10 September 2013
W3C RDF Validation Workshop
User Experience

Using SPARQL to Validate Open Annotation RDF Graphs

Anna Gerber (agerber@itee.uq.edu.au)

Tim Cole (t-cole3@Illinois.edu)

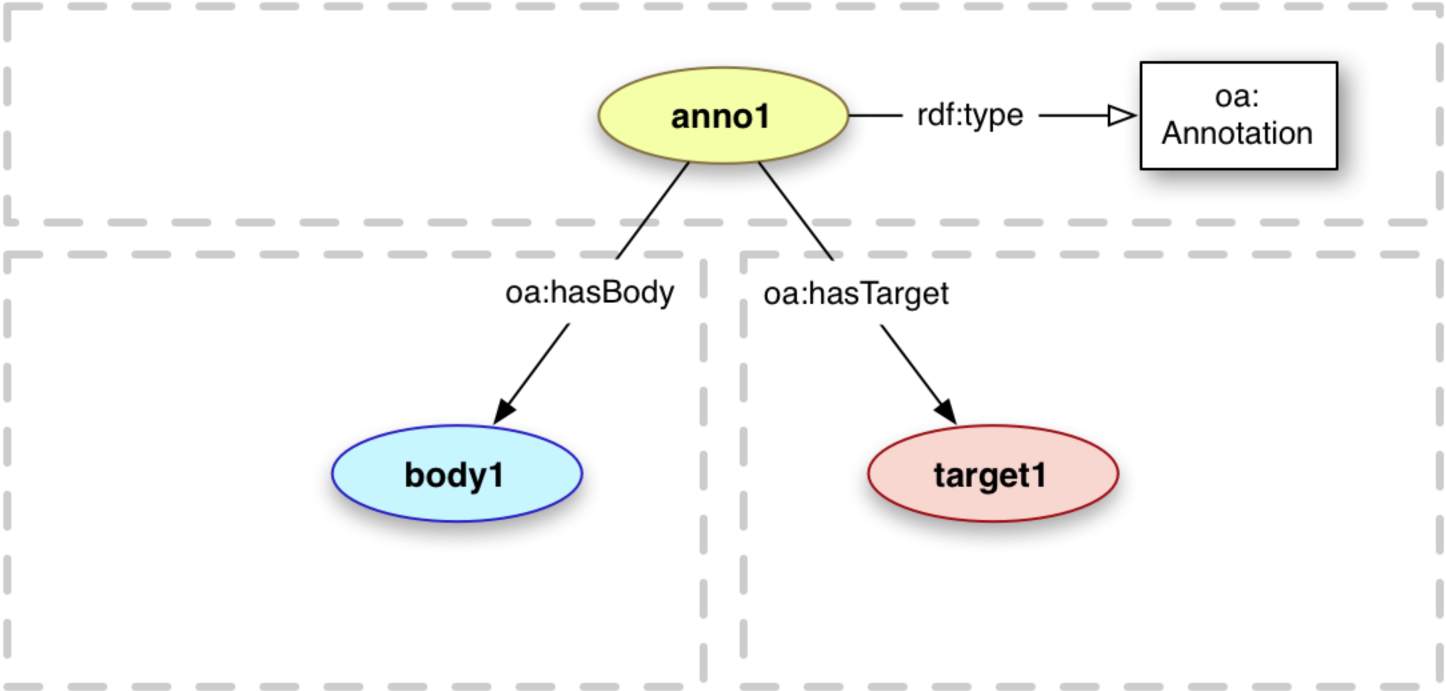
David Lowery (lowery@cs.umb.edu)

W3C Open Annotation Community Group

Context – W3C Open Annotation Community Group

- Founded late 2011 by Open Annotation Collaboration, Annotation Ontology initiative, et al. (currently 100+ members)
- Prime objective: Create a Web & Resource-centric model for describing annotations to facilitate interoperability, annotation sharing, annotations as resources that themselves can be annotated, ...
  - Facilitate tools / services that can span repositories, interoperate, ...
  - Leverage existing models and vocabularies as much as possible
  - Informed by Annotea, etc.
- RDF an obvious choice for the OA model
- 2013: Increasing focus on implementation – validation seen as critical to broad adoption and use

