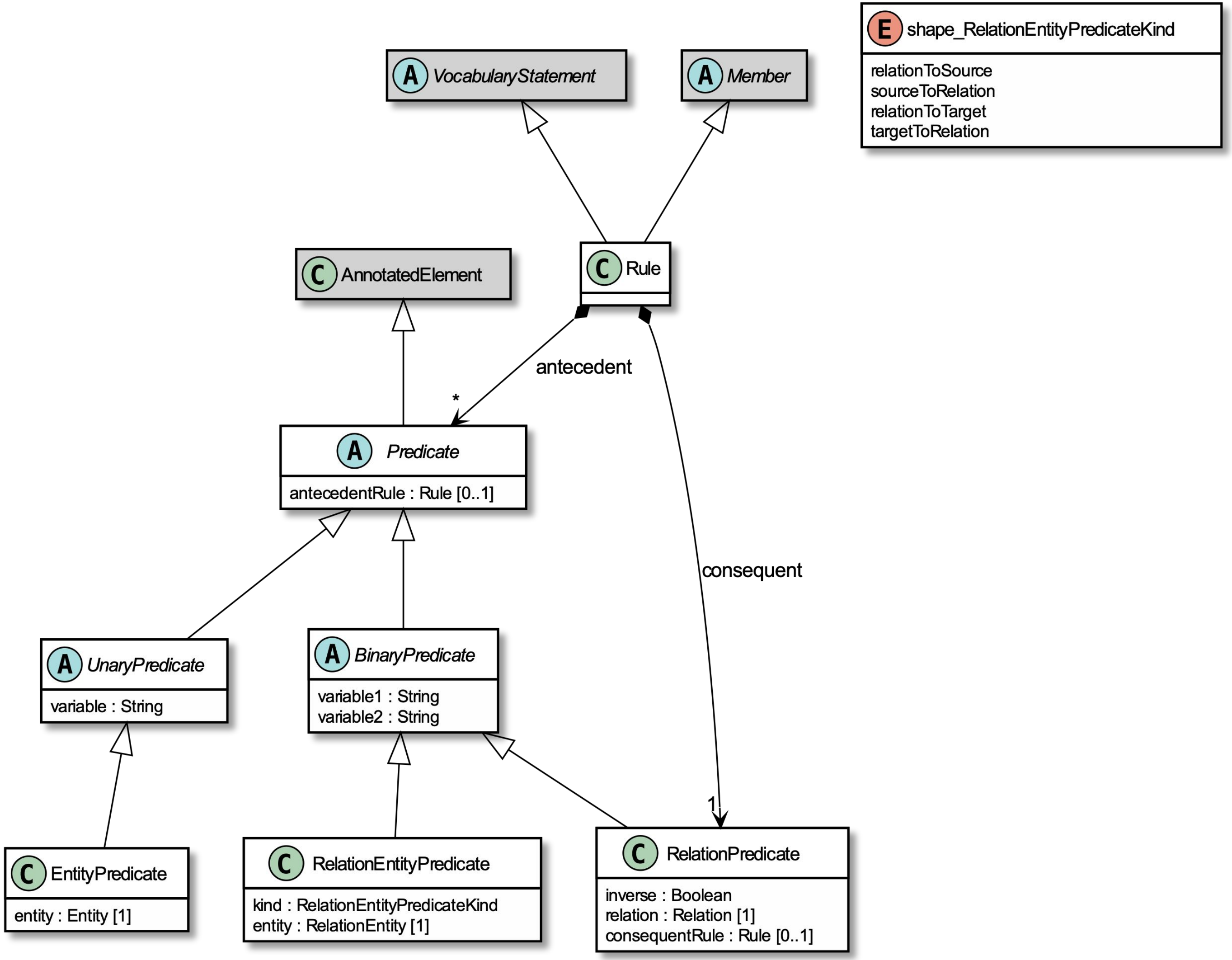


Types



A SpecializableTerm
A Type
A Classifier
A Scalar
ownedPropertyRestrictions
*
A Entity
A PropertyRestrictionAxiom
C Structure
C EnumeratedScalar
C FacetedScalar
length : UnsignedInteger
minLength : UnsignedInteger
maxLength : UnsignedInteger
pattern : String
language : String
ownedRelationRestrictions
ownedKey
literals
minInclusive
minExclusive
maxInclusive
maxExclusive
*
0..1
C RelationEntity
functional : Boolean
inverseFunctional : Boolean
symmetric : Boolean
asymmetric : Boolean
reflexive : Boolean
irreflexive : Boolean
transitive : Boolean
source : Entity [1]
target : Entity [1]
0..1
0..1
0..1
*
0..1
C Aspect
C Concept
C RelationRestrictionAxiom
C KeyAxiom
A Literal
forward
inverse
1
0..1
C ForwardRelation
C InverseRelation

