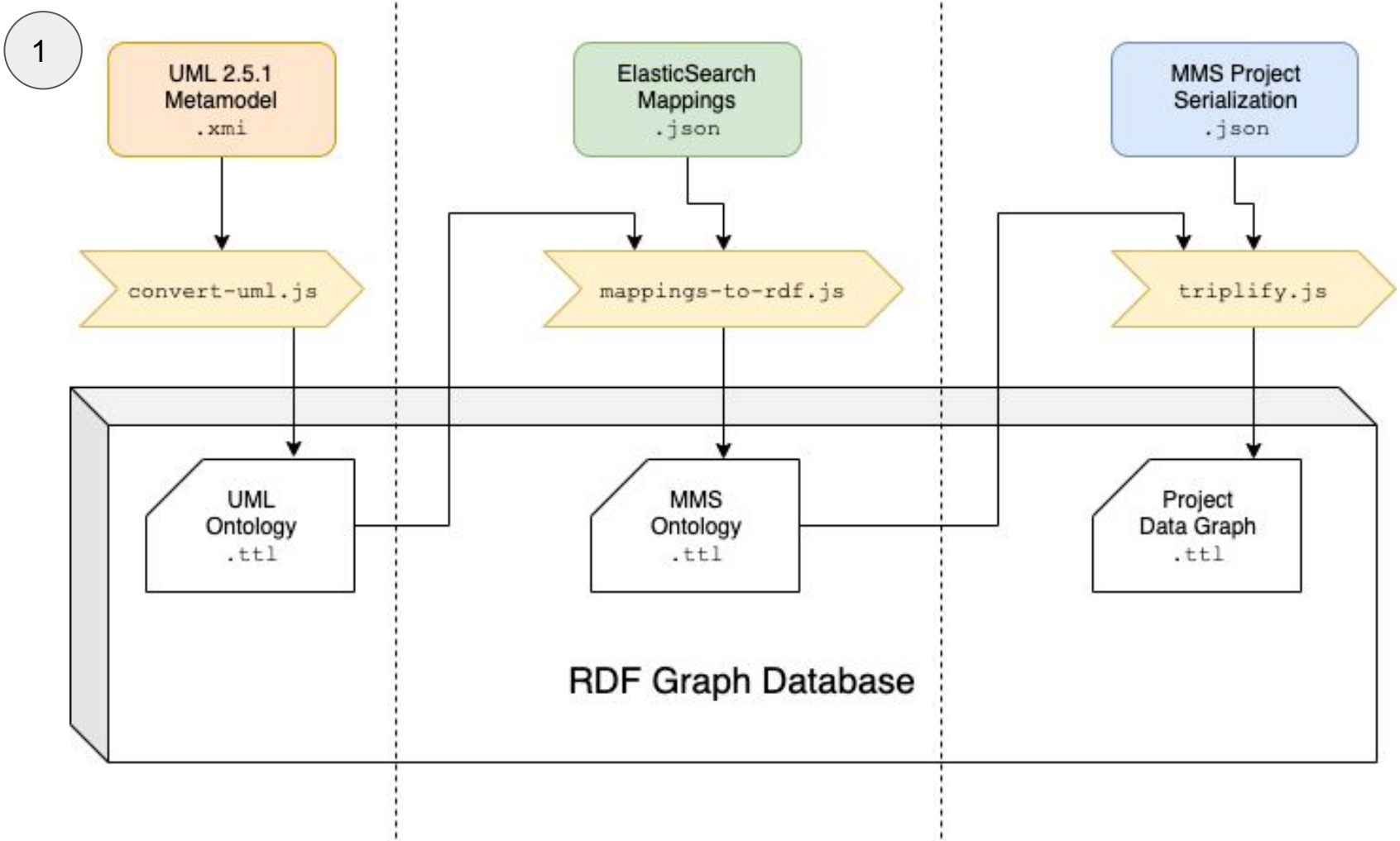
MMS in RDF

Blake Regalia


1
UML 2.5.1
Metamodel
.xmi
ElasticSearch
Mappings
.json
MMS Project
Serialization
.json
convert-uml.js
mappings-to-rdf.js
triplify.js
UML
Ontology
.ttl
MMS
Ontology
.ttl
Project
Data Graph
.ttl
RDF Graph Database

UML Metamodel Excerpt: ‘Nodes’