Context – the OA Core Data Model

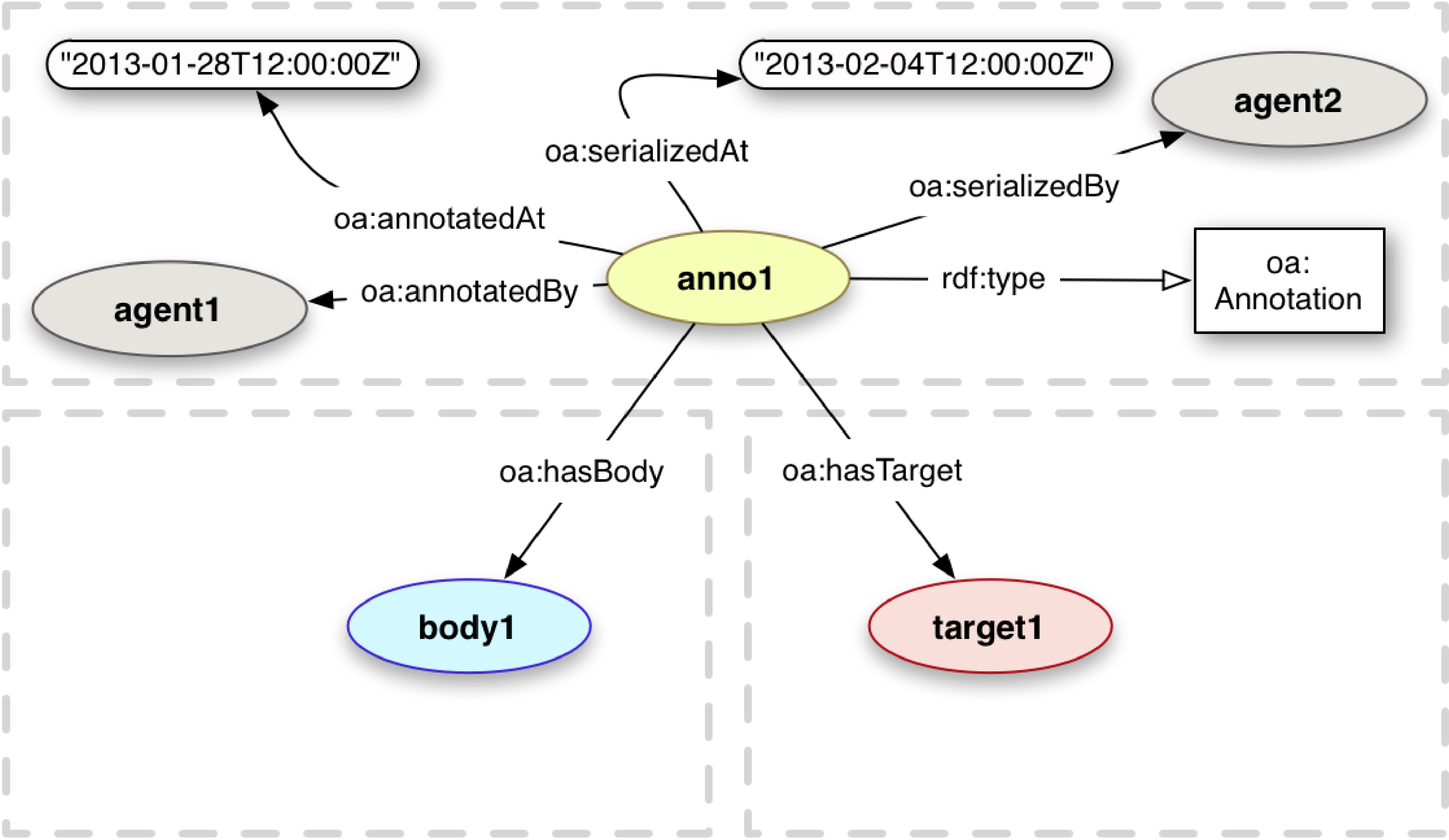


Annotation: The conceptual linkage between body and target

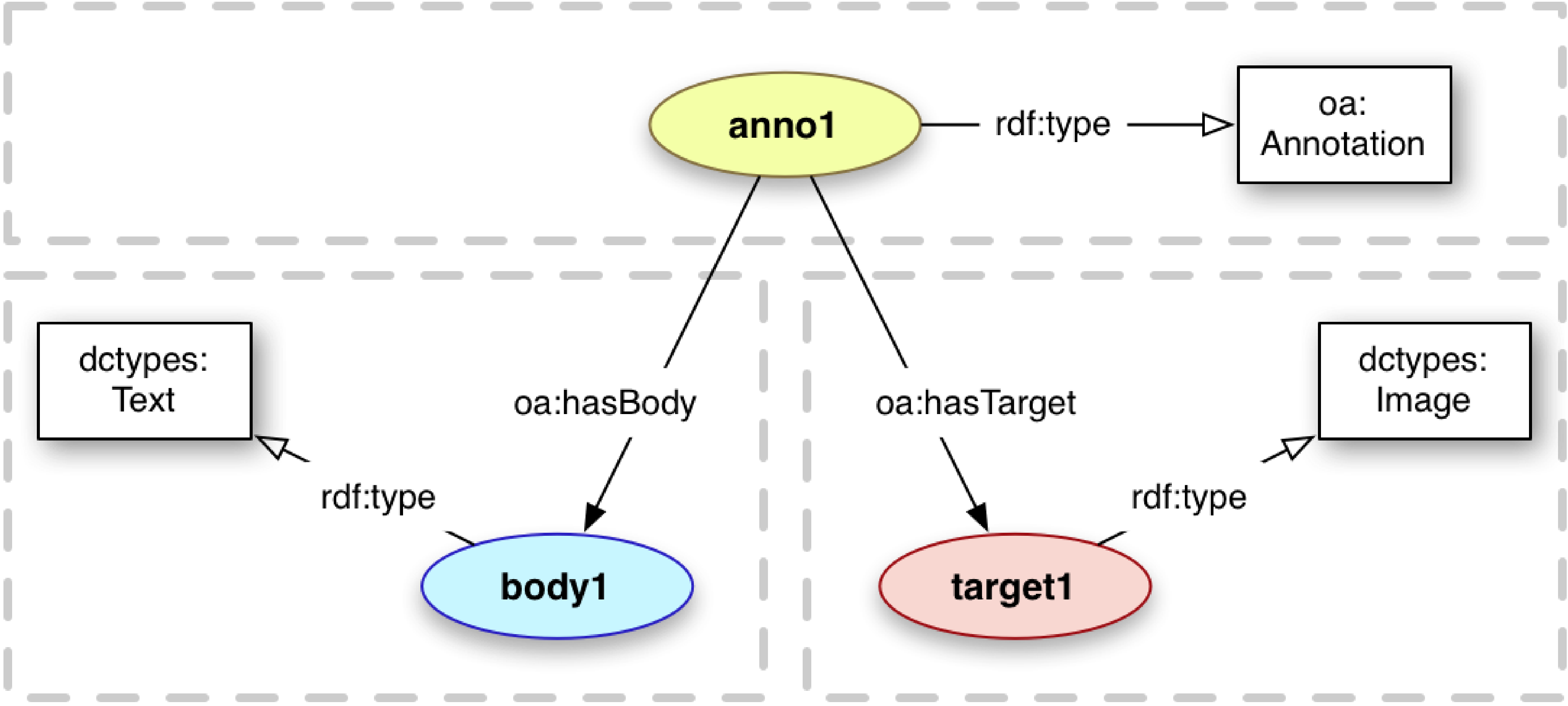