Rules



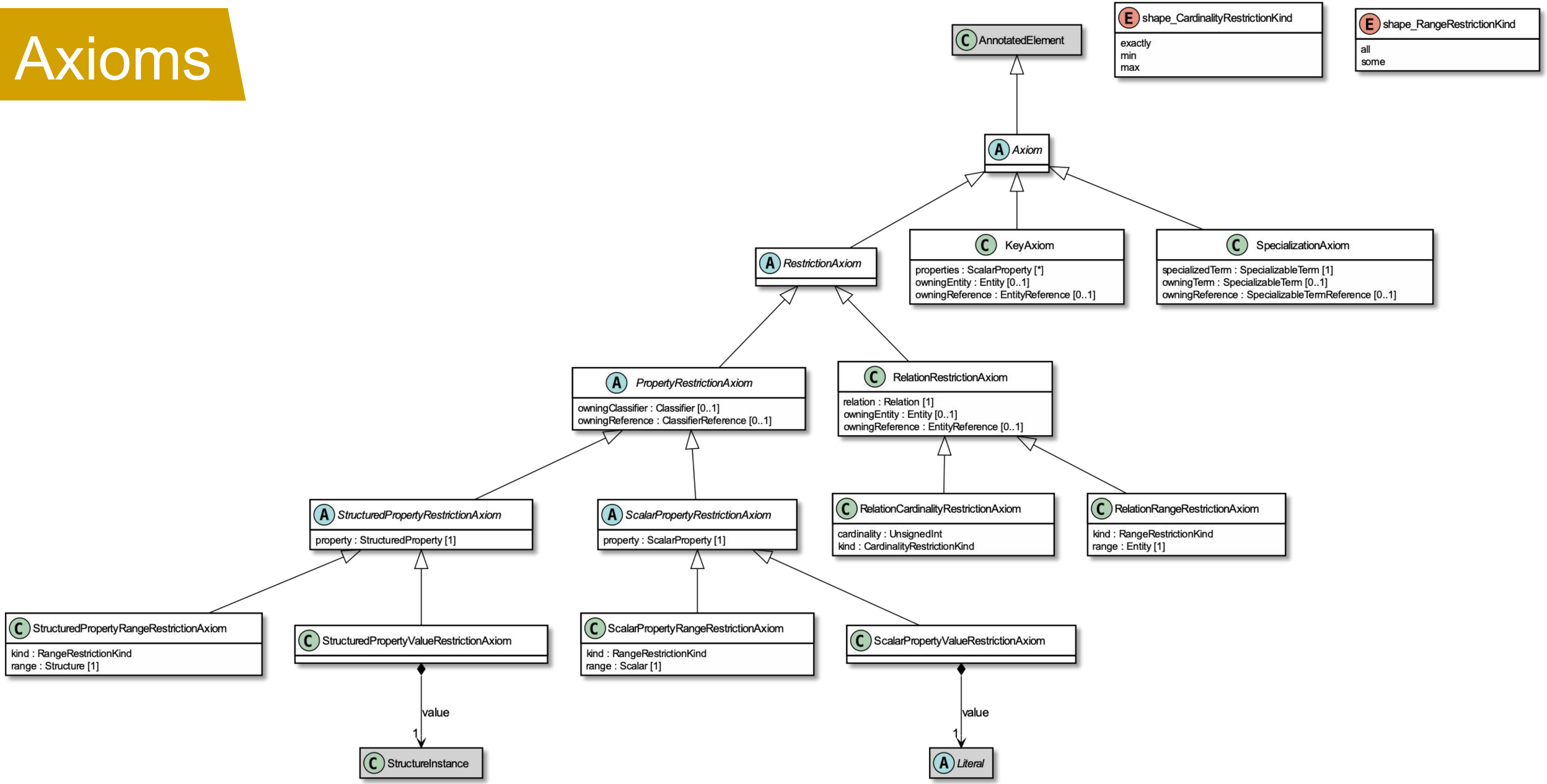
E shape_RelationEntityPredicateKind
relationToSource
sourceToRelation
relationToTarget
targetToRelation
A VocabularyStatement
A Member
C Rule
C AnnotatedElement
antecedent
*
consequent
A Predicate
antecedentRule : Rule [0..1]
A UnaryPredicate
variable : String
A BinaryPredicate
variable1 : String
variable2 : String
1
C EntityPredicate
entity : Entity [1]
C RelationEntityPredicate
kind : RelationEntityPredicateKind
entity : RelationEntity [1]
C RelationPredicate
inverse : Boolean
relation : Relation [1]
consequentRule : Rule [0..1]

Axioms

- Vocabularies can specify axioms on its own or imported members
  - **Specialization Axiom**
  - Restriction Axiom
    - Property Restriction Axiom
      - Scalar Property Restriction Axiom
        - **Scalar Property Value Restriction Axiom**
        - **Scalar Property Range Restriction Axiom**
        - **Scalar Property Cardinality Restriction Axiom**
      - Structured Property Restriction Axiom
        - **Structured Property Value Restriction Axiom**
        - **Structured Property Range Restriction Axiom**
        - **Structured Property Cardinality Restriction Axiom**
    - Relation Restriction Axiom
      - **Relation Range Restriction Axiom**
      - **Relation Cardinality Restriction Axiom**
  - **Key Axiom**

```
vocabulary <http://opencaesar.io/mission> with # as mission {
    extends <http://www.w3.org/2001/XMLSchema>
    concept Component :> PerformingElement [
        restricts relation performs to Function
        restricts relation performs to min 1
    ]
    concept Task :> Function [
        restricts scalar property name to TaskName
        key name
    ]
    @label “Task Name”
    scalar TaskName :> xsd:string [
        min length 10
        pattern “[0..9]+”
    ]
    annotation property label
}
```

Axioms



C AnnotatedElement

E shape_CardinalityRestrictionKind
exactly
min
max

E shape_RangeRestrictionKind
all
some

A *Axiom*

A *RestrictionAxiom*

C KeyAxiom
properties : ScalarProperty [*]
owningEntity : Entity [0..1]
owningReference : EntityReference [0..1]

C SpecializationAxiom
specializedTerm : SpecializableTerm [1]
owningTerm : SpecializableTerm [0..1]
owningReference : SpecializableTermReference [0..1]