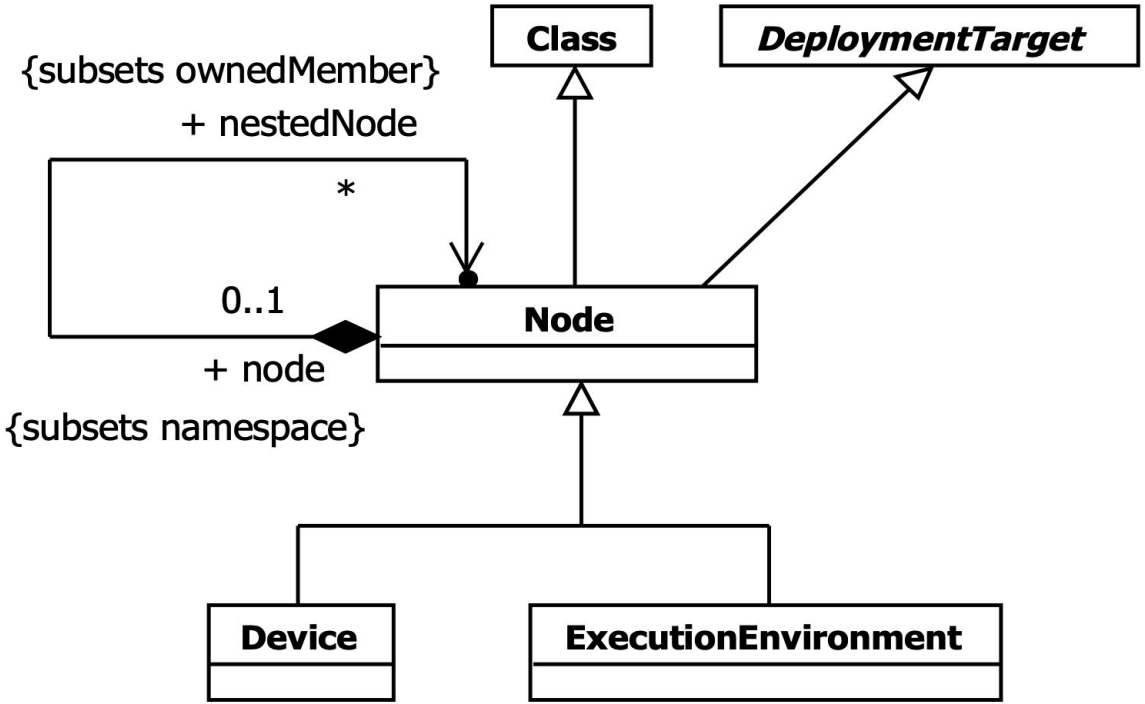
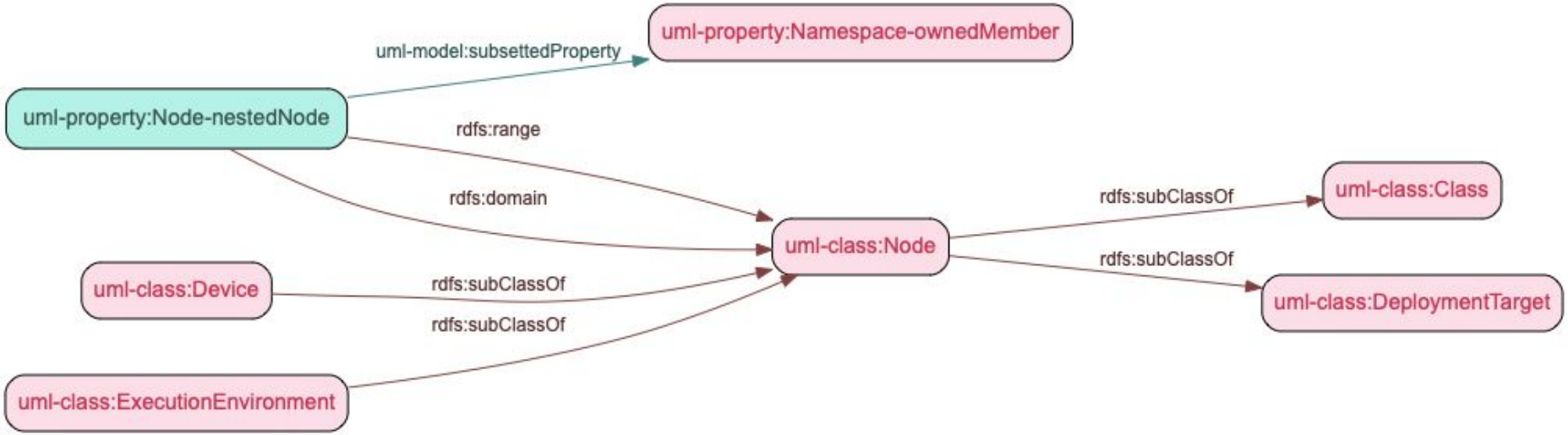
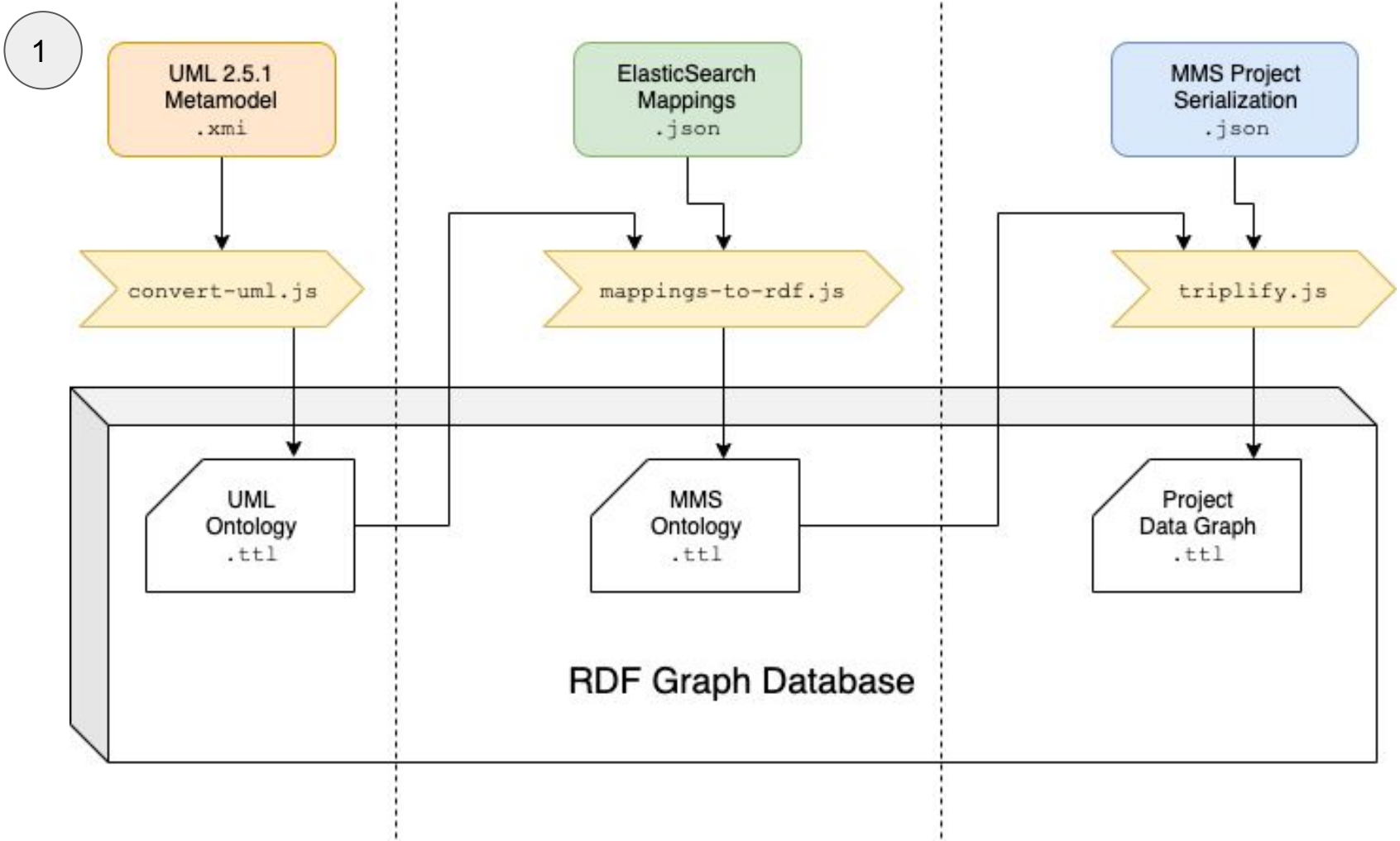