Body: The comment or resource which is “about” the Target

Target: The resource which is being discussed

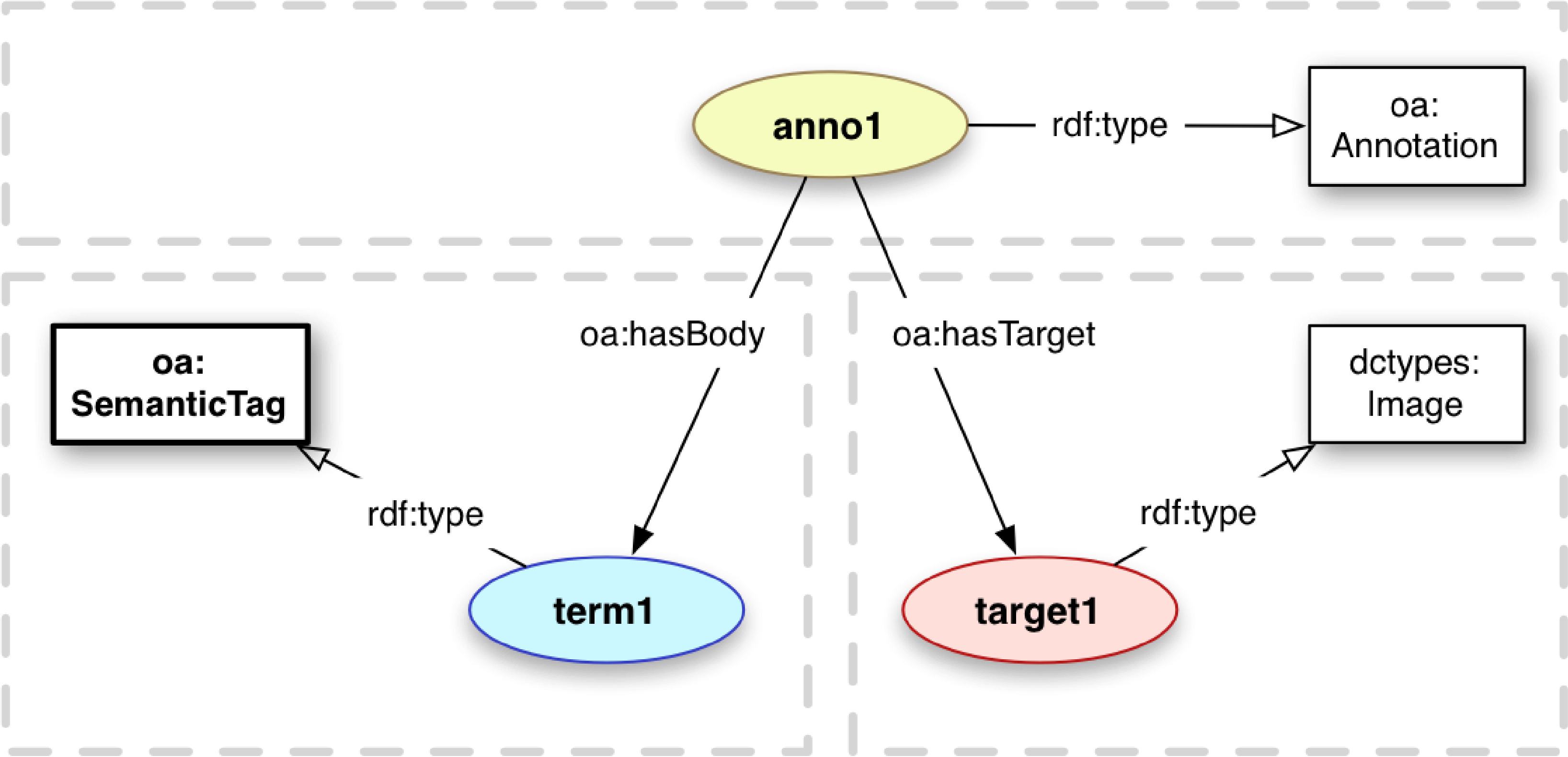
Elaborations & Complexity in the OA Data Model (1)