A *PropertyRestrictionAxiom*
owningClassifier : Classifier [0..1]
owningReference : ClassifierReference [0..1]

C RelationRestrictionAxiom
relation : Relation [1]
owningEntity : Entity [0..1]
owningReference : EntityReference [0..1]

A *StructuredPropertyRestrictionAxiom*
property : StructuredProperty [1]

A *ScalarPropertyRestrictionAxiom*
property : ScalarProperty [1]

C RelationCardinalityRestrictionAxiom
cardinality : UnsignedInt
kind : CardinalityRestrictionKind

C RelationRangeRestrictionAxiom
kind : RangeRestrictionKind
range : Entity [1]

C StructuredPropertyRangeRestrictionAxiom
kind : RangeRestrictionKind
range : Structure [1]

C StructuredPropertyValueRestrictionAxiom

C ScalarPropertyRangeRestrictionAxiom
kind : RangeRestrictionKind
range : Scalar [1]

C ScalarPropertyValueRestrictionAxiom

value
1

C StructureInstance

value
1

A *Literal*

Vocabulary Bundles

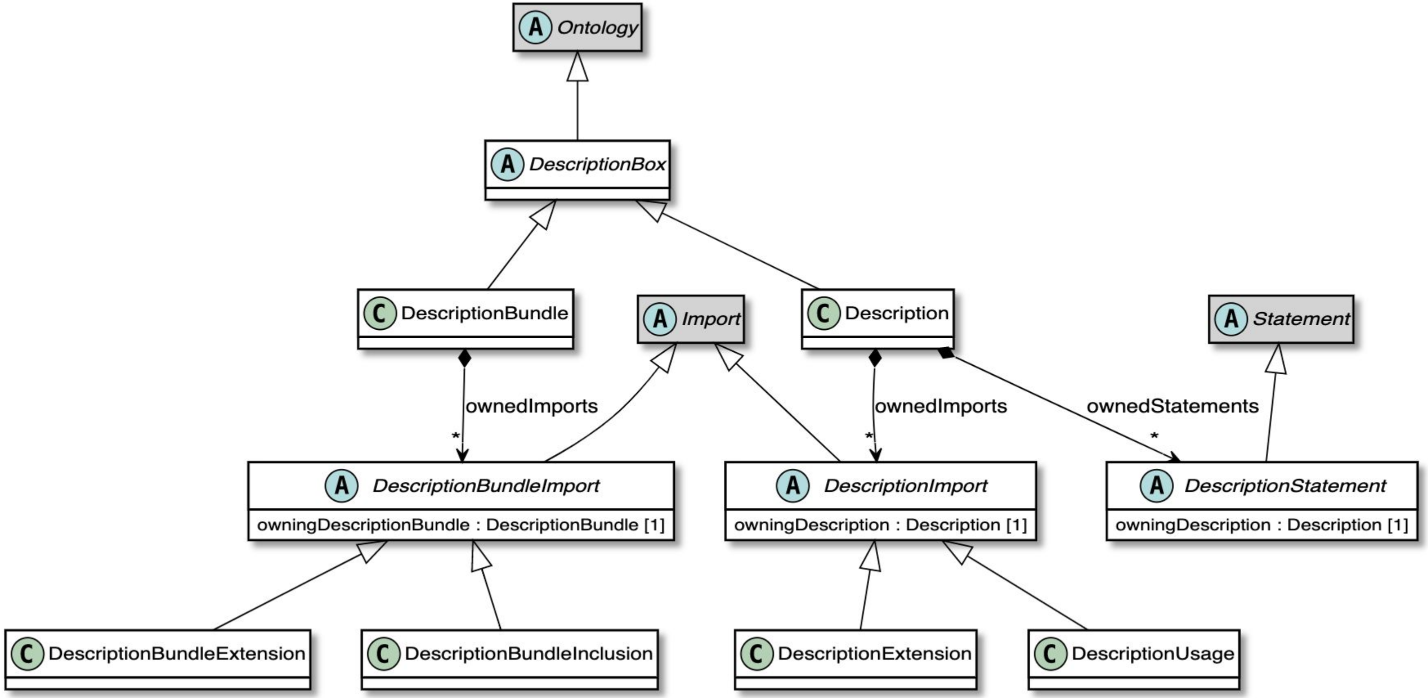
- Vocabulary Bundles includes vocabularies and closes the world on them
    - This means terms without explicit common sub terms become disjoint
    - This means the extent of super terms become the union of those of sub terms
- Vocabulary Bundles can be used to define model kinds in a methodology

```
vocabulary bundle <http://opencaesar.io/uml> with # as uml {
    includes <http://opencaesar.io/uml/classes>
    includes <http://opencaesar.io/uml/statecharts>
    …
}
```

```
vocabulary <http://opencaesar.io/sysml/blocks> with # as blocks {
    extends <http://opencaesar.io/uml/classes>
}
```