Figure 19.11 Nodes

UML ‘Nodes’ Represented in RDF



```
uml-class:Node xmi:type uml:Class ;
    xmi:id "Node" ;
    xmi:packagedElementOf uml-class:Deployments .

uml-class:Node rdfs:comment "A Node is computational resource upon which artifacts may be deployed for
    execution. Nodes can be interconnected through communication paths to define network structures."@en .

uml-class:Node rdfs:subClassOf uml-class:Class .

uml-class:Node rdfs:subClassOf uml-class:DeploymentTarget .

uml-property:Node_nestedNode xmi:type uml:Property ;
    xmi:id "Node-nestedNode" ;
    xmi:ownedAttributeOf uml-class:Node ;
    rdfs:label "Node-nestedNode" ;
    uml-model:name "nestedNode" ;
    rdfs:domain uml-class:Node ;
    uml-model:compositeAggregation true ;
    rdfs:range uml-class:Node .

uml-property:Node_nestedNode uml-model:subsettedProperty uml-property:Namespace_ownedMember .

uml-property:Node_nestedNode rdfs:comment "The Nodes that are defined (nested) within the Node."@en .

uml-property:Node_nestedNode uml-model:lowerValue uml-property:Node_nestedNode_upperValue .

uml-property:Node_nestedNode_upperValue xmi:type uml:LiteralUnlimitedNatural ;
    xmi:id "Node-nestedNode-_upperValue" ;
    uml-model:value "*" .
```

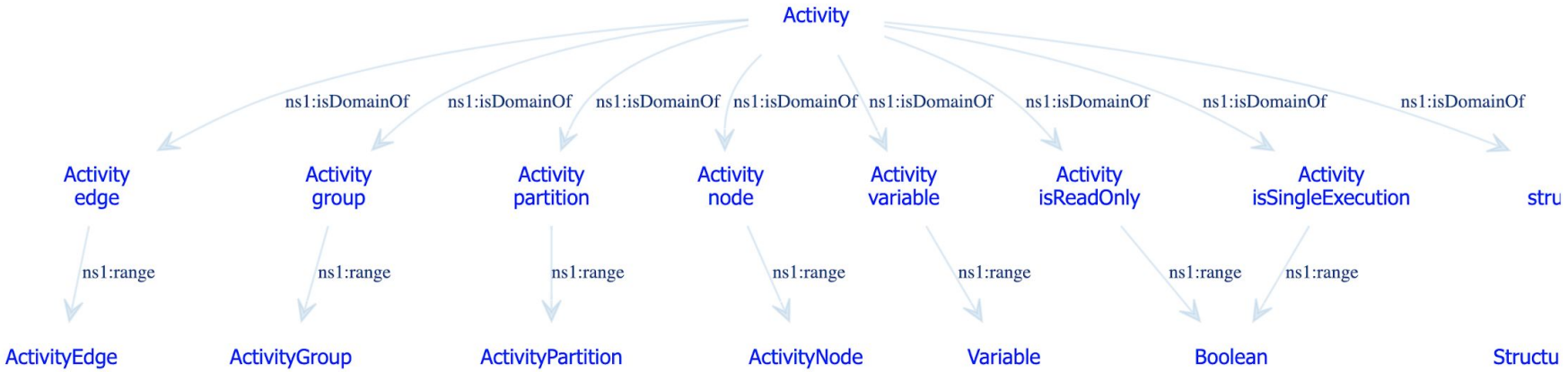

1
UML 2.5.1
Metamodel
.xmi
ElasticSearch
Mappings
.json
MMS Project
Serialization
.json
convert-uml.js
mappings-to-rdf.js
triplify.js
UML
Ontology
.ttl
MMS
Ontology
.ttl
Project
Data Graph
.ttl
RDF Graph Database

Query UML model for range of all properties that belong to an Activity

In [3]:

```
%display diagram

construct {
    ?domain :isDomainOf ?property .
    ?property :range ?range .
} from mms-graph:vocabulary {
    ?property xmi:type uml:Property ;
              rdfs:domain ?domain ;
              rdfs:range ?range .
    values ?domain {
        uml-class:Activity
    }
}
```

