Service Requirements and Standard API

Orlando Technical Meeting 06/24/16

Team

- **Axel Reichwein (Lead)**
  - Koneksys
- **Hedly Apperly**
  - PTC
- **Josh Fiengold**
  - Tom Sawyer Software
- **Charles Galey**
  - No Magic Inc

AGENDA

- Open Services for Lifecycle Collaboration (OSLC) v3.0, (30 Min)
- SysML2 Service requirements and areas of discussion, (15 min)

OPEN SERVICES FOR LIFECYCLE COLLABORATION (OSLC) CORE V3.0

OSLC v3.0: Introduction

Open Services for Lifecycle Collaboration is based on World Wide Web and Linked Data principles, such as those defined in the W3C Linked Data Platform [*LDP*], to create a cohesive set of specifications that can enable products, services, and other distributed network resources to interoperate successfully

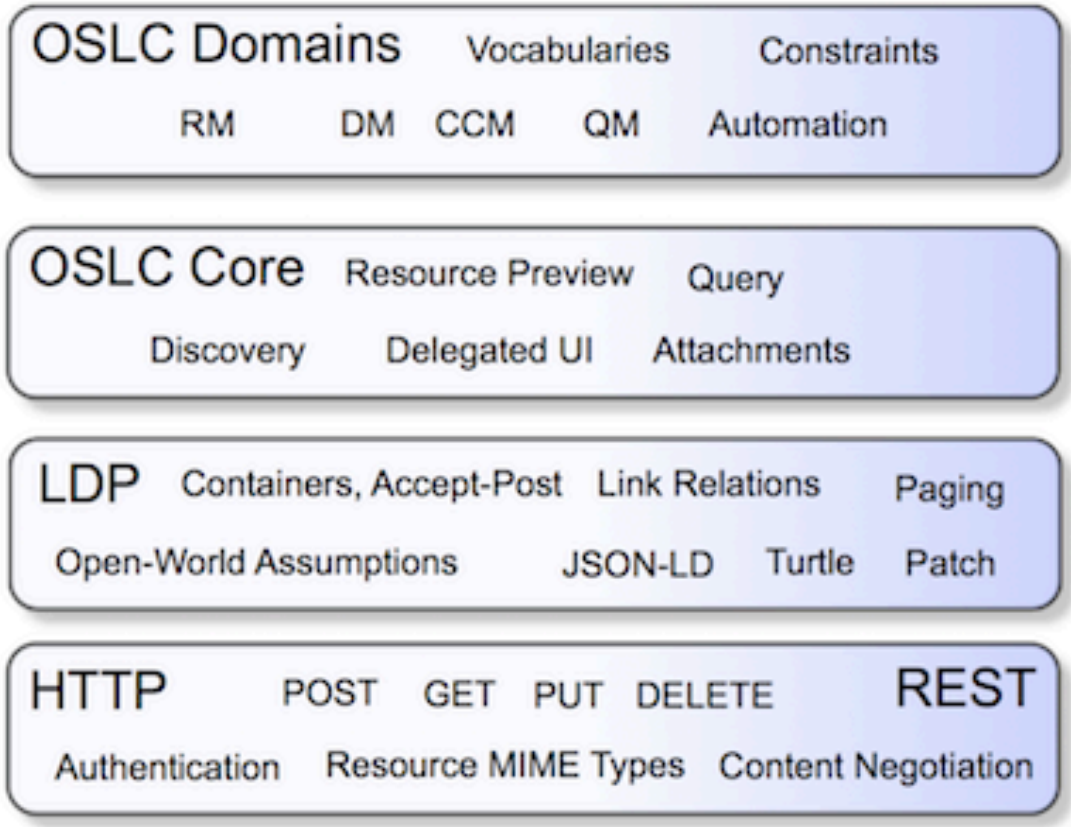
OSLC v3.0: Introduction

Domains of interest that maintain separation of concerns and establish collaborative value streams through integration

Discoverability through Minimal, discoverable, self-describing capabilities to *enable* application integration

Reducing Variability through Self-describing, semantically rich, linked data resources leveraging HATEOAS