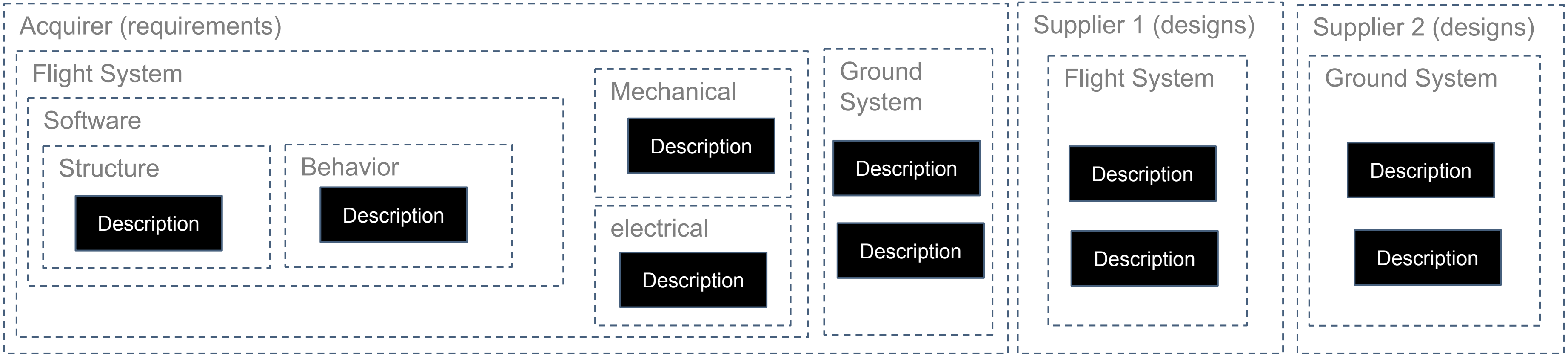
```
vocabulary bundle <http://opencaesar.io/sysml> with # as sysml {
    extends  <http://opencaesar.io/uml>
    includes <http://opencaesar.io/sysml/blocks>
    includes <http://opencaesar.io/sysml/requirements>
    includes <http://opencaesar.io/sysml/parametrics>
    …
}
```

Descriptions



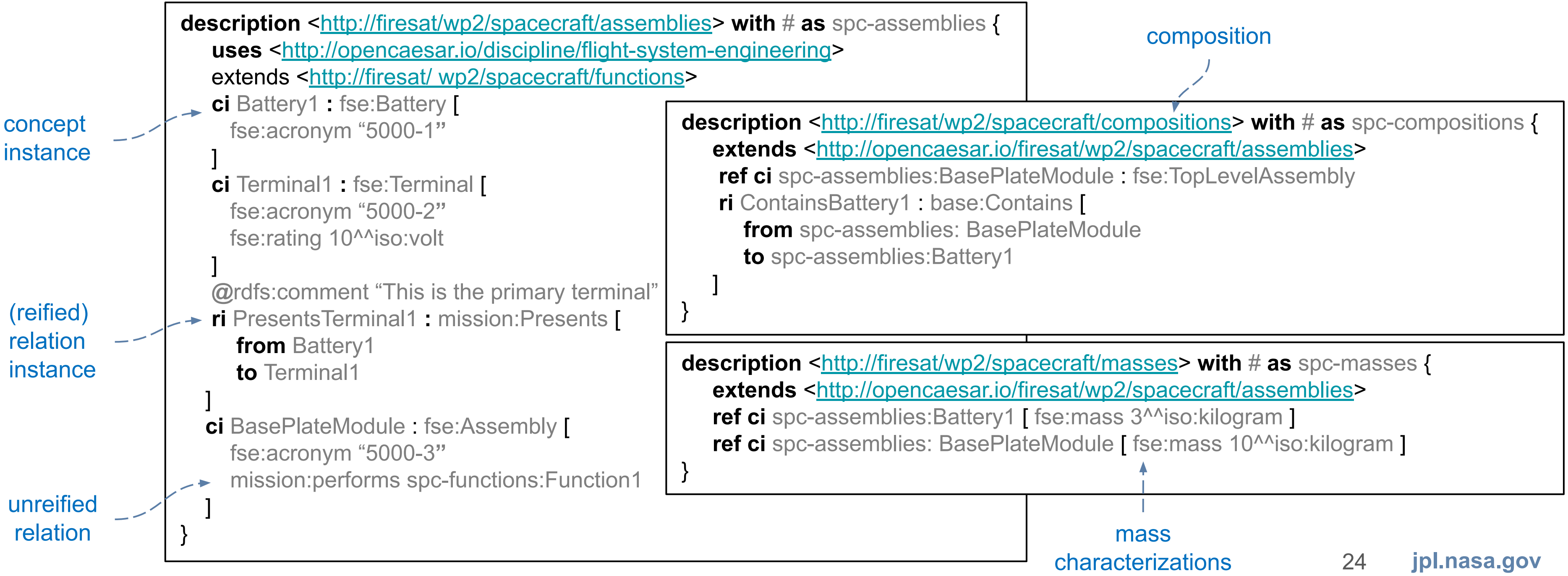
Descriptions

- Description uses vocabularies (or a vocabualry bundles) to describe a system
- Descriptions can be organized across methodological boundaries like
  - Disciplines (e.g., structure, behavior, fault tolerance, V&V, I/T)
  - Domains (e.g., electrical, mechanical, software)
  - Subsystems (e.g., launch vehicle, spacecraft, payload)
  - Organizations (e.g., acquirer, supplier, contractor)



Instances

- Description describes a system using a bundle of methodology-specific vocabularies
- Description consists of assertions made on concept and relation instances