Display: svg



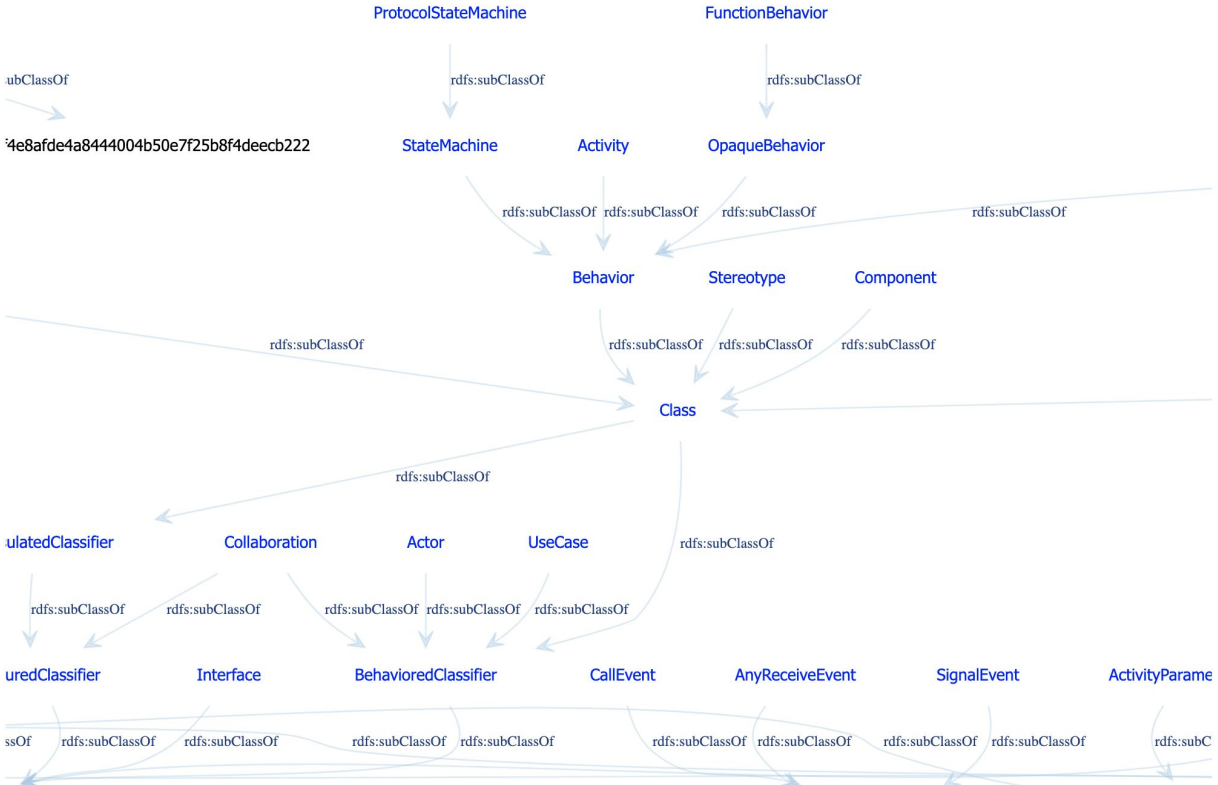
%3

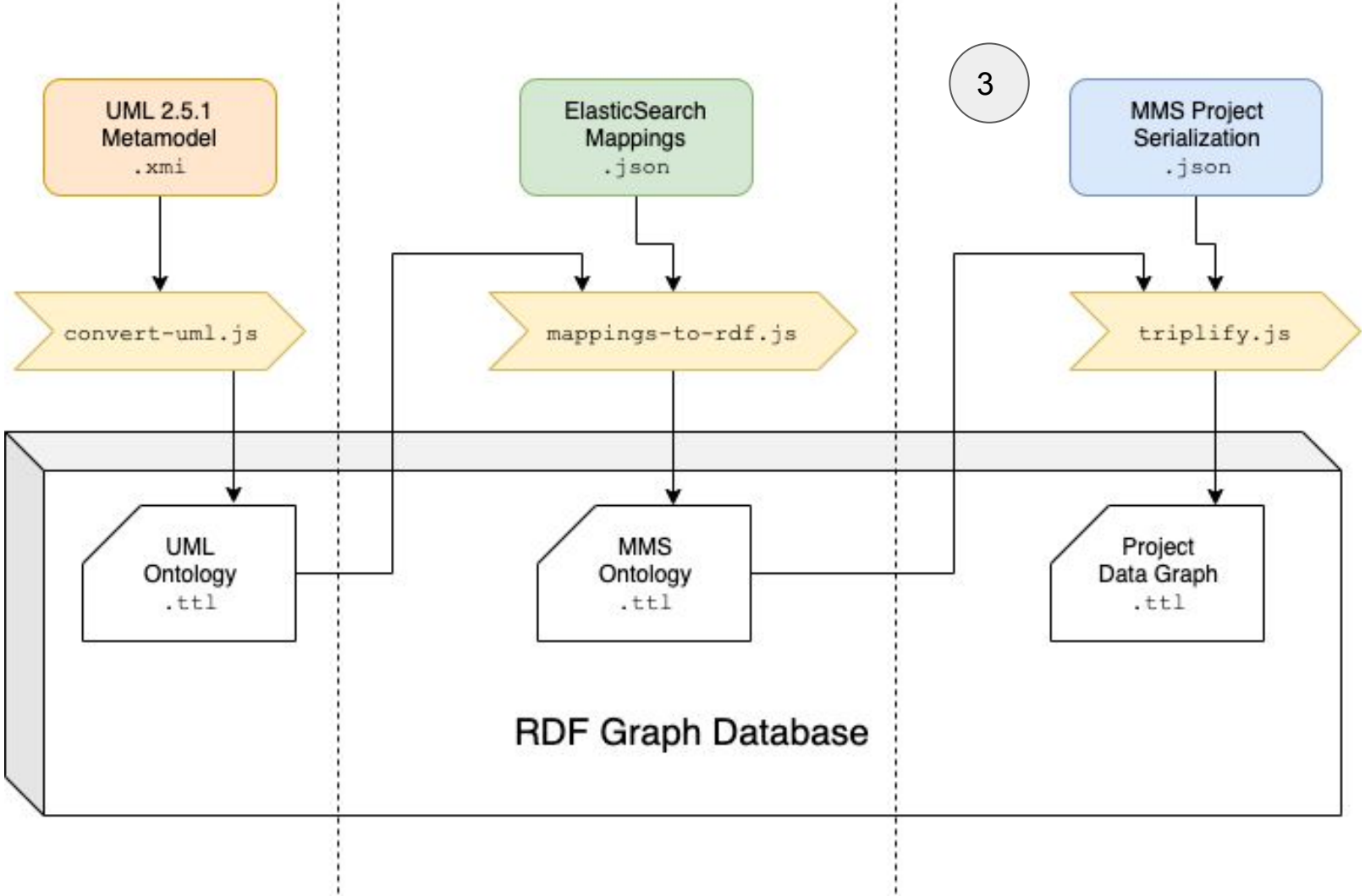
Construct diagram of UML metamodel class hierarchy

In [4]:

```
%display diagram

construct {
    ?class rdfs:subClassOf ?super .
} from mms-graph:vocabulary {
    ?class rdfs:subClassOf ?super .
}
```

Display: svg




UML 2.5.1
Metamodel
.xmi
ElasticSearch
Mappings
.json
MMS Project
Serialization
.json
convert-uml.js
mappings-to-rdf.js
triplify.js
UML
Ontology
.ttl
MMS
Ontology
.ttl
Project
Data Graph
.ttl
RDF Graph Database