Address Complexity through HTTP and REST as the standard mechanism for distributed, loosely coupled APIs



OSLC Change Management 3.0 and OSLC Configuration Management 1.0 Specifications, OASIS

OSLC Core 3.0 Specification, OASIS

LDP 1.0 Specification, LDP.next Working Group, W3C

HTTP 1.1 Specification, IETF

OSLC v3.0: Introduction

The core spec is broken into ~~8~~ 7 parts:

- Overview (This Section)
- Discovery
- Resource Preview (Outside Scope)
- Delegated Dialogs (Outside Scope)
- Attachments (Outside Scope)
- ~~Query~~ (deprecated after 2.0)
- Resource Shape
- Vocabulary

OSLC v3.0: Discovery

A common problem with building interoperable solutions is having a mechanism for a client to explore a server API or end-point to learn if the target application supports a set of capabilities such as:

- How to authenticate
- If the target tool is capable of receiving creation requests
- Whether creation can be handled via a user interface
- What fields are required, with what types of data
- If the record can be pre-filled with some data

New for OSLC v3.0 Builds upon W3C Linked Data Platform standard

https://tools.oasis-open.org/version-control/svn/oslc-core/trunk/specs/discovery.html

OSLC: Discovery

OSLC Discovery 3.0 defines a capability providing client applications a standard way to introspect servers to determine:

- What resource types the server supports
- How to preview, select or create instances of those resources
- Any constraints on resource creation or update.

OSLC: Discovery Approaches

Static Up-Front (Legacy)

- OSLC 2.0 Compatible
- Performed on Startup

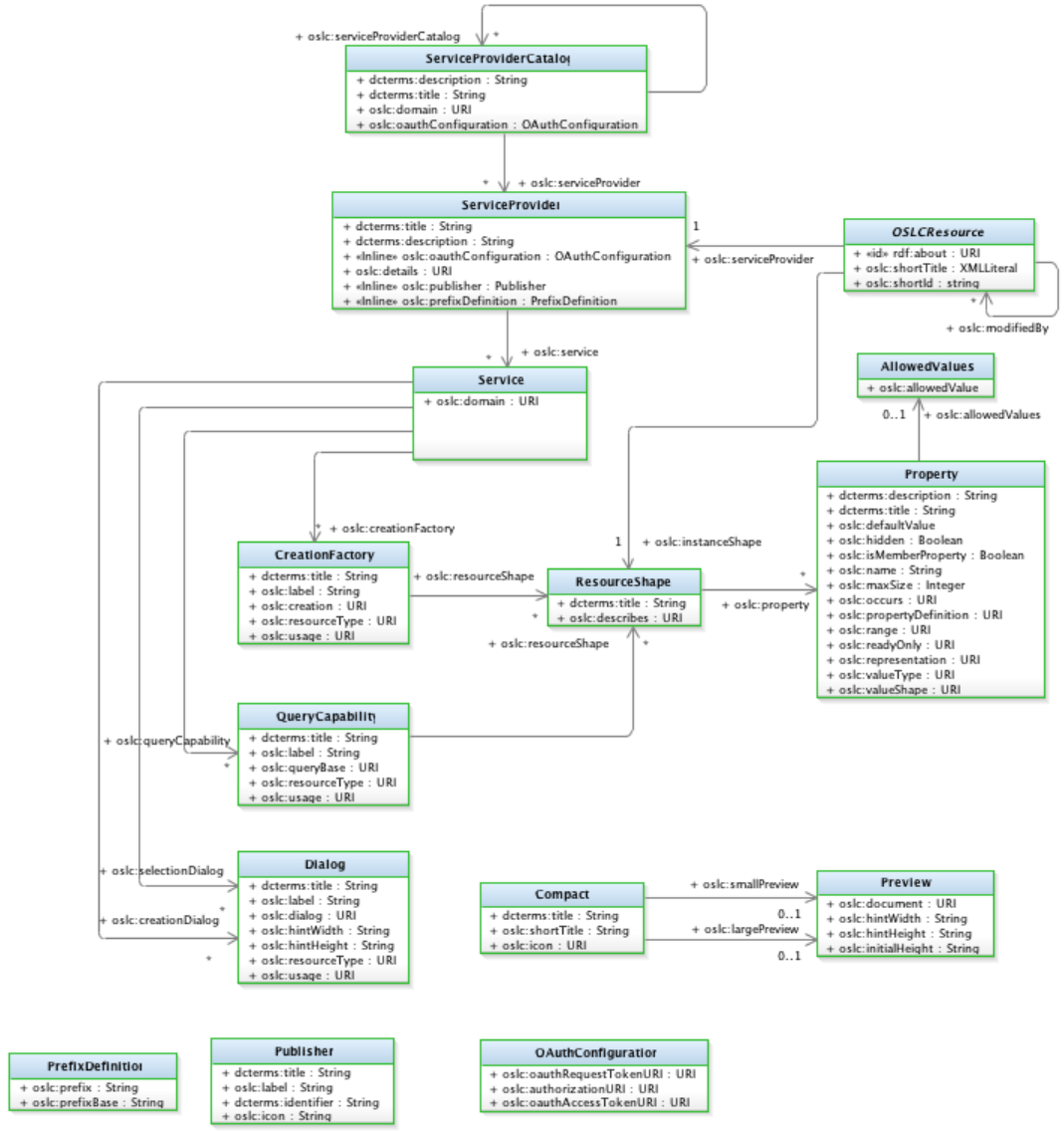
Dynamic Incremental (New)