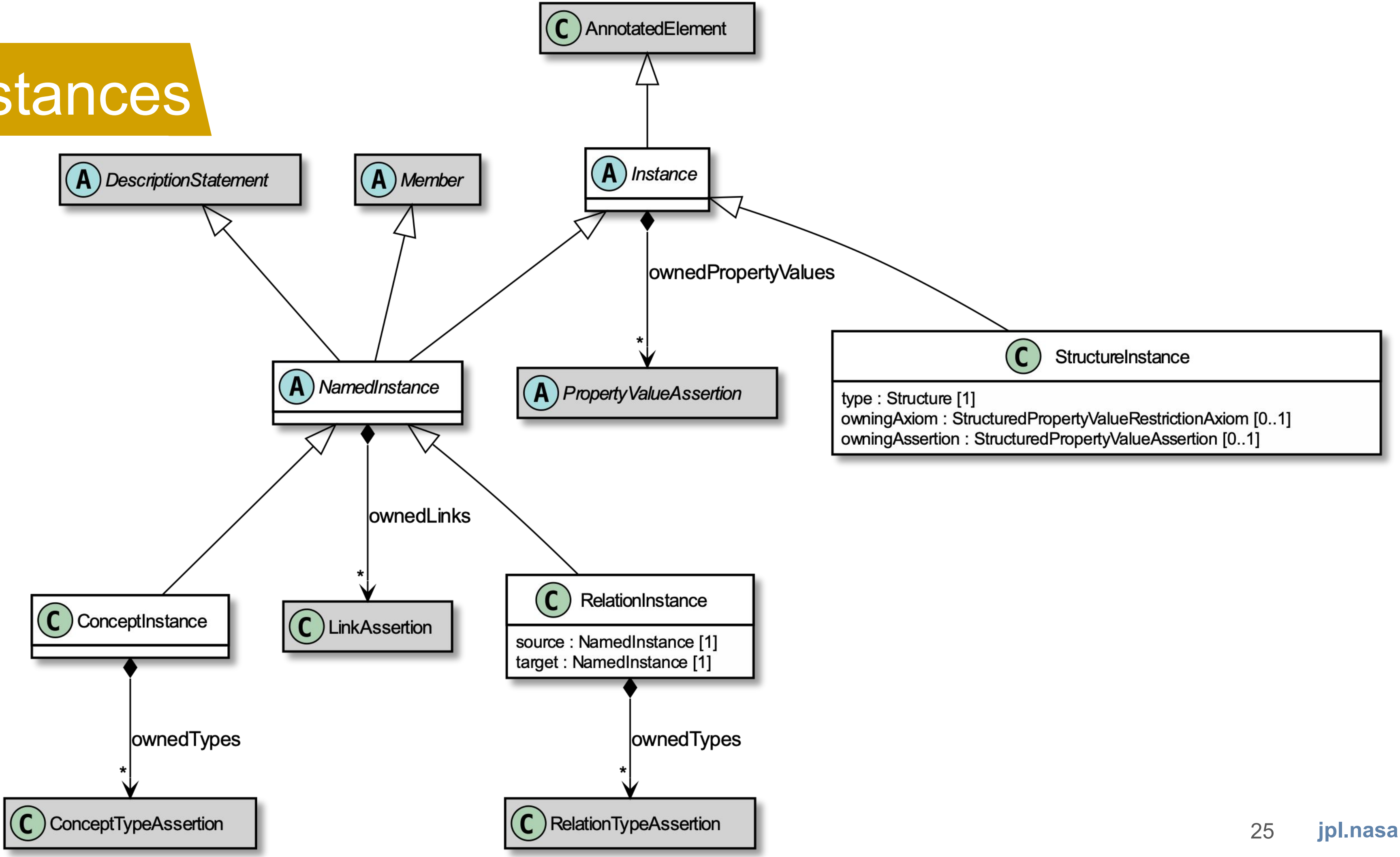


```
description <http://firesat/wp2/spacecraft/assemblies> with # as spc-assemblies {
    uses <http://opencaesar.io/discipline/flight-system-engineering>
    extends <http://firesat/ wp2/spacecraft/functions>
    ci Battery1 : fse:Battery [
      fse:acronym "5000-1"
    ]
    ci Terminal1 : fse:Terminal [
      fse:acronym "5000-2"
      fse:rating 10^^iso:volt
    ]
    @rdfs:comment "This is the primary terminal"
    ri PresentsTerminal1 : mission:Presents [
       from Battery1
       to Terminal1
    ]
    ci BasePlateModule : fse:Assembly [
      fse:acronym "5000-3"
      mission:performs spc-functions:Function1
    ]
}
```