Model Element Data

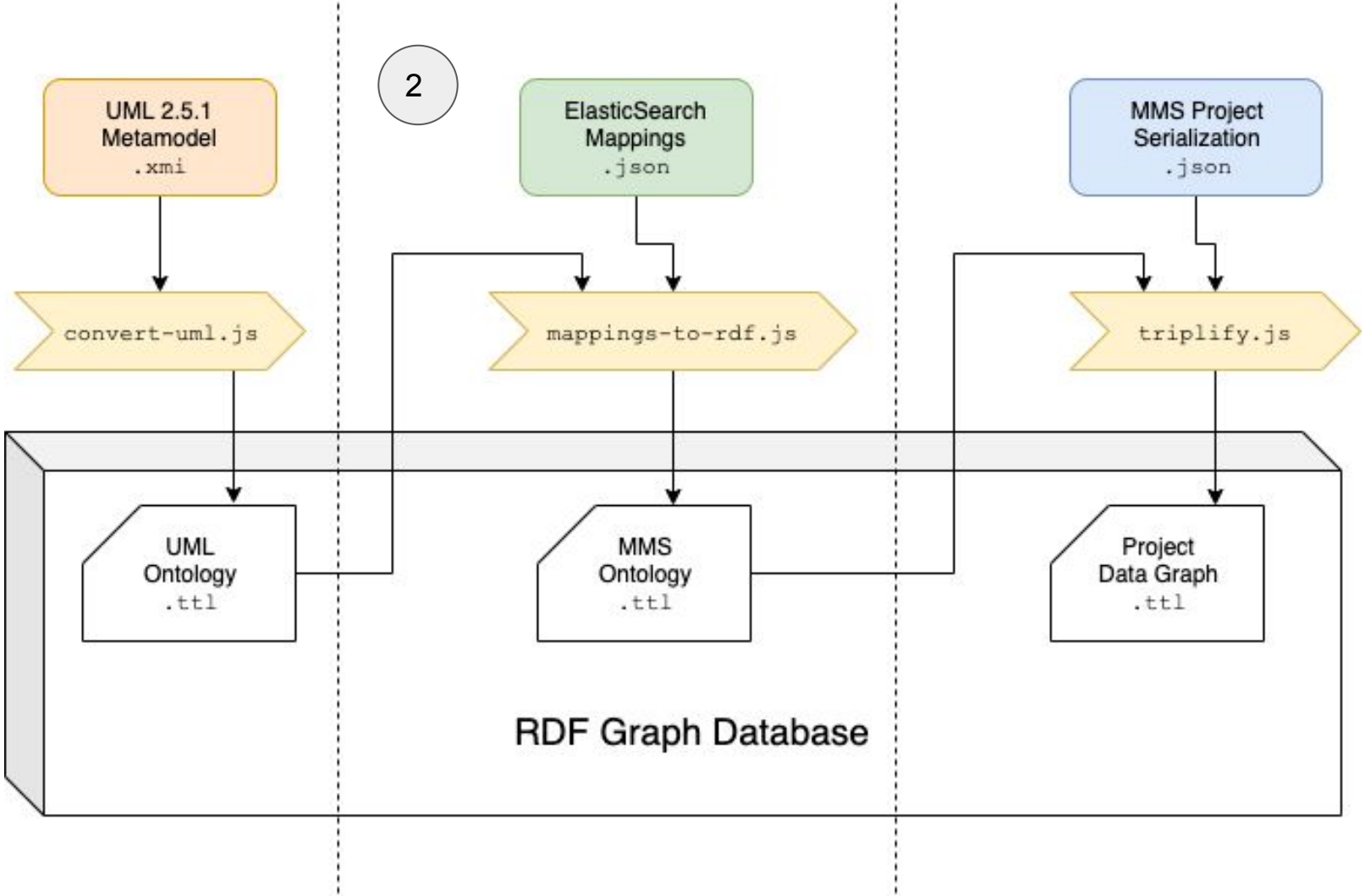
The input data source is MMS' serialization of model elements in JSON.

Keys of JSON objects represent either:

- some **UML property** -- such as the keys "isAbstract" or "targetIds".
- or some **derived property** including metadata about the commit such as: *who and when created or last modified the element, the commit id, the branch*, etc.

Need to have **knowledge** of serialization's structure in order to **transform model element data into RDF** in a way that is conformant with generated UML ontology.

```
      "_inRefIds": [
        "master"
      ],
      "structuredNodeInputIds": [],
      "localPreconditionIds": [],
      "bodyPartIds": [
        "_18_0_6_876026b_1494866664018_615922_183752",
        "_18_0_6_876026b_1494866989432_632113_183800",
        "_18_0_6_876026b_1494869500672_400531_183955"
      ],
      "_elasticId": "bf0002eb-3965-42ac-bd2f-0fc9db5fb793",
      "structuredNodeOutputIds": [],
      "nameExpression": null,
      "packageImportIds": [],
      "edgeIds": [
        "_18_0_6_876026b_1494867441408_836874_183823",
        "_18_0_6_876026b_1494869500681_910063_183957",
        "_18_0_6_876026b_1494869610935_433369_183994"
      ],
      "variableIds": [],
      "inStructuredNodeId": null,
      "_refId": "master",
      "id": "_18_0_6_876026b_1494866614221_588750_183735",
      "_modifier": "rkarbanAdmin",
      "outgoingIds": [],
      "clientDependencyIds": [],
      "handlerIds": [],
      "inActivityId": null,
      "mdExtensionsIds": [],
      "incomingIds": [
        "_18_0_6_876026b_1494865902273_651463_183694"
      ],
```


2
UML 2.5.1
Metamodel
.xmi
ElasticSearch
Mappings
.json
MMS Project
Serialization
.json
convert-uml.js
mappings-to-rdf.js
triplify.js
UML
Ontology
.ttl
MMS
Ontology
.ttl
Project
Data Graph
.ttl
RDF Graph Database

Deriving ‘MMS Ontology’: a UML Ontology Extension

Keeping the UML Ontology as is, **define a new RDF property** for each ‘relation’ that arises from applying the ElasticSearch mapping template.

A ‘relation’ in this case represents either:

- a **direct alias** of a UML property -- e.g., ‘Transition-source’ => ‘sourceVertex’
- a **reification** of a UML property to contain either a **set or list** of objects.
- or a **derived property** (such as metadata about the commit) that does not exist in the UML metamodel.

Directly Aliasing UML Properties

```
# In the original UML ontology...
uml-property:Transition_source xmi:type uml:Property ;
    xmi:id "Transition-source" ;
    rdfs:label "Transition-source" ;
    rdfs:comment "Designates the originating Vertex (State or Pse
    xmi:ownedAttributeOf uml-class:Trans
    uml-model:name "source" ;
    rdfs:domain uml-class:Transition ;
    rdfs:range uml-class:Vertex .

uml-property:DirectedRelationship_sour
    xmi:id "DirectedRelationship-source"
    rdfs:label "DirectedRelationship-so
    rdfs:comment "Specifies the source
    xmi:ownedAttributeOf uml-class:Dire
    uml-model:name "source" ;
    rdfs:domain uml-class:DirectedRelat
    rdfs:range uml-class:Element .
```

```
# In the derivative MMS ontology...
mms-property:sourceVertex a mms-ontology:UmlObjectPrope
    mms-ontology:key "sourceId" ;
    rdfs:label "sourceVertex" ;
    rdfs:comment "The Vertex that is the source of this
    rdfs:domain uml-class:Transition ;
    rdfs:range uml-class:Vertex ;
    mms-ontology:umlPropertySource uml-property:Transiti

mms-property:sourceElement a mms-ontology:UmlObjectProp
    mms-ontology:key "sourceId" ;
    rdfs:label "sourceElement" ;
    rdfs:comment "The Element that is the source of this
    rdfs:domain uml-class:DirectedRelationship ;
    rdfs:range uml-class:Element ;
    mms-ontology:umlPropertySource uml-property:Directed
```

Reified UML Properties to Sets/Lists

```
# In the original UML ontology...
uml-property:InstanceSpecification_slot xmi:type uml:Property ;
    xmi:id "InstanceSpecification-slot" ;
    xmi:ownedAttributeOf uml-class:InstanceSpecification ;
    rdfs:label "InstanceSpe
    uml-model:name "slot" ;
    rdfs:domain uml-class:I
    uml-model:compositeAggr
    rdfs:range uml-class:Sl
```

```
# In the derivative MMS ontology...
mms-property:slots a mms-ontology:UmlObjectProperty ;
    mms-ontology:key "slotIds" ;
    rdfs:label "slots" ;
    rdfs:comment "List of Slots that belong to this InstanceS
    rdfs:domain uml-class:InstanceSpecification ;
    rdfs:range mms-class:SlotsList ;
    mms-ontology:listItemRange uml-class:Slot ;
    mms-ontology:umlPropertySource uml-property:InstanceSpec

mms-class:SlotsList a owl:Class ;
    mms-ontology:category mms-class:ElementList ;
    rdfs:subClassOf [
        rdf:type owl:Class ;
        owl:intersectionOf (
            rdf:List
            [
```

Reified UML Properties to Sets/Lists cont’d

The **need for a container** is inferred from the “Ids” suffix of the mapping key label.

A **new class is created** (e.g., “SlotsList”) which defines a **list of elements** that **must be** of the given **type** (e.g., “Slot”).

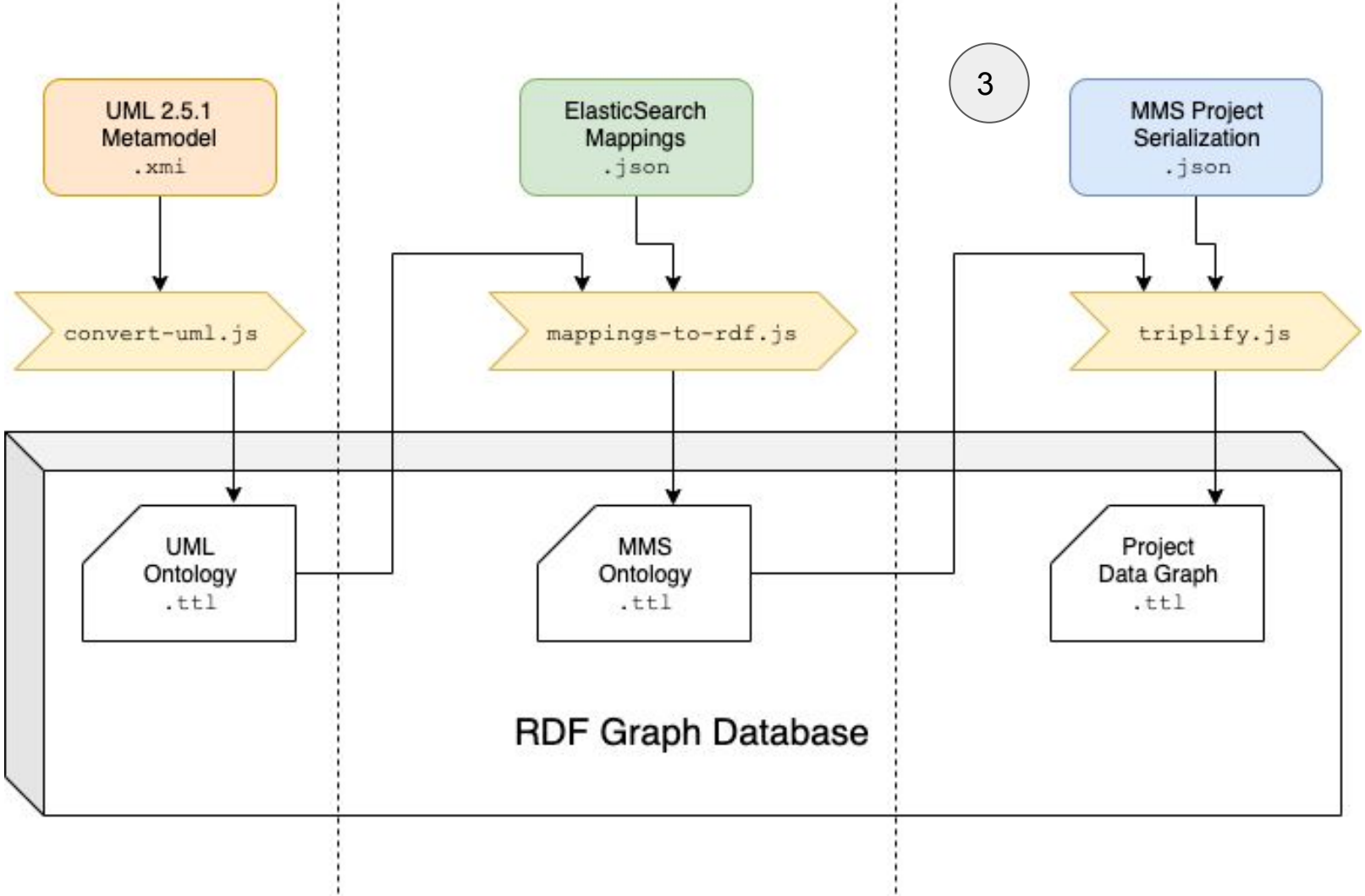
Together with the **range** of the **reified property**, these assertions will enable **validation** of instance data in the eventual project graph.

Derived Properties

```
# Derived properties that describe commit metadata
mms-property:created a mms-ontology:DerivedDatatypeProperty ;
    mms-ontology:key "_created" ;
    rdfs:label "created" ;
    rdfs:comment "Based on the key '_created'"@en ;
    rdfs:domain mms-class:Element ;
    rdfs:range xsd:dateTime ;
    mms-ontology:mappingDomain mms-class:Element .

mms-property:creator a mms-ontology:DerivedObjectProperty ;
    mms-ontology:key "_creator" ;
    rdfs:label "creator" ;
    rdfs:comment "Based on the derived property '_creator'."@en ;
    rdfs:domain mms-class:Element ;
    rdfs:range mms-class:User ;
    mms-ontology:mappingDomain mms-class:Element .
```

Derived Properties cont’d

Integrating **commit metadata** into the graph (alongside model element data) will allow users to write queries based on arbitrary constraints from **both the instance-data-level as well as the metadata-level** (...plus the metamodel level).


UML 2.5.1
Metamodel
.xmi
ElasticSearch
Mappings
.json
MMS Project
Serialization
.json
convert-uml.js
mappings-to-rdf.js
triplify.js
UML
Ontology
.ttl
MMS
Ontology
.ttl
Project
Data Graph
.ttl
RDF Graph Database

Converting Model Element Data to RDF

Triplifier script ingests MMS' serialization of the project and **queries** generated **MMS ontology** in order to **transform each element** to a set of **RDF triples**.

Each non-derived key of an object represents some specific UML property, but the label of the key **does not uniquely identify** the property. It also depends on the **type** of the element.

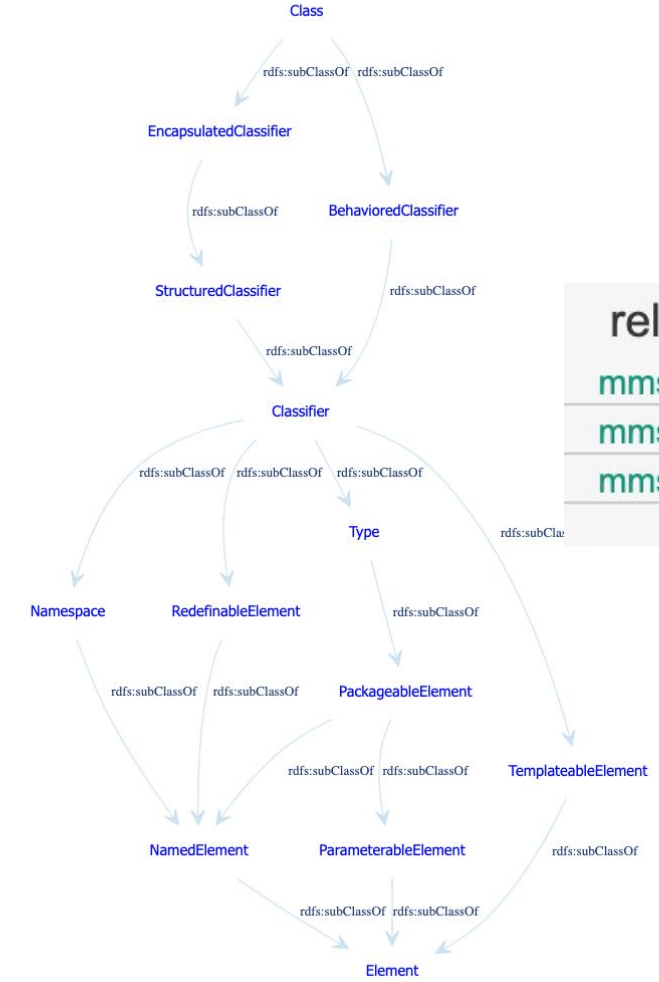
Properties that are candidates for “targetId” key

relation	domain
mms-property:targetInputPinFromSendObjectAction	uml-class:SendObjectAction
mms-property:targetInputPinFromDestroyObjectAction	uml-class:DestroyObjectAction
mms-property:targetElement	uml-class:DirectedRelationship
mms-property:targetVertex	uml-class:Transition
mms-property:targetActivityNode	uml-class:ActivityEdge
mms-property:targetInputPinFromCallOperationAction	uml-class:CallOperationAction
mms-property:targetInputPinFromSendSignalAction	uml-class:SendSignalAction

Properties that are candidates for “isAbstract” key

relation	key	class
mms-property:isAbstractClassifier	"isAbstract"	uml-class:Classifier
mms-property:isAbstractBehavioralFeature	"isAbstract"	uml-class:BehavioralFeature
mms-property:isAbstractClass	"isAbstract"	uml-class:Class

Display: svg



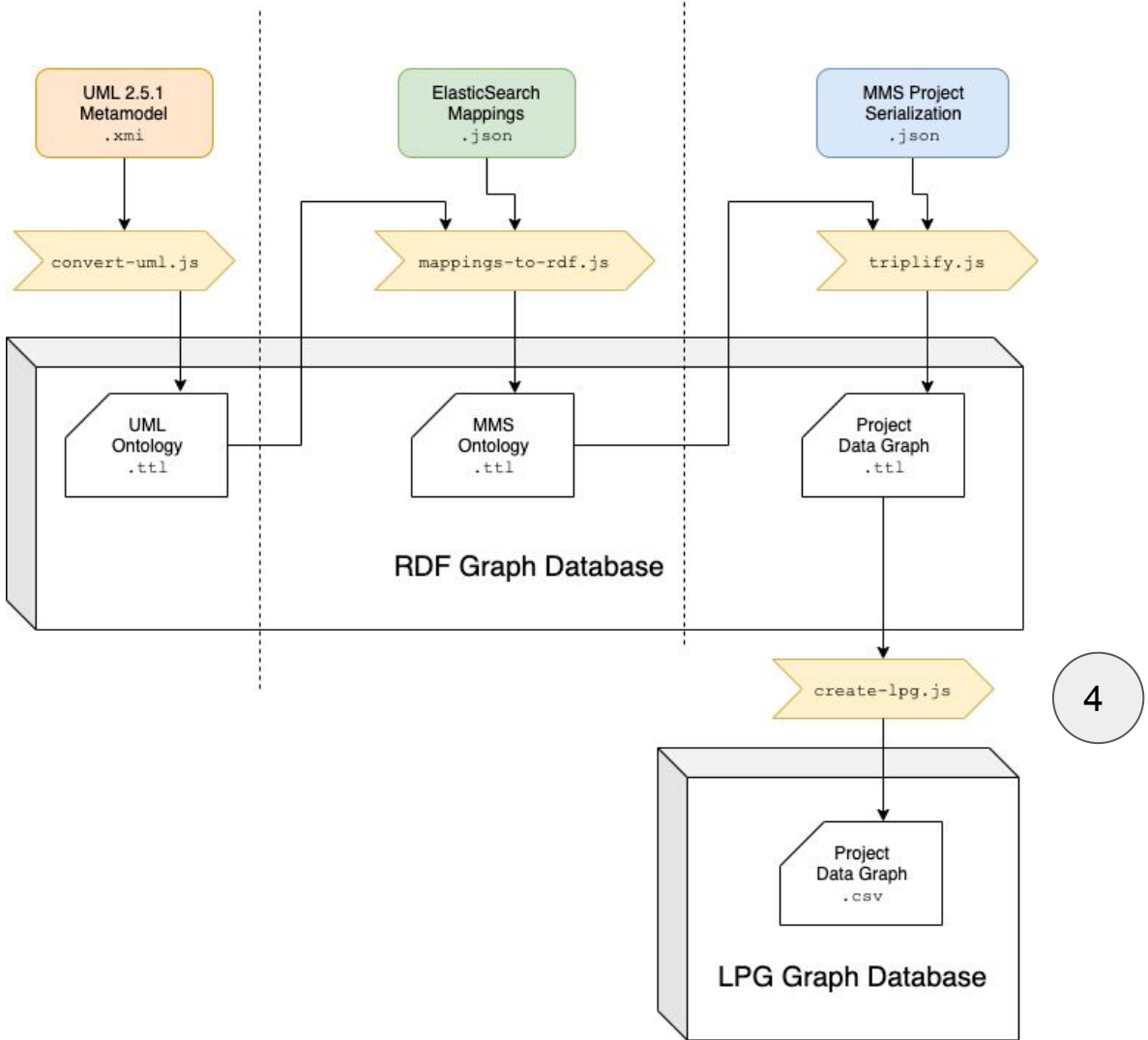
However, “Class” is a also subclass of “Classifier”...

Which relation should be used?

relation	domain
mms-property:isAbstractClassifier	uml-class:Classifier
mms-property:isAbstractBehavioralFeature	uml-class:BehavioralFeature
mms-property:isAbstractClass	uml-class:Class

Need to choose the property with the domain that is most specific to the given element type.

```
select * from mms-graph:vocabulary {
    ?property a ?propertyType ;
        mms-ontology:key ?keyLabel ;
        rdfs:domain ?propertyDomain ;
        rdfs:range ?propertyRange .

    ?propertyType rdfs:subClassOf* mms-ontology:Property .

    ?propertyDomain (^rdfs:subClassOf)* mms-class:Signal .


    # select property with most specific domain class
    filter not exists {
        ?subProperty a ?subPropertyType ;
            mms-ontology:key ?keyLabel ;
            rdfs:domain ?subPropertyDomain .

        ?subPropertyType rdfs:subClassOf* mms-ontology:Property .

        ?subPropertyDomain (^rdfs:subClassOf)* mms-class:Signal .

        filter(?subProperty != ?property)

        ?subPropertyDomain rdfs:subClassOf+ ?propertyDomain .
    }

    values ?keyLabel {
        "isAbstract"
    }
}
```


UML 2.5.1
Metamodel
.xmi
ElasticSearch
Mappings
.json
MMS Project
Serialization
.json
convert-uml.js
mappings-to-rdf.js
triplify.js
UML
Ontology
.ttl
MMS
Ontology
.ttl
Project
Data Graph
.ttl
RDF Graph Database
create-lpg.js
Project
Data Graph
.csv
LPG Graph Database

Labeled Property Graph

A **shortcoming** of the SPARQL query language is that it **does not facilitate graph traversal** queries, such as answering the question: *“what is the shortest path between these two nodes?”*

Gremlin is a graph traversal query language for labeled property graphs (LPGs).

The RDF graph-based data model **cannot be mapped** to LPGs without making some sacrifices (i.e., the transformation will inevitably be lossy).

For example, the **ontology and metamodel are omitted** from transformation since definitions for properties cannot link to the edges which instantiate them.

Labeled Property Graph cont’d

Instead, **only the project data graph** is converted to LPG as **best they can**.

Containers such as sets/lists of objects present a challenge; edges in LPGs can only have a single value so ordered lists must be reified as ad-hoc linked lists.

The **linking of datatype nodes** within RDF literals is lost in the transformation.

Cannot query for things that are a **subClass of A**, nor relations that are **subProperties of P**. Likewise, all inherent reasoning capability is lost.