- Utilizes HTTP OPTIONS or HEAD methods based on LDP
- Most applicable to
  - Rapidly changing services
  - Quick operations that aren't part of a long running communication

OSLC: Discovery Approaches

Both approaches are based on a uniform discovery capability:

- OSLC Service types (from OSLCCore2) are special kinds of LDP Containers
- Allows clients to use either approach to discover server provided services



ASIDE

W3C LINKED DATA PLATFORM V1.0

Aside: W3C Linked Data Platform 1.0

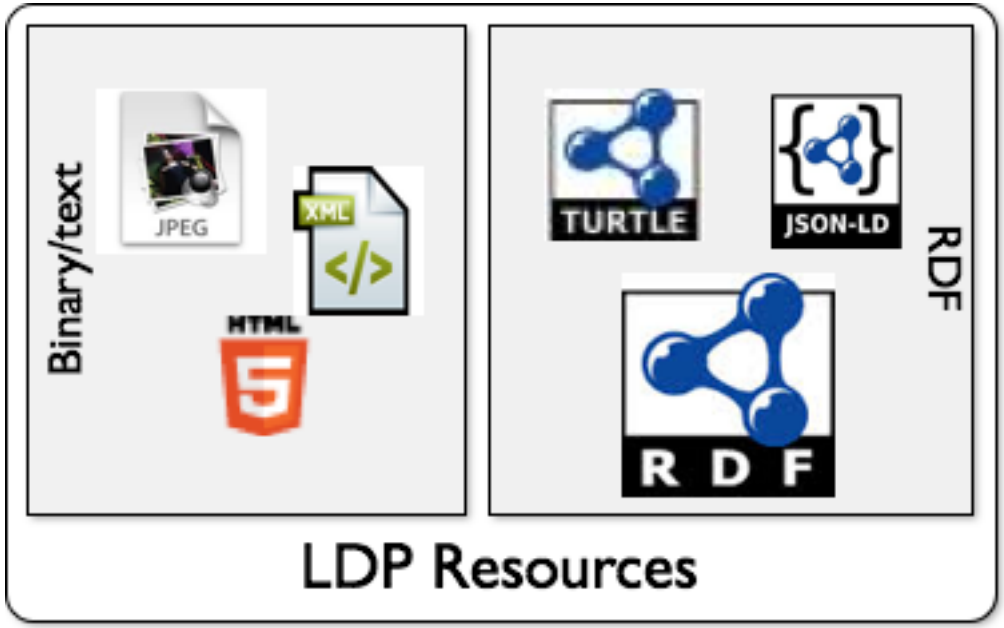
- Describes use of HTTP for CRUD on servers that expose resources as Linked Data.
- Clarifies and expands on the rules of Linked Data
- Discusses standard HTTP and RDF techniques for constructing servers that can CRUD Linked Data Platform Resources (LDPR)
- Defines special kind of LDPR called a Container (LDPC) for building collections of resources
- An Extension to this spec provides ability to break resource representations up into multiple paged responses (LDP-Paging)

Specification: https://www.w3.org/TR/ldp/
Primer: https://www.w3.org/TR/ldp-primer/
LDP-Paging: http://www.w3.org/TR/ldp-paging/

Aside LDP: Linked Data Platform Resources

The rules for LDP Resources address:

- What resource representations should be used?
- How is optimistic collision detection handled for updates?
- What should client expectations be for changes to linked-to resources, such as type changes?
- How can the server make it easy for the client to create resources?
- How do I GET the representation of a large resource broken up into pages?

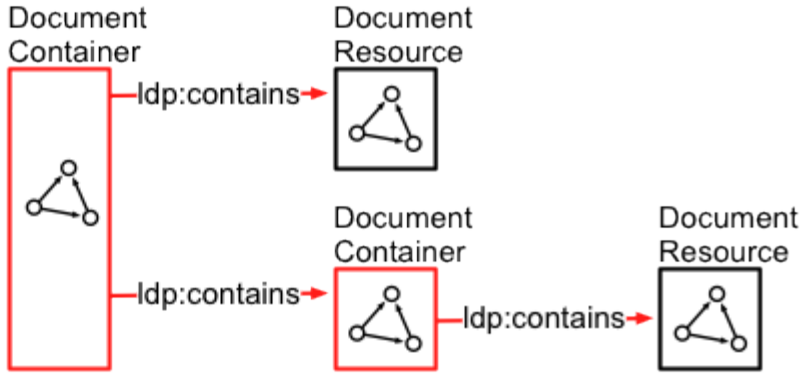


Aside LDP: Linked Data Platform Containers

Containers answer some basic questions, which are:

- To which URLs can I POST to create new resources?
- Where can I GET a list of existing resources?
- How do I get information about the members along with the container?
- How can I ensure the resource data is easy to query?
- How is the order of the container entries expressed?

Example:



Turtle:

```
@prefix ldp: <http://www.w3.org/ns/ldp#> .
   @prefix dcterms: <http://purl.org/dc/terms/> .
   <http://example.org/alice/> a ldp:Container,
   ldp:BasicContainer ; dcterms:title "Alice's
   data storage on the Web"; ldp:contains
   <http://example.org/alice/foaf> ,
   <http://example.org/alice/photos/> .
```

OSLC v3.0: Query

To-Be Deprecated

- *OSLCCore2* introduced a query capability and syntax
  - Intended to be simple
  - Implementable on a wide range of existing server architectures
  - Which ultimately proved to be more difficult than anticipated
- OSLC Core 3.0 is built on *LDP* and future OSLC domain servers may provide more consistent access to SPARQL endpoints.

OSLC v3.0: Resource Shape

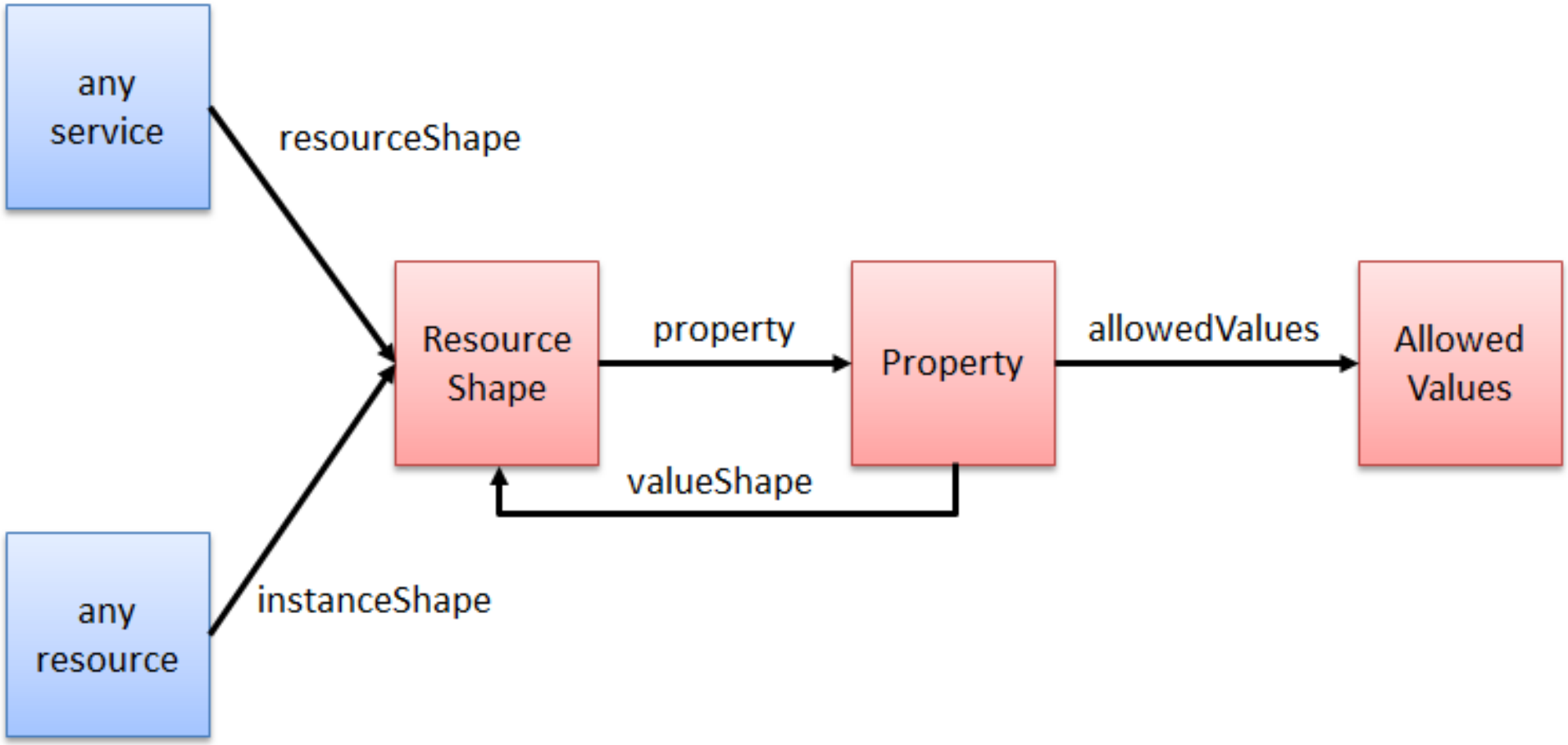
Vocabularies don't define constraints on what can be asserted

- They define the meaning of what was asserted often through the use of reasoners that introduce inferred assertions
- There are also situations where a vocabulary needs to be constrained in order to be used in a specific context for a specific purpose.
- RDFS and OWL are often not useful for this purpose as unless the vocabularies are carefully designed, reasoners can introduce unintended assertions that are not consistent with the specified purpose.

To handle this, OSLC Defines the Resource Shape

- a high-level RDF vocabulary for describing the *shape* of RDF resources.

OSLC: Resource Shape



OSLC: Resource Shape

Resource Shape Definition

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix oslc: <http://open-services.net/ns/core#> .
@prefix oslc_cm: <http://open-services.net/ns/cm#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@base <http://example.com/shape/> .
```

Resource Shape

```
<oslc-change-request> a oslc:ResourceShape ;
        dcterms:title "Creation shape of OSLC Change Request"^^rdf:XMLLiteral ;
        oslc:describes oslc_cm:ChangeRequest ;
        oslc:property
                <oslc-change-request#dcterms-title> ,
                <oslc-change-request#oslc_cm-status> .
```

Property Definitions

```
<oslc-change-request#dcterms-title> a oslc:Property ;
        oslc:propertyDefinition dcterms:title ;
        oslc:name "title" ;
        oslc:occurs oslc:Exactly-one .

<oslc-change-request#oslc_cm-status> a oslc:Property ;
        oslc:propertyDefinition oslc_cm:status ;
        oslc:name "status" ;
        oslc:occurs oslc:Zero-or-one ;
        oslc:allowedValues <status-allowed-values> .
```

Allowed Values Definition

```
@prefix oslc: <http://open-services.net/ns/core#> .
```

Allowed Values

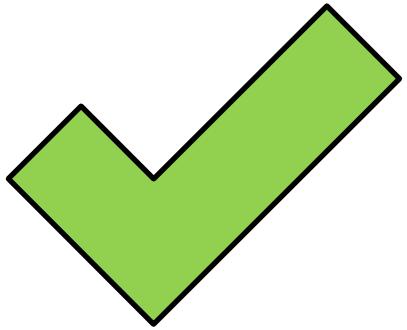
```
<http://example.com/shape/status-allowed-values> a oslc:AllowedValues ;
        oslc:allowedValue "Done" , "InProgress" , "Submitted" .
```

OSLC: Resource Shape

Example 1

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix oslc: <http://open-services.net/ns/core#> .
@prefix oslc_cm: <http://open-services.net/ns/cm#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<http://example.com/bugs/1> a oslc_cm:ChangeRequest ;
          dcterms:title "Null pointer exception in web ui";
          oslc_cm:status "Submitted" ;
          oslc:instanceShape <http://example.com/shape/oslc-change-request> .
```

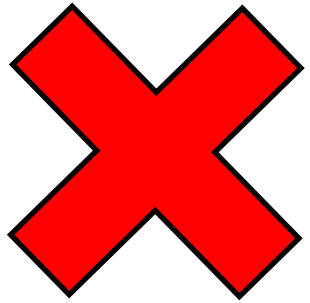


```
<oslc-change-request> a oslc:ResourceShape ;
          dcterms:title "Creation shape of OSLC Change Request"^^rdf:XMLLiteral ;
          oslc:describes oslc_cm:ChangeRequest ;
          oslc:property
                    <oslc-change-request#dcterms-title> ,
                    <oslc-change-request#oslc_cm-status> .

<oslc-change-request#dcterms-title> a oslc:Property ;
          oslc:propertyDefinition dcterms:title ;
                    oslc:name "title" ;
                    oslc:occurs oslc:Exactly-one .

<oslc-change-request#oslc_cm-status> a oslc:Property ;
          oslc:propertyDefinition oslc_cm:status ;
          oslc:name "status" ;
          oslc:occurs oslc:Zero-or-one ;
          oslc:allowedValues <status-allowed-values> .
```

OSLC: Resource Shape



```
<oslc-change-request> a oslc:ResourceShape ;
          dcterms:title "Creation shape of OSLC Change Request"^^rdf:XMLLiteral ;
          oslc:describes oslc_cm:ChangeRequest ;
          oslc:property
                    <oslc-change-request#dcterms-title> ,
                    <oslc-change-request#oslc_cm-status> .

<oslc-change-request#dcterms-title> a oslc:Property ;
          oslc:propertyDefinition dcterms:title ;
                    oslc:name "title" ;
                    oslc:occurs oslc:Exactly-one .

<oslc-change-request#oslc_cm-status> a oslc:Property ;
          oslc:propertyDefinition oslc_cm:status ;
          oslc:name "status" ;
          oslc:occurs oslc:Zero-or-one ;
          oslc:allowedvalues <status-allowed-values> .
```

Example 2

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix oslc: <http://open-services.net/ns/core#> .
@prefix oslc_cm: <http://open-services.net/ns/cm#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

<http://example.com/bugs/2> a oslc_cm:ChangeRequest ;
          dcterms:title "Wrong arguments"^^rdf:XMLLiteral ;
          oslc_cm:status "Submitted", "InProgress" ;
          oslc:instanceShape <http://example.com/shape/oslc-change-request> .
```

OSLC: Vocabulary

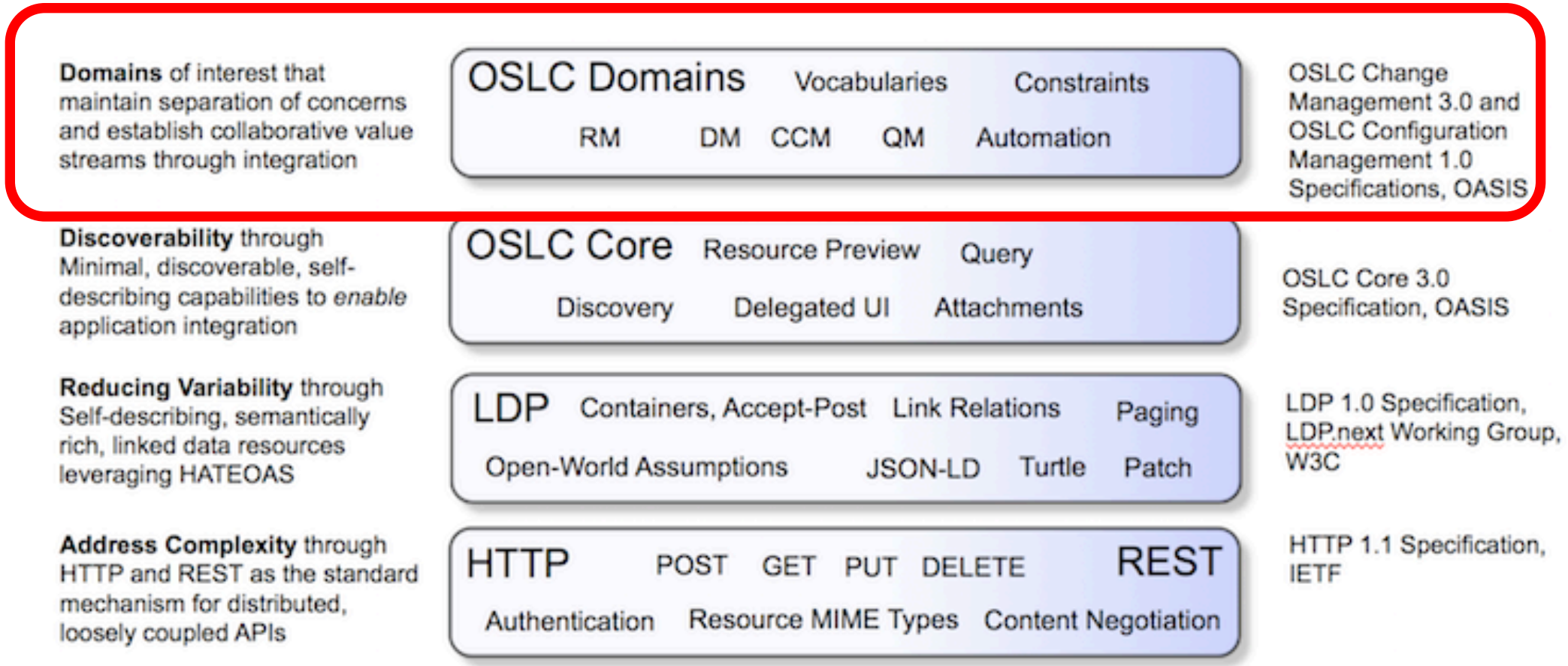
- Vocabulary is largely defined by the sum of the other parts
- There are two additional elements:
  - Error
  - Discussion
- These provide specific terms for sharing commonly defined error and discussion/comment data

SYSML V2.0 SERVICES

SysML v2.0: Services

- Where does SysML fit in?
- What does an O

SysML v2.0: Fit



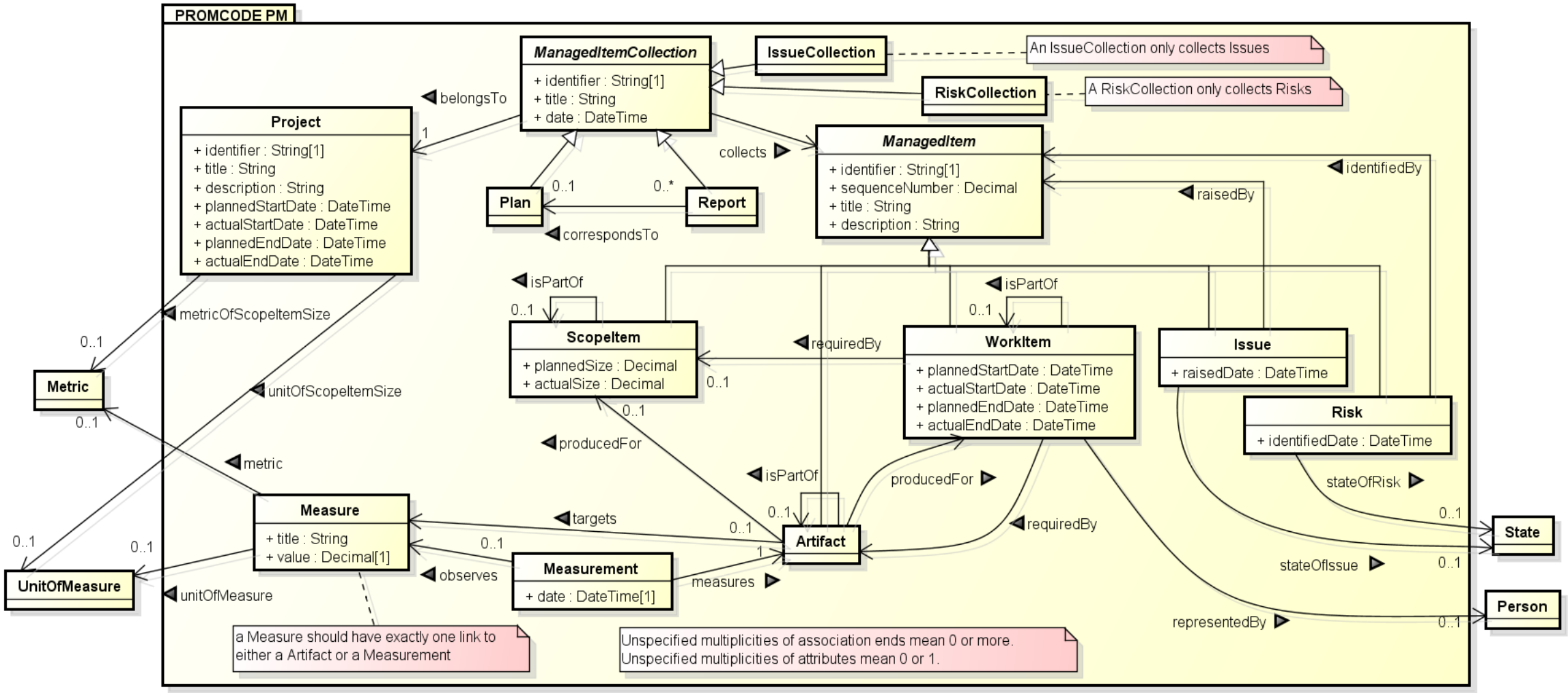
SysML v2.0: Fit Example PROMCODE

PROMCODE (PROject Management of Contracted Delivery)

- An open specification for exchanging project management data across organizational boundaries
- Defines a vocabulary for this exchange based and executed on the OSLC v.3 framework

Source: OASIS OSLC PROMCODE Specification v. 1.0

SysML v2.0: Fit Example PROMCODE



Source: OASIS OSLC PROMCODE Specification v. 1.0

SysML v2.0: Work so far

- Agreed SysML v2.0 interoperability standard should be based on OSLC v.3.0
- Need a philosophy on the set of terms
  - Total SysML->RDF coverage
  - Partial coverage focusing on general modeling needs
  - Inclusion of server specific terms (i.e. version and branch)
- Should we ***require*** certain services to exist in the REST interface?
  - For example element creation (POST) and update (PATCH) which are not required by OSLC or LDP
  - What are these services?
  - Agreement on necessity of those services (MUST vs SHOULD)

SysML v2.0: Work so far, aside

- Need to agree on specification terminology
  - Standards in ISO and OASIS/W3C utilize different specification terminology
  - The key words *MUST*, *MUST NOT*, *REQUIRED*, *SHOULD*, *SHOULD NOT*, *RECOMMENDED*, *MAY*, and *OPTIONAL* are to be interpreted as described in [*RFC2119*]