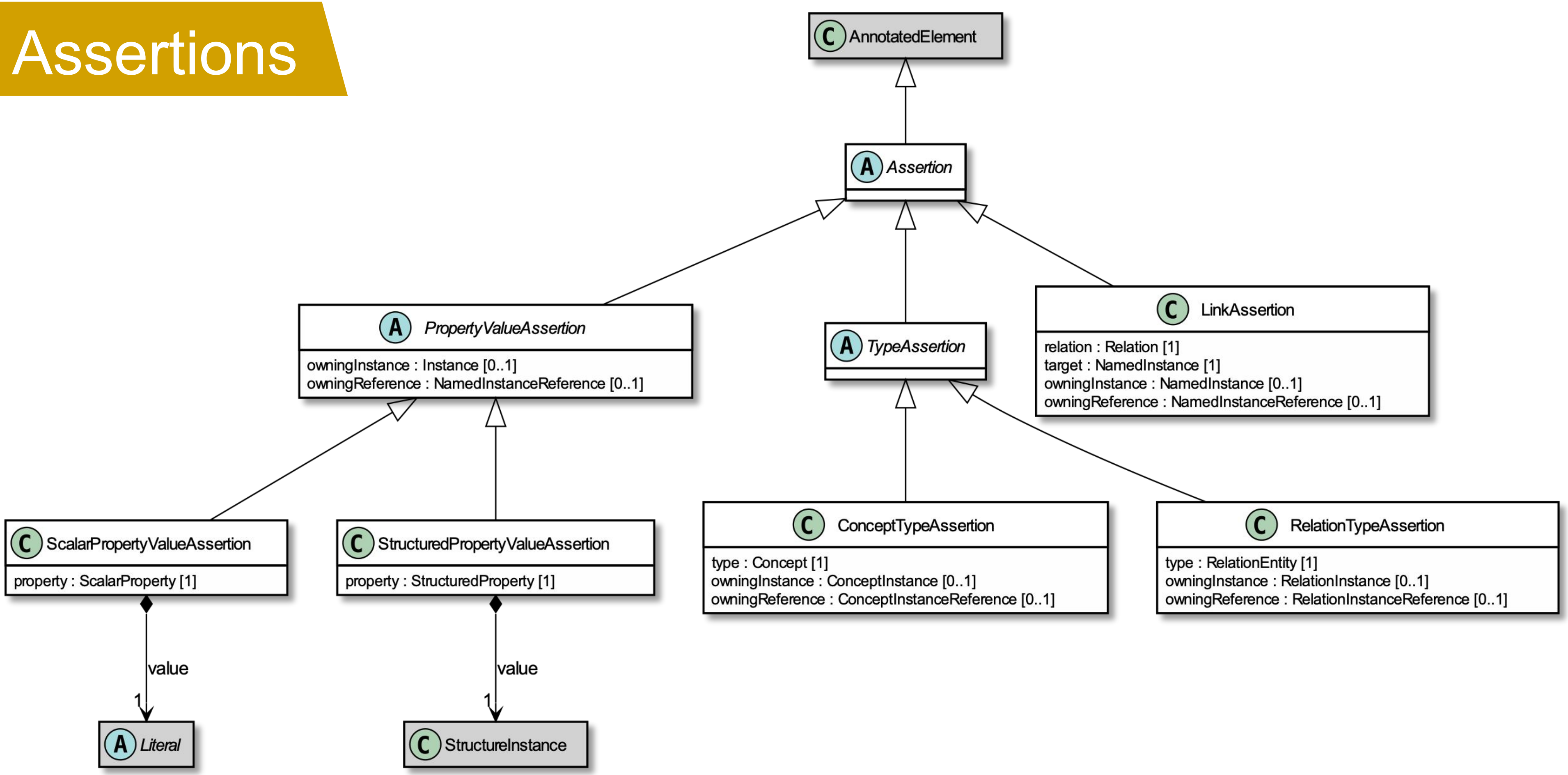
```
description <http://firesat/wp2/spacecraft/compositions> with # as spc-compositions {
    extends <http://opencaesar.io/firesat/wp2/spacecraft/assemblies>
     ref ci spc-assemblies:BasePlateModule : fse:TopLevelAssembly
     ri ContainsBattery1 : base:Contains [
        from spc-assemblies: BasePlateModule
        to spc-assemblies:Battery1
     ]
}
```

```
description <http://firesat/wp2/spacecraft/masses> with # as spc-masses {
    extends <http://opencaesar.io/firesat/wp2/spacecraft/assemblies>
    ref ci spc-assemblies:Battery1 [ fse:mass 3^^iso:kilogram ]
    ref ci spc-assemblies: BasePlateModule [ fse:mass 10^^iso:kilogram ]
}
```

Instances


C AnnotatedElement
A DescriptionStatement
A Member
A Instance
ownedPropertyValues
*
A NamedInstance
A PropertyValueAssertion
C StructureInstance
type : Structure [1]
owningAxiom : StructuredPropertyValueRestrictionAxiom [0..1]
owningAssertion : StructuredPropertyValueAssertion [0..1]
ownedLinks
*
C ConceptInstance
C LinkAssertion
C RelationInstance
source : NamedInstance [1]
target : NamedInstance [1]
ownedTypes
*
ownedTypes
*
C ConceptTypeAssertion
C RelationTypeAssertion

Assertions


C AnnotatedElement
A Assertion
A PropertyValueAssertion
owningInstance : Instance [0..1]
owningReference : NamedInstanceReference [0..1]
A TypeAssertion
C LinkAssertion
relation : Relation [1]
target : NamedInstance [1]
owningInstance : NamedInstance [0..1]
owningReference : NamedInstanceReference [0..1]
C ScalarPropertyValueAssertion
property : ScalarProperty [1]
C StructuredPropertyValueAssertion
property : StructuredProperty [1]
C ConceptTypeAssertion
type : Concept [1]
owningInstance : ConceptInstance [0..1]
owningReference : ConceptInstanceReference [0..1]
C RelationTypeAssertion
type : RelationEntity [1]
owningInstance : RelationInstance [0..1]
owningReference : RelationInstanceReference [0..1]
value
1
A Literal
value
1
C StructureInstance

Description Bundles

- Description Bundles includes descriptions that can be reasoned on together
  - These bundles can represent a union of fragments, design alternatives, etc.
  - These are the bundles that we will run consistency check on

```
description bundle <http://opencaesar.io/project> with # as project {
    includes <http://opencaesar.io/project/wbs>
    includes <http://opencaesar.io/project/system/decomp>
    includes <http://opencaesar.io/project/system/mass>
    includes <http://opencaesar.io/project/system/power>
    …
}
```