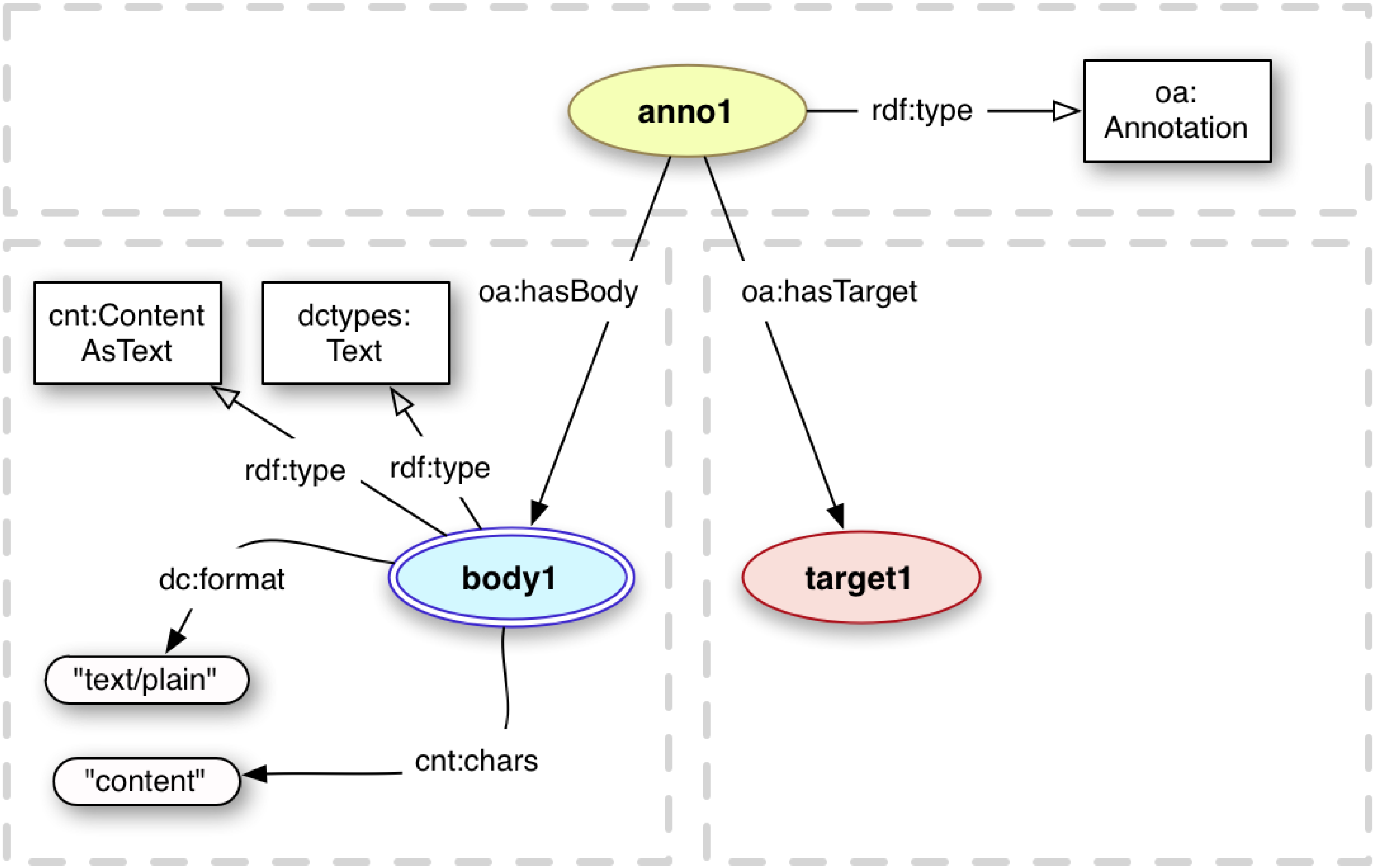
Elaborations & Complexity in the OA Data Model (2)



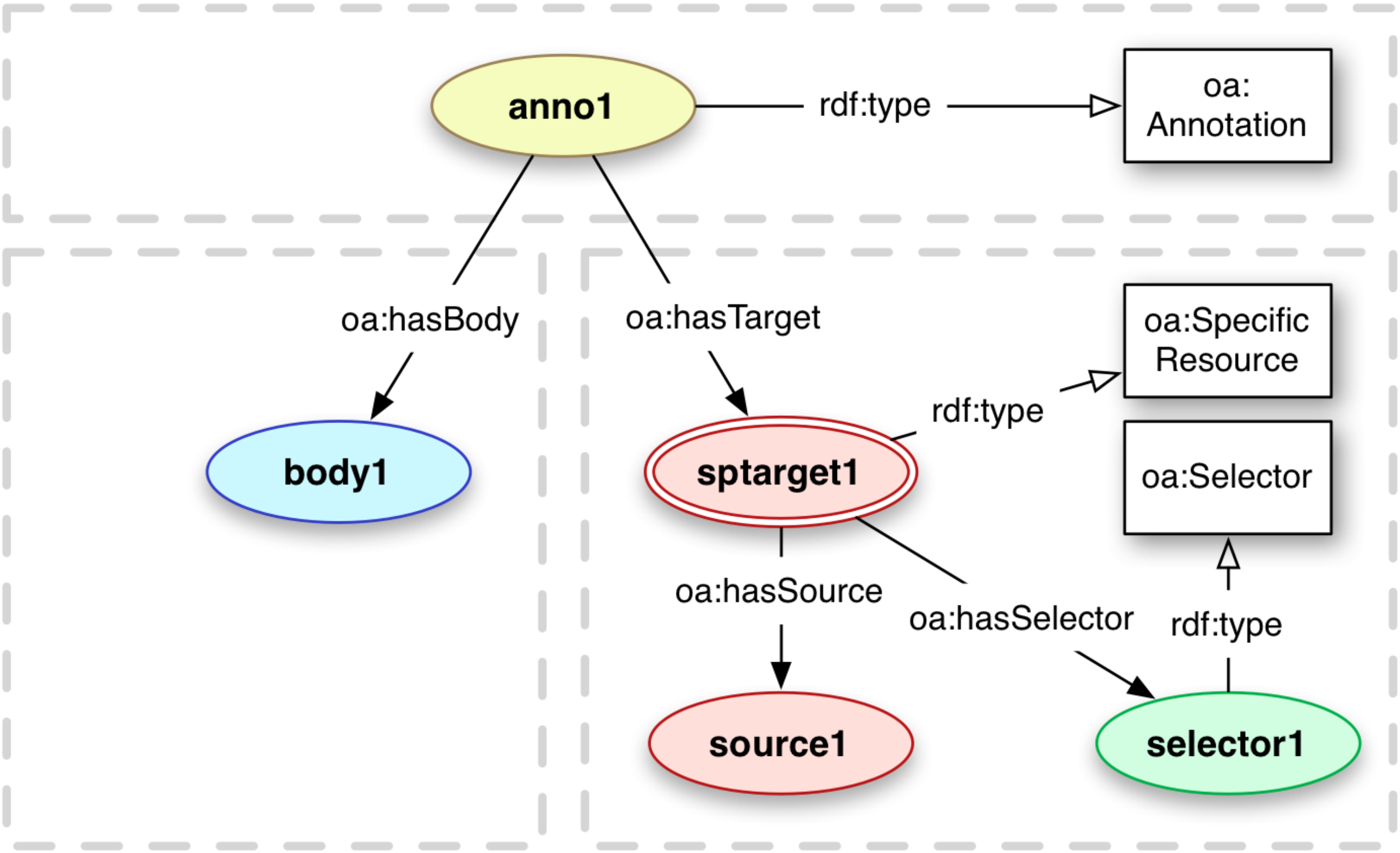
Elaborations & Complexity in the OA Data Model (3)



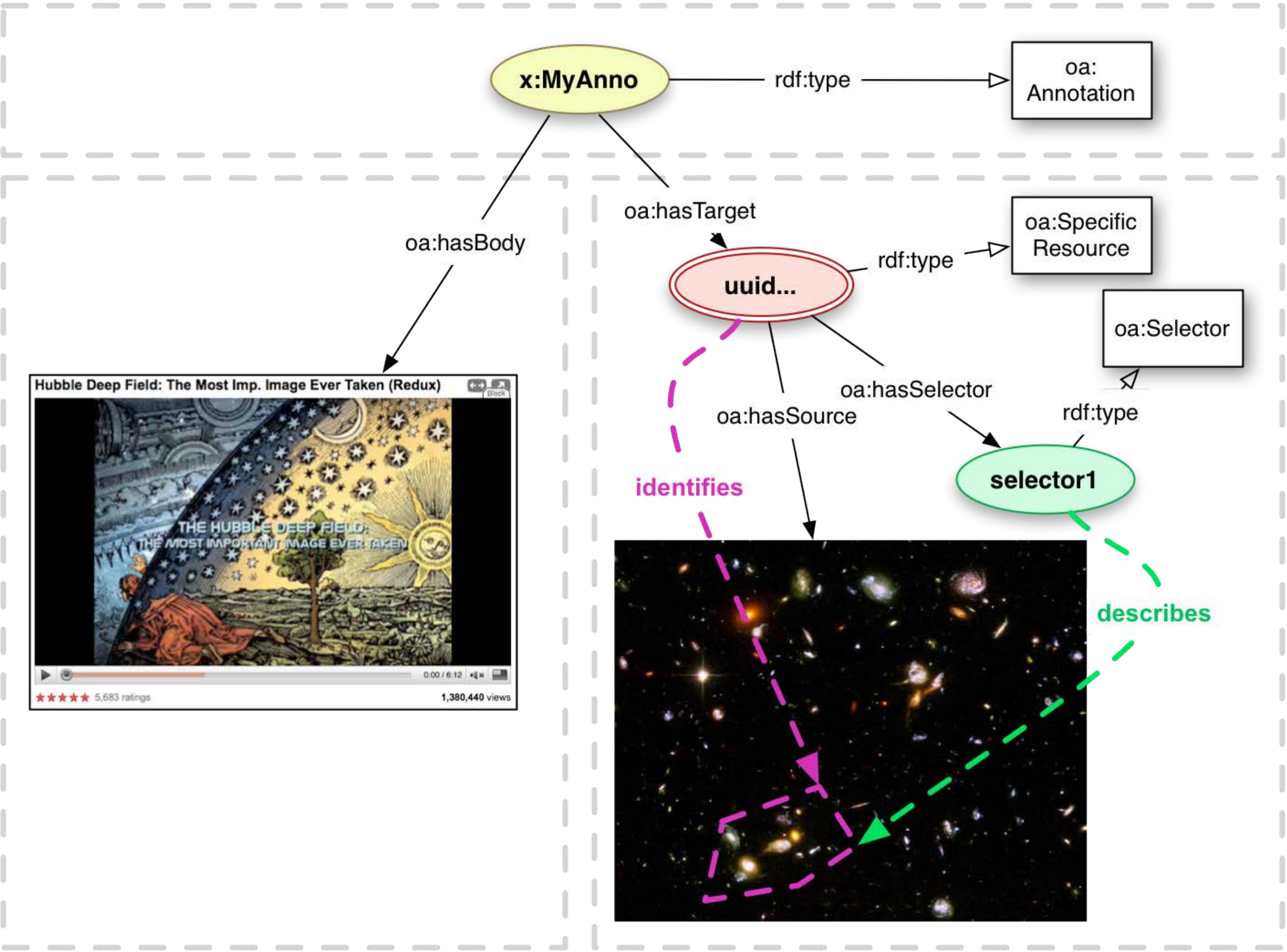
Elaborations & Complexity in the OA Data Model (4)



Elaborations & Complexity in the OA Data Model (5)



Elaborations & Complexity in the OA Data Model



The OA Ontology

Namespace: http://www.w3.org/ns/oa#

Available: http://www.w3.org/ns/oa.rdf, http://www.w3.org/ns/oa.ttl, …

- 19 Classes
- 23 Properties
- References RDFS, the SKOS core, and W3C PROV

- Some classes & properties required, some recommended, some optional
- Meant to be easily extensible....
- OA OWL specification is incomplete – i.e., some constraints (e.g., cardinality) are only expressed in human-readable specification: http://www.openannotation.org/spec/core/

LoreStore Annotation Repository

Application to store, search, query, display and validate annotations.

•Queensland / AustESE implementation available at:
http://austese.net/lorestore/
http://austese.net/lorestore/validate.html

•Can be deployed locally from github:
https://github.com/uq-eresearch/lorestore
Dependencies:
Apache Tomcat
MySQL (expects specific database & db user)

•Validation functionality available through RESTful API

The approach we are using to validate OA RDF

1. Identify constraints, e.g., as expressed in OA ontology & OA data model spec
2. Categorize as warning or error (Should/Recommended vs. Must)
3. Check for conformance using pairs of SPARQL queries:
   - Precondition query – does constraint apply to this annotation description? {yes | no}
   - Primary query – if yes, is constraint satisified? {yes | no}

4. As applicable, result of precondition check is displayed.
5. As applicable, warning or error message is displayed, along with link to part of data model specification expressing constraint
6. Current list of ~55 SPARQL queries used for generic OA Validation: https://github.com/uq-eresearch/lorestore/blob/master/src/main/resources/OAConstraintsSPARQL.json

Illustration 1: Exactly 1 node must be type oa:Annotation

```
{
    "ref": "2.1.0. (2) Body and Target Resources",
    "url": "http://www.openannotation.org/spec/core/core.html#BodyTarget",
    "description": "The oa:Annotation class MUST be associated with each Annotation.",
    "severity": "error",
    "preconditionMessage": "No Annotations identified",
    "precondition": "PREFIX oa: <http://www.w3.org/ns/oa#> ASK WHERE {{?annotation oa:hasTarget ?t}UNION {?annotation a oa:Annotation}}",
    "query": "PREFIX oa: <http://www.w3.org/ns/oa#> SELECT ?annotation WHERE {?annotation oa:hasTarget ?t . FILTER(NOT EXISTS { ?annotation a oa:Annotation })}"
  }
```

Illustration 2: should use dc:type to describe body/target nodes

```
{
        "ref": "2.1.1. (2) Typing of Body and Target",
        "url": "http://www.openannotation.org/spec/core/core.html#BodyTargetType",
        "description": "The Dublin Core Types vocabulary is RECOMMENDED.",
        "severity": "warn",
        "preconditionMessage": "No body or target present",
        "precondition": "PREFIX oa: <http://www.w3.org/ns/oa#> ASK WHERE { {?annotation oa:hasTarget
?resource} UNION {?annotation oa:hasBody ?resource} }",
        "query": "PREFIX oa: <http://www.w3.org/ns/oa#> SELECT ?resource ?type WHERE { {?annotation
oa:hasTarget ?resource . FILTER(NOT EXISTS{?resource oa:hasSource ?i})} UNION {?annotation
oa:hasBody ?resource . FILTER(NOT EXISTS{?resource oa:hasSource ?i})} UNION {?annotation
oa:hasTarget ?i . ?i oa:hasSource ?resource} UNION {?annotation oa:hasBody ?i . ?i oa:hasSource
?resource} .  FILTER NOT EXISTS{{?resource a ?type . FILTER
regex(str(?type),\"^http://purl.org/dc/dcmitype/\")}UNION{?resource a ?type . FILTER regex(str(?type),
\"http://www.w3.org/ns/oa#\")}}}"
    }
```

Illustration 3: hasSource cardinality (exactly 1)

```
{
	"ref": "3.1.0. (2) Specifiers and Specific Resources",
	"url": "http://www.openannotation.org/spec/core/specific.html#Specific",
	"description": "There MUST be exactly 1 oa:hasSource relationship associated with a Specific Resource.",
	"severity": "error",
	"preconditionMessage": "No SpecificResources identified",
	"precondition": "PREFIX oa: <http://www.w3.org/ns/oa#> ASK WHERE { {?res oa:hasSource ?source } UNION {?res a oa:SpecificResource}}",
	"query": "PREFIX oa: <http://www.w3.org/ns/oa#> SELECT ?res WHERE { {?res oa:hasSource ?source } UNION {?res a oa:SpecificResource} . OPTIONAL{?res oa:hasSource ?source}} group by ?res having(count(distinct ?source) != 1)"
    }
```

Illustration 4: hasState cardinality (0 or 1)

```
{
        "ref": "3.3.0. (1) States",
        "url": "http://www.openannotation.org/spec/core/specific.html#States",
        "description": "There MAY be 0 or 1 oa:hasState relationship for each SpecificResource.",
        "severity": "error",
        "preconditionMessage": "No SpecificResources identified",
        "precondition": "PREFIX oa: <http://www.w3.org/ns/oa#> ASK WHERE { {?res oa:hasSource ?source } UNION {?res a oa:SpecificResource}}",
        "query": "PREFIX oa: <http://www.w3.org/ns/oa#> SELECT ?res WHERE { ?res oa:hasState ?state } group by ?res having (count(distinct ?state) > 1)"
    }
```

Needs Illustration (1) – Equivalent of XML Schema Choice

- An oa:SpecificResource identifies a new resource derived from an existing resource (associated with oa:SpecificResource using oa:hasSource)
  - Each oa:SpecificResource must be the subject of exactly 1 oa:hasSource predicate
  - Each oa:SpecificResource must be the subject of at least 1 ‘has Specifier’ predicate
  - Specifier is in essence the union of oa:Selector, oa:State and oa:Scope classes but Specifier not defined in OA ontology & not meant to be used in instances
  - oa:hasSelector, oa:hasState, oa:hasScope have ranges of oa:Selector ... oa:Scope and each has individual cardinality constraints (generally 0 or 1)
  - How best to express this kind of constraint? E.g., DC Application Profile, etc.
  - Currently OA Validator requires exactly 1 oa:hasSelector

Needs Illustration (2) – Validate But Allow Extensibility

- For example, the OA Ontology defines several Selector classes (but we can assume more will be needed):
  - oa:DataPositionSelector
  - oa:FragmentSelector
  - oa:SvgSelector
  - oa:TextPositionSelector
  - oa: TextQuoteSelector
- OA Ontology defines range of oa:hasSelector as oa:Selector, so each of these are defined as subclasses of oa:Selector & we test for oa:hasSelector
  - Some subclasses bring additional constraints, e.g., oa:TextQuoteSelector must be subject of exactly 1 oa:exact predicate.
- Need validation approach that easily supports extensibility as community extends with different kinds of Selectors.

Needs Illustration (3) – must vs. should/recommend constraints

- OA specifications uses *Must, Should, Recommend, May, Optional, ...*
- Useful to provide 2 levels of feedback, e.g., *error* vs. *warning*
- Must have

```
<anno1> a oa:Annotation ;
      oa:hasTarget <target1> .
```

- Recommended that you have

```
<anno1> a oa:Annotation ;
      oa:hasTarget <target1> .

 <target1> a dctypes:Image .
```

Needs Illustration (4) – validation dependencies on values

- Most of the core OA constraints are relatively straightforward
  - Require one resource that is typed as oa:Annotation
  - Cardinality of oa:hasTarget
  - oa:SpecificResource implies exactly 1 oa:hasSource
  - Can’t have oa:hasScope without an oa:SpecificResource
  - ....
- Communities are identifying more complex constraints based on values
- For example in FilteredPush annotation application, only certain combinations of Body type values and Expectation values are allowed

Overview of FilteredPush (FP) RDF Validation

- FP focus is on annotation of data at the record level and below
  - Datasets often have URIs, records rarely do. oa:Selectors matter!
  - FP defines some Selectors based on data queries of several (SQL, KVPair, Xpath…)
  - (Data are natural science collection specimen metadata, as many as 3.5Bn)
- Annotations parsed and interpreted usually only if valid both for OA and domain vocabulary annotation content.
  - OA validity generally stable due to annotation generation application
  - OA content (Target, Body, …) more volatile hence(?) validation is more critical
- Validation preconditions; grouping of rules into rulesets (for common pass or fail); error information...
  - Presently configured by an XML Schema
  - Could/should/will use JSON to live happily with LoreStore OA validator, probably as a Java library.

Fragment of Body type dependancy SPARQL Rule

```
#     Return target and body for valid Annotations
SELECT ?target ?body WHERE { ?anno a oa:Annotation . ?anno oa:hasBody ?body . anno oa:hasTarget ?target .
      MINUS {
#     Annotation with dwcFP:Identification oa:Body is valid under this rule only when oad:Expectation is
#        oad:Expectation_Update or oad:Expecation_Insert (and several predicate values obtain)
               {
                ?body a dwcFP:Identification .
                ?anno oad:hasExpectation ?exp .
                { ?exp a oad:Expectation_Update } UNION { ?exp a oad:Expectation_Insert } .

                  ?target a oa:SpecificResource .
                  ?target oa:hasSelector ?selector .
                  OPTIONAL { ?selector dwc:occurrenceID ?occurrenceId } .
                 …
                   FILTER ( # Pass as valid only those having particular domain predicates  bound
                  …)
                  }
                }
    ….  # UNION of four more similar conditions on oa:Body rdf:type in domain ontology;
```