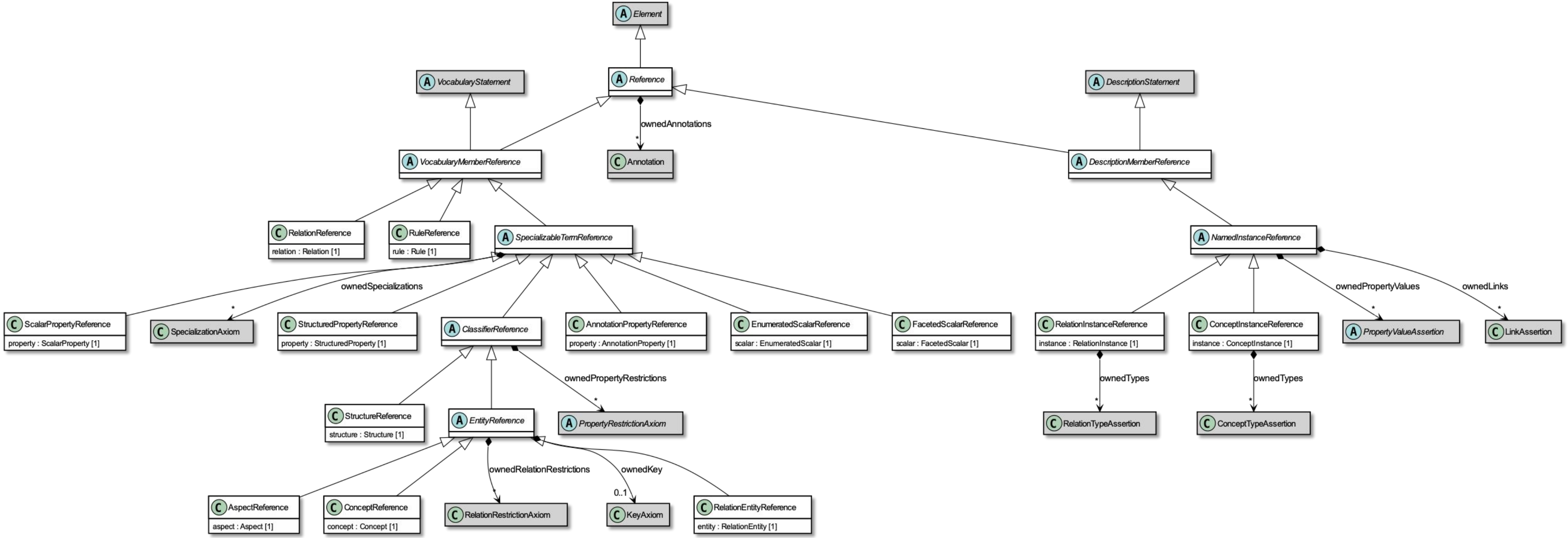
```
description <http://opencaesar.io/project/wbs> with # as wbs {
    extends <opencaesar.io/project/system/subsystems>
}
```

```
description <http://opencaesar.io/project/system/mass> with # as mass {
    extends <opencaesar.io/project/system/components>
}
```

```
description <http://opencaesar.io/project/system/decomp> with # as
decomp {
    extends <opencaesar.io/project/system/components>
}
```

```
description <http://opencaesar.io/project/system/power> with # as power {
    extends <opencaesar.io/project/system/components>
}
```

References


A Element
A Reference
A VocabularyStatement
A DescriptionStatement
ownedAnnotations
*
A VocabularyMemberReference
C Annotation
A DescriptionMemberReference
C RelationReference
relation : Relation [1]
C RuleReference
rule : Rule [1]
A SpecializableTermReference
A NamedInstanceReference
ownedSpecializations
ownedPropertyValues
ownedLinks
C ScalarPropertyReference
property : ScalarProperty [1]
*
C SpecializationAxiom
C StructuredPropertyReference
property : StructuredProperty [1]
A ClassifierReference
C AnnotationPropertyReference
property : AnnotationProperty [1]
C EnumeratedScalarReference
scalar : EnumeratedScalar [1]
C FacetedScalarReference
scalar : FacetedScalar [1]
C RelationInstanceReference
instance : RelationInstance [1]
C ConceptInstanceReference
instance : ConceptInstance [1]
*
A PropertyValueAssertion
*
C LinkAssertion
ownedPropertyRestrictions
ownedTypes
ownedTypes
*
C StructureReference
structure : Structure [1]
A EntityReference
A PropertyRestrictionAxiom
*
C RelationTypeAssertion
*
C ConceptTypeAssertion
ownedRelationRestrictions
ownedKey
*
0..1
C AspectReference
aspect : Aspect [1]
C ConceptReference
concept : Concept [1]
C RelationRestrictionAxiom
C KeyAxiom
C RelationEntityReference
entity : RelationEntity [1]

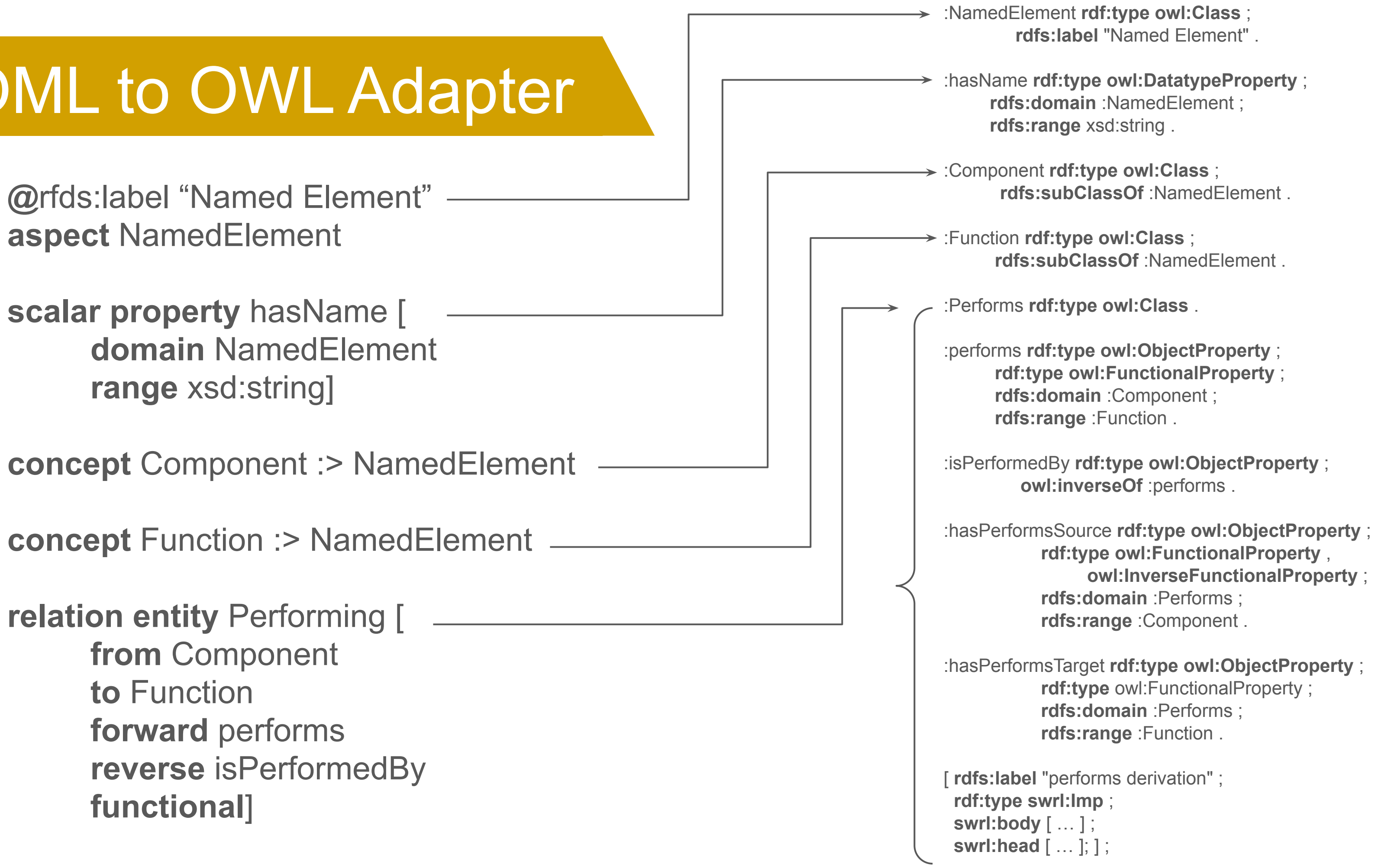
Example

- A set of core vocabularies are available on:
  - https://github.com/opencaesar/core-vocabularies
- A set of example SE vocabularies are available on:
  - https://github.com/modelware/oml-example/vocabularies
- A set of example project descriptions are available on:
  - https://github.com/modelware/oml-example/descriptions

OML Tools

- **OML Tools (https://github.com/opencaesar/oml-tools)**
  - **OML Bikeshed**: generates documentation
  - **OML Merge**: merges several OML datasets into one
  - **OML Validate**: validates an OML dataset
- **OWL Adapter (https://github.com/opencaesar/owl-adapter)**
  - **OML to OWL**: converts a dataset from OML to OWL
- **OWL Tools (https://github.com/opencaesar/owl-tools)**
  - **OWL Close World**: a library of bundle closure algorithms
  - **OWL Fuseki**: starts and stops a headless Fuseki server (triple store)
  - **OWL Diff**: calculates delta between two OWL datasets
  - **OWL Reason**: runs DL reason on an OWL dataset
  - **OWL Load**: loads a dataset to a SPARQL endpoint
  - **OWL Query**: sends a set of SPARQL queries to a triple store
  - **OWL Shacl**: sends a set of Shacl validation rules to a triple store

OML to OWL Adapter



```
@rfds:label “Named Element”
aspect NamedElement

scalar property hasName [
        domain NamedElement
        range xsd:string]

concept Component :> NamedElement

concept Function :> NamedElement

relation entity Performing [
        from Component
        to Function
        forward performs
        reverse isPerformedBy
        functional]
```

```
:NamedElement rdf:type owl:Class ;
          rdfs:label "Named Element" .

:hasName rdf:type owl:DatatypeProperty ;
       rdfs:domain :NamedElement ;
       rdfs:range xsd:string .

:Component rdf:type owl:Class ;
        rdfs:subClassOf :NamedElement .

:Function rdf:type owl:Class ;
        rdfs:subClassOf :NamedElement .

:Performs rdf:type owl:Class .

:performs rdf:type owl:ObjectProperty ;
        rdf:type owl:FunctionalProperty ;
        rdfs:domain :Component ;
        rdfs:range :Function .

:isPerformedBy rdf:type owl:ObjectProperty ;
           owl:inverseOf :performs .

:hasPerformsSource rdf:type owl:ObjectProperty ;
               rdf:type owl:FunctionalProperty ,
                     owl:InverseFunctionalProperty ;
               rdfs:domain :Performs ;
               rdfs:range :Component .

:hasPerformsTarget rdf:type owl:ObjectProperty ;
               rdf:type owl:FunctionalProperty ;
               rdfs:domain :Performs ;
               rdfs:range :Function .

[ rdfs:label "performs derivation" ;
 rdf:type swrl:Imp ;
 swrl:body [ … ] ;
 swrl:head [ … ]; ] ;
```

Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov