Data Modeling 101

What, How, and Why it’s more than PCDM

Your Fearless Facilitators

Julie

Metadata Analyst at Indiana University, @jlhardes

Christina

Works on Metadata at Cornell, @cm_harlow

Tom

Ph.D. Student at University of Wash. iSchool; DCE, @no_reply

Mark

Collaboration & Interoperability Architect, Stanford @anarchivist

bit.ly/HC16DataModels

Link for Google Drive Folder with Slides, Notes, Examples, Resources, & Other Workshop Materials

Communication Channels

- Alert a facilitator if you need help or have questions
- Project Hydra Slack: **#hc2016-datamodeling** channel
  - https://project-hydra.slack.com/
  - Sign up for an invitation here: http://slack.projecthydra.org/

Schedule

9-9:15	Introduction
9:15-10:15	Intro to Data Models: How Models Work w/Systems
10:15-10:30	*15 minute break*
10:30-10:45	Intro to Data Models: Expressing Models
10:45-11	Breakout groups: Intro & Setup
11-11:45	Breakout groups: Data Models Breakouts
11:45-Noon	*15 minute break*
Noon-12:15	Breakout groups: Recaps
12:15-12:30	Going Forward with Data Modeling & Hydra

Our Expectations of You

- Follow the Project Hydra Code of Conduct
- Follow the Recurse Center Social Rules (a.k.a. "Hacker School Rules")
- Be ready to learn about data modeling!

Project Hydra Code of Conduct

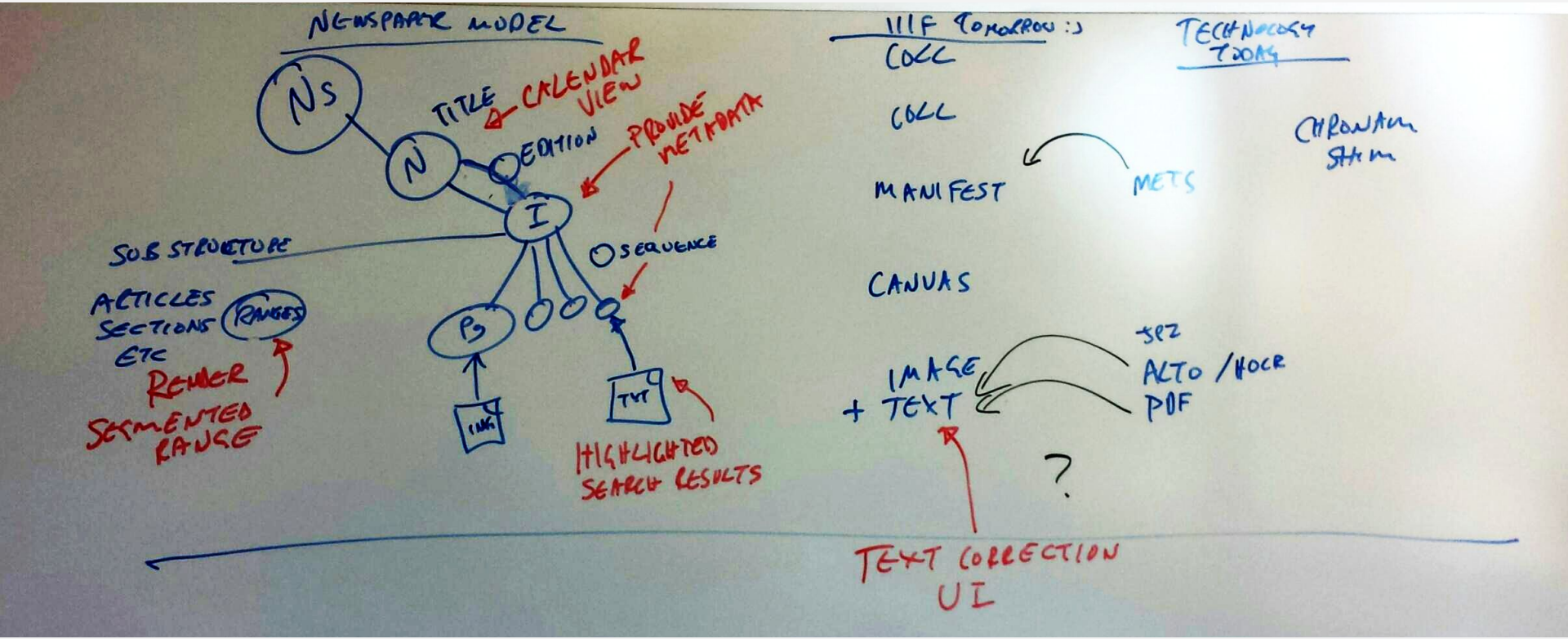
https://wiki.duraspace.org/x/GJsOB

Recurse Center Social Rules
(a.k.a. Hacker School Rules)

https://www.recurse.com/manual#sub-sec-social-rules

- No feigning surprise
- No well-actually's
- No back-seat driving
- No subtle -isms
  - More info: https://www.recurse.com/blog/38-subtle-isms-at-hacker-school

Are you ready to data model?



Reminder

This is an informal workshop - ask questions!

This workshop is an attempt to

- Help build data modeling capacity in the Hydra (and broader) communities
- Help build best practices for data models

Our goals for this workshop

- Share motivations for data modeling as part of the application development process
- Equip you with knowledge needed to instigate modeling work at your institutions and participate in broader community discussions
- Demonstrate modeling practices and pitfalls
- Give context for data modeling, standards, and interoperability work in Hydra and related communities

However, this workshop is not ...

- A linked data or RDF workshop
- A metadata standards workshop
- A PCDM workshop

... but options exist for learning more about these topics!

Now: your goals for this workshop?

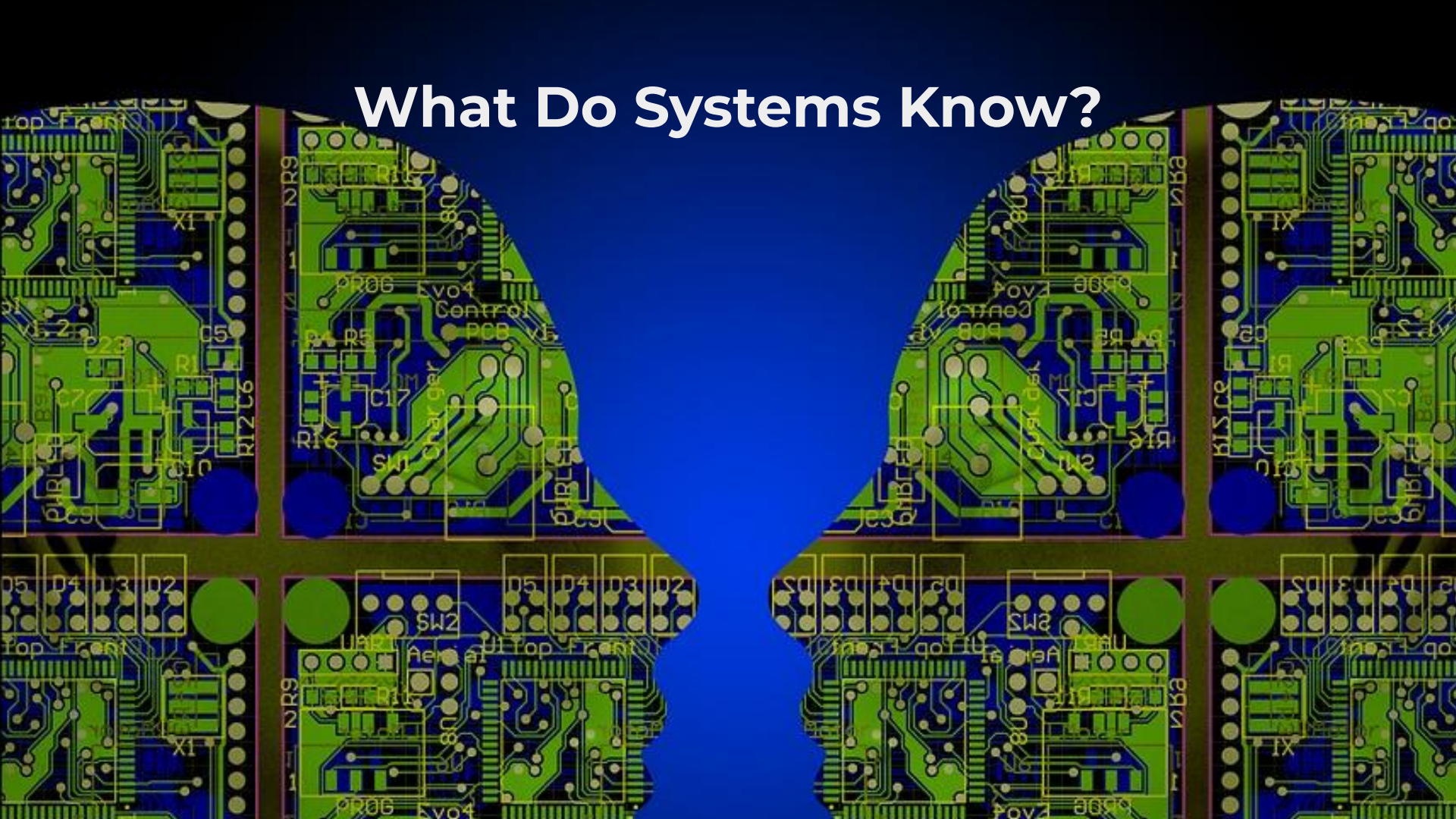
- Why are you attending this workshop?
- What are your goals - immediate or long-term?
- What's your level of comfort and experience with data modeling?

We want to capture your goals, return to them throughout the workshop & going forward - feel free to add to responses to the shared notes.

Introduction to Data Modeling

What is it & why do it?

What Do Systems Know?



What Does a Fax Machine Know?

Questions for Modelers

- What things/kinds of thing does the system work with?
- What functions/actions does it perform?
- What does it remember?
- What does it need to be able to tell its users?
- How do the answers change as the system does its work?

Questions for Modelers

What is the domain?

Where is the Knowledge?

- Database & System State
  - Long term storage, in-memory representations.
- Code
  - OO Models, other code that handles data.
- Serializations
  - Output for interchange, etc…
- Design Documents
  - Diagrams, Descriptions, UML, etc…

Where is the Knowledge?

‘Every information system embodies a conceptual schema. Without a conceptual schema, a system could not perform any useful function… *The only available options are to explicitly define the schema or to have it in the minds of the designers.*’

– Conceptual Modeling of Information Systems. p. 21

Defining “Data Model”

Abstract Syntax

Data Type

Captures structure outside the context of a representation.

E.g. RDF “Data Model”:

‘This document defines an abstract syntax (a data model) which serves to link all RDF-based languages and specifications. The abstract syntax has two key data structures: RDF graphs are sets of subject-predicate-object triples, where the elements may be IRIs, blank nodes, or datatyped literals.’

– RDF Concepts & Abstract Syntax (abstract)

Physical Model

Data Structure

A representation of data in concrete form.

- A serialized document;
- A database schema;
- A data structure (array, hashmap, B-tree, etc…).

Conceptual Model

Logical Model

Captures the concepts and relationships of the data. Higher-level than an abstract syntax; instead: An abstract *schema*.

The Three-Schema Approach:

External ↤ Conceptual ↦ Internal

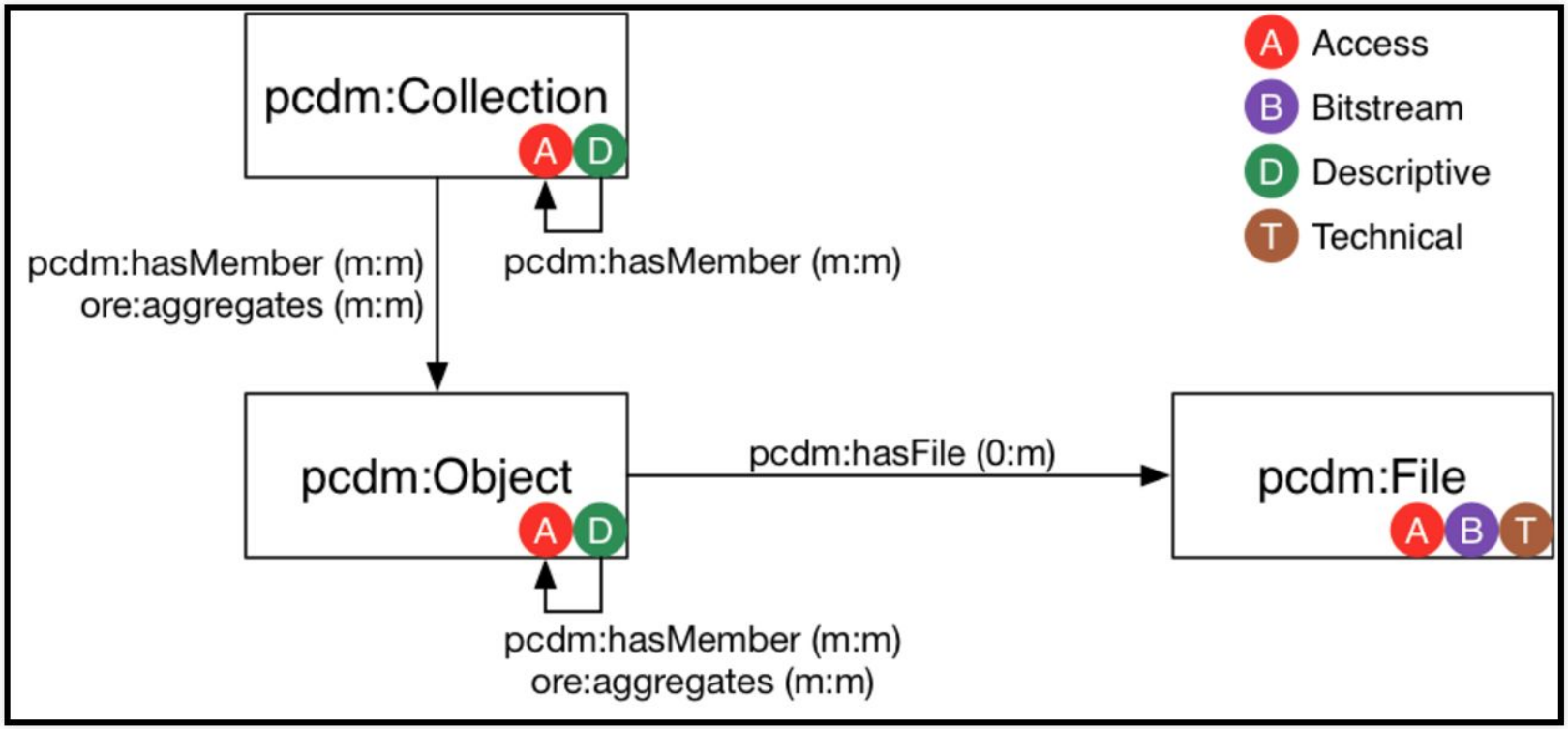
RDFS, SKOS, PCDM?

Low-Level Model / Upper Ontology

A model that can be used to describe other, higher level models. An Ontology that presents a view of the world common to all Ontologies.

‘A metaschema is a schema that represents general knowledge about a domain that consists of a schema.’

– Conceptual Modeling of Information Systems. p. 400



github.com/duraspace/pcdm/wiki

Motivations for Modeling

Model as an Application Artifact

- The model acts as a specification of the Information System.
- Model documents serve as documentation for the design team, developers, and users.

Modeling for Design Feedback

- Provides common language and point of discussion for programmers, designers & domain experts.
- Early opportunity to address ethical issues:
  - How does what we model in our systems create affordances and restrictions for our users?
  - Falsehoods Programmers Believe

Dissemination, Interchange, Interoperability

- “The Informative Function” (Olivé)
  - Display to user as map to a View Schema.
- Interchange
  - Map to External Schema for publication or API functions.
- Interoperability
  - With systems that share a model or metamodel;
  - Transformations/crosswalks to systems that operate in a different domain or use a different model.

Consistency, Validation, Workflow

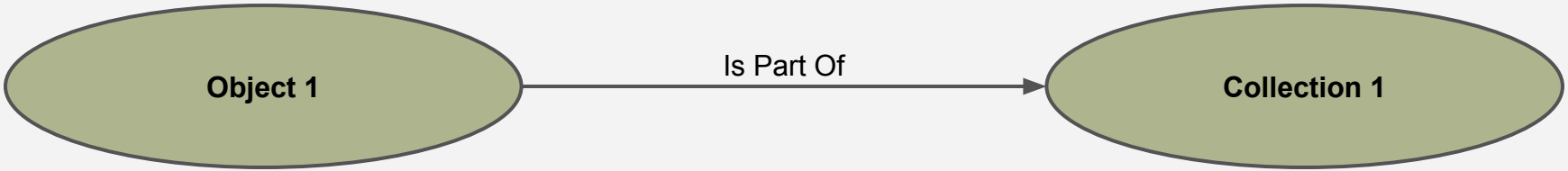
- Database Constraints
  - Cardinality
  - Value Constraints (*Consistency* in database theory)
- Validation Throughout a Lifecycle
  - Pre/Post-condition
- Defined Workflows
  - Events, Actions, State Transitions

One Approach:
Semantic Modeling

Objects & Relations

- One kind of “Object” (Entity, Resource)
  - Objects are distinct from their names & from their records.
- Binary Relations between objects
  - Interpreted as sentences: (person isAuthorOf book)
  - And as *facts*.
- Group objects into Classes (or Entity Types)
  - (person isA Person)

Objects & Relations



Types of Relations

- Instances of a Relation Class
- Domain & Range
  - Not a constraint!
- Common relation types:
  - Partitive (*chapter isPartOf book*), (*book hasPart chapter*)
  - Membership (*book isMemberOf collection*)
  - Materialization (*book isPortrayalOf work*)
  - Role
    - Type (person isA author)
    - Surrogate (author isRoleOf person, book hasAuthor author)

Subsumption

- Class
  - If **A** is a subclass of **B** and *x* is a member of **A**, *x* is a member of **B**.
  - *cf.* Liskov Substitution Principle in OO
  - E.g. Mammal subClassOf Animal
- Property
  - If **P** is a subproperty of **P′** and *x* **P** *y*, then *x* **P′** *y*.
  - E.g. hasMother subPropertyOf hasParent

RDFS

Construct	Syntactic form	Description
Class (a class)	**C** rdf:type rdfs:Class	**C** (a resource) is an RDF class
Property (a class)	**P** rdf:type rdf:Property	**P** (a resource) is an RDF property
type (a property)	**I** rdf:type **C**	**I** (a resource) is an instance of **C** (a class)
subClassOf (a property)	**C1** rdfs:subClassOf **C2**	**C1** (a class) is a subclass of **C2** (a class)
subPropertyOf (a property)	**P1** rdfs:subPropertyOf **P2**	**P1** (a property) is a sub-property of **P2** (a property)
domain (a property)	**P** rdfs:domain **C**	domain of **P** (a property) is **C** (a class)
range (a property)	**P** rdfs:range **C**	range of **P** (a property) is **C** (a class)

Our Semantic Modeling Bias: Other Approaches

- Hierarchical
- Network (see: *Programmer as Navigator*, Bachmann)
- Relational
- Object Oriented

More abstractly:

- Entity-Relationship
- Object-Role
- Definitional Theory

Our Semantic Modeling Bias: What Gives?

- Avoid “Database Orientation”
  - or language specific “data structure orientation”.
  - or “document model orientation”.
- Focus on Interoperability
  - Semantic Modeling = data is self-describing in a known semantics. (wat?!)
- Simplicity
  - model reduces to a single kind of object and binary relations.
  - “Simple” not to be confused with “easy”.

15 Minute Break
Reconvene at 10:30 AM

Modeling in (a) Context

What does Data Modeling as described look like?

‘Unfortunately, these disciplines have not yet reached agreement regarding the terminology, concepts, and mechanisms that we use to distinguish between the objects and relationships in a domain. Consequently, we do not have a solid theoretical basis on which to base our study, and, as is often the case **when discussion information systems, we must adopt a humble and eclectic attitude**.’

- Conceptual Modeling of Information Systems. p. 10

Communication & Data Modeling

“humble & eclectic attitude”

Photo from http://observer.com/2013/03/dj-spooky-spins-off-from-90s-turntable-phenom-to-the-mets-first-artist-in-residence/

Communication of Needs: An Example

Our repository contains digital collections. Digital collections contain objects that are both born-digital and digital surrogates of analog objects. Objects can be a book, journal, image, artwork, or other. Objects can contain multiple parts. Each object or part can have a digital representation. It may have multiple files for different types of representation.

Digital collections should have curators, titles, identifiers. Objects could have titles, subjects, identifiers, creators, languages, and relationships to other bibliographic resources.

Some digital collections and some objects can be viewed by only a subset of users. All objects can be edited only by a editors.

When Starting Towards Modeling...

1. What concepts exist?
    a. Entities?
    b. Relationships?
    c. Types?
    d. Operations?
2. What information structure do you want to preserve?
3. Any existing models or patterns you could use?

Communication of Needs: An Example

Our repository contains digital collections. Digital collections contain objects that are both born-digital and digital surrogates of analog objects. Objects can be a book, journal, image, artwork, or other. Objects can contain multiple parts. Each object or part can have a digital representation. It may have multiple files for different types of representation.

Digital collections should have curators, titles, identifiers. Objects could have titles, subjects, identifiers, creators, languages, and relationships to other bibliographic resources.

Some digital collections and some objects can be viewed by only a subset of users. All objects can be edited only by a editors.

Communication of Needs: An Example

Our repository contains digital collections. Digital collections contain objects that are both born-digital and digital surrogates of analog objects. Objects can be a book, journal, image, artwork, or other. Objects can contain multiple parts. Each object or part can have a digital representation. It may have multiple files for different types of representation.

Digital collections should have curators, titles, identifiers. Objects could have titles, subjects, identifiers, creators, languages, and relationships to other bibliographic resources.

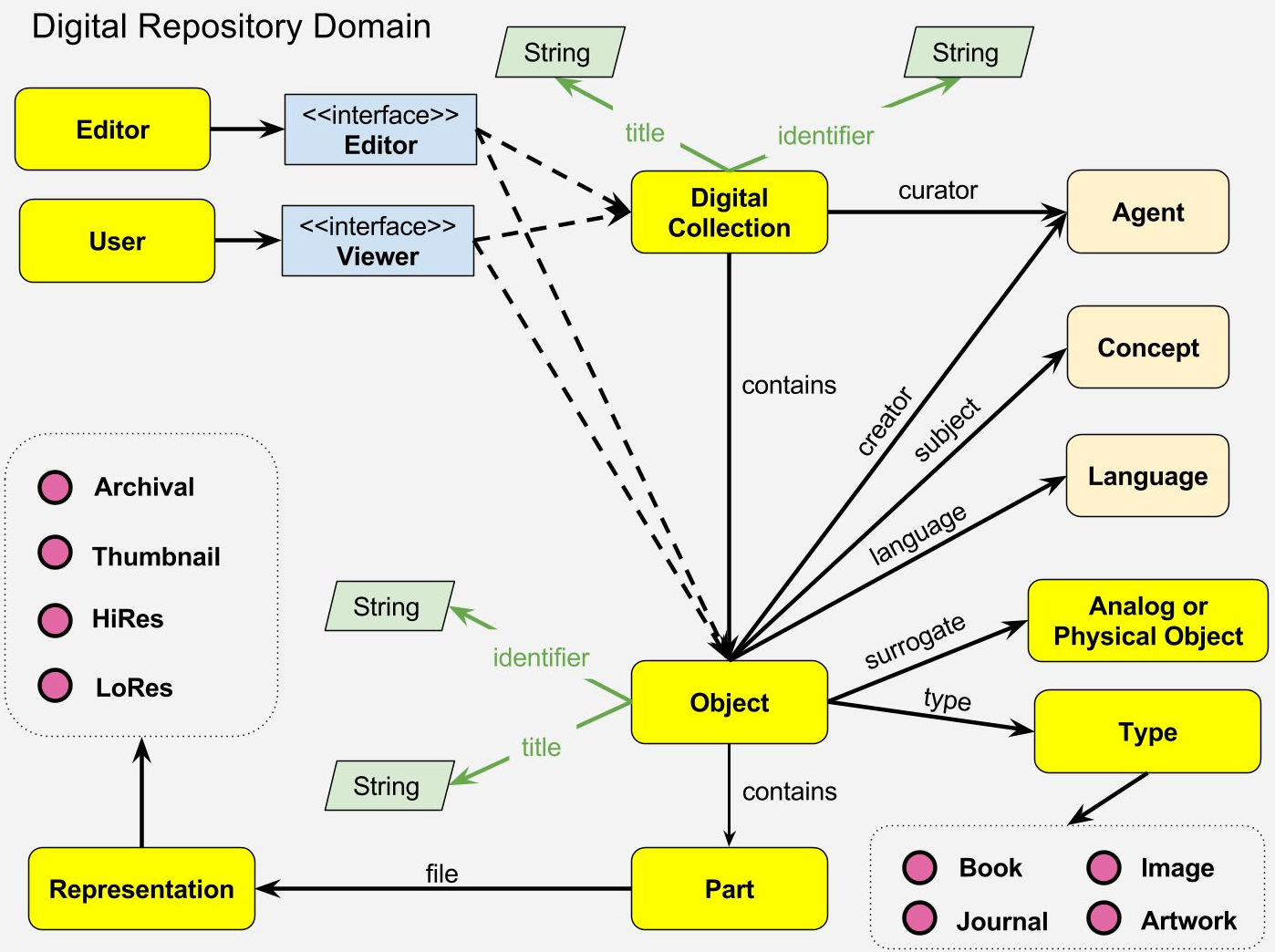
Some digital collections and some objects can be viewed by only a subset of users. All objects can be edited only by a editors.

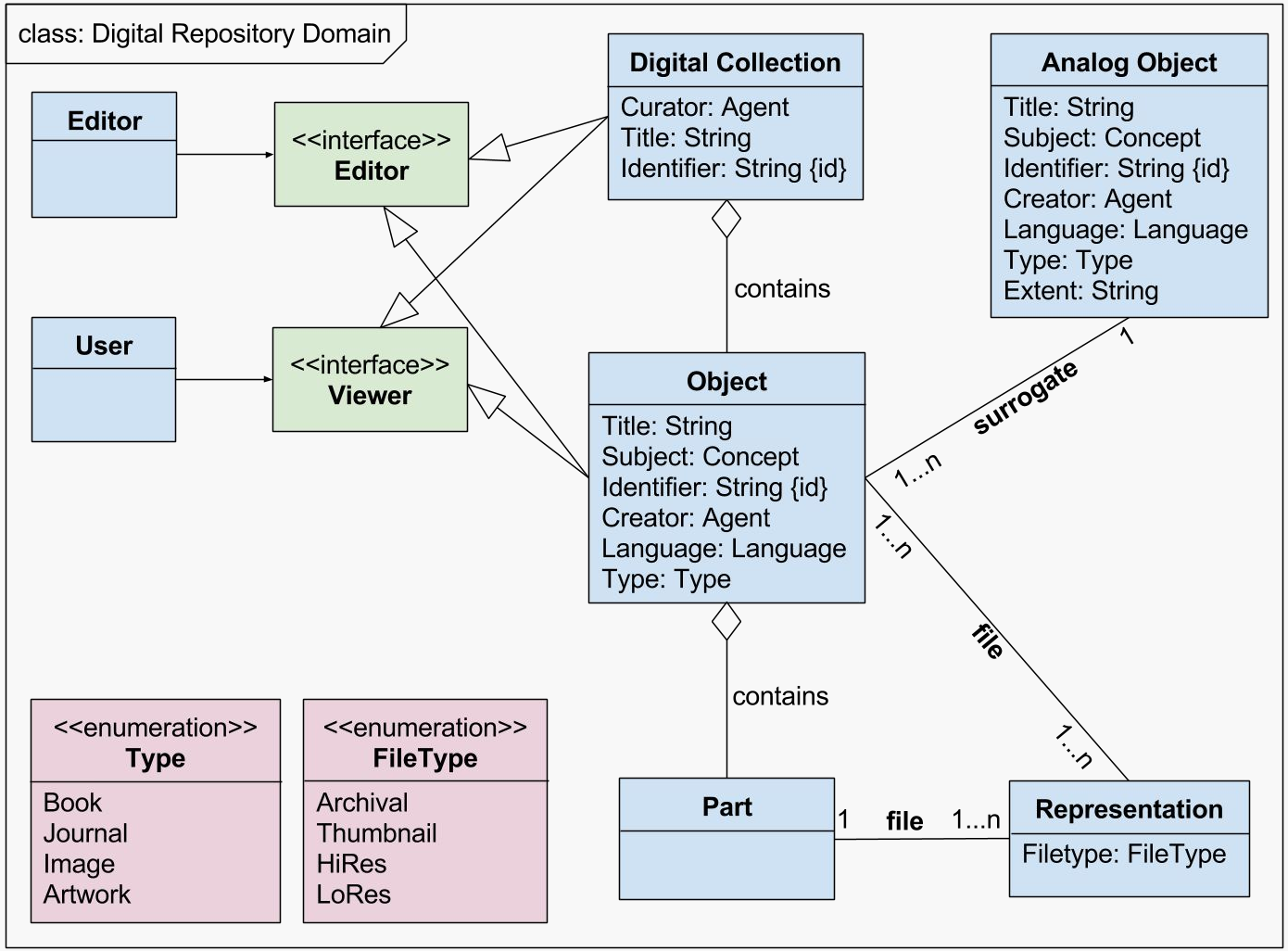
Communication of Needs: An Example

Domain: Digital Repository

- ***Classes***: digital collections, (digital) objects, analog/physical objects, parts, representation, formats, user, editor, …
- ***Attributes***: titles, subjects, identifiers, creators, languages, …
- ***Relationships***:
  - Collection contains Object ; Object contains Part
  - Object or Part has Representation
  - Object is a digital Surrogate of Analog/Physical
- ***Operations***:
  - Subset of Users can view Objects
  - Editors (Type of Users) can edit Objects
- ***Types/Enumerations***: (*format)* book, journal, image, artwork, …

One Possible View...

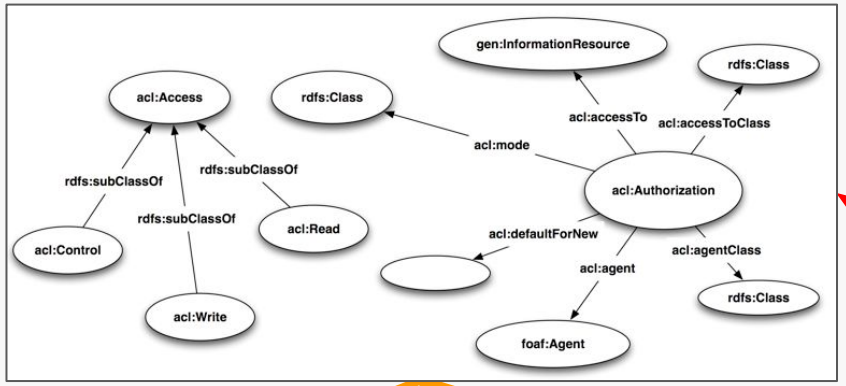


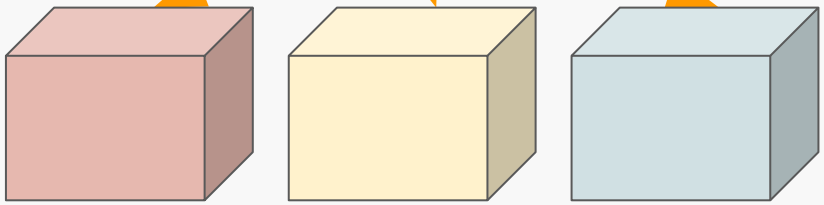


Another Possible View...

Data Models & Application Independence

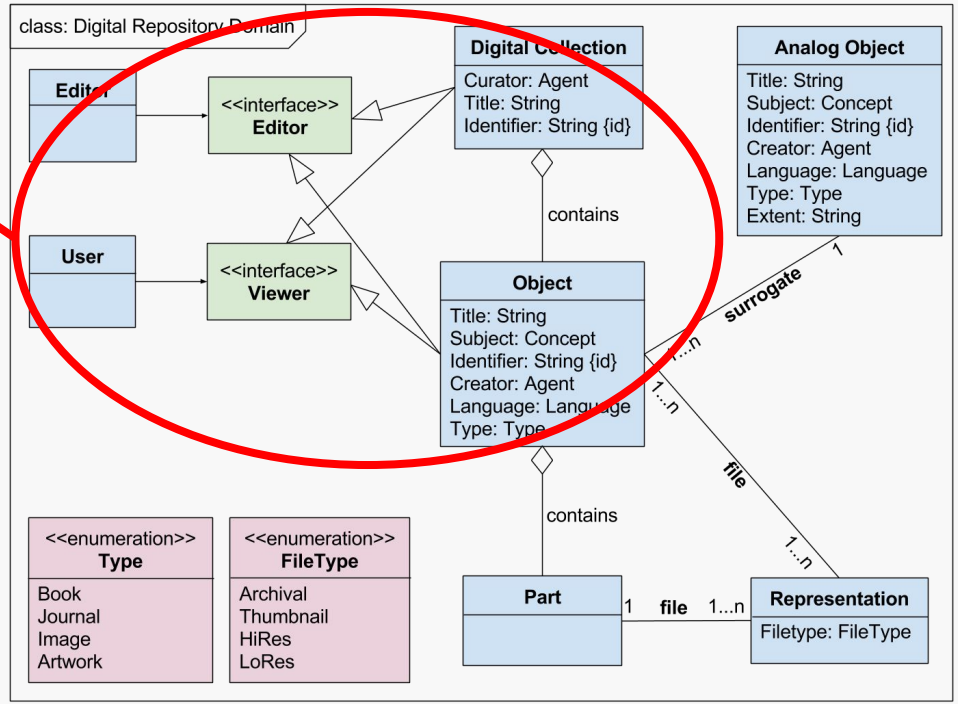
W3C WebAccessControl Model





WebAC Implementations

Our Model



Preservation of Information via Data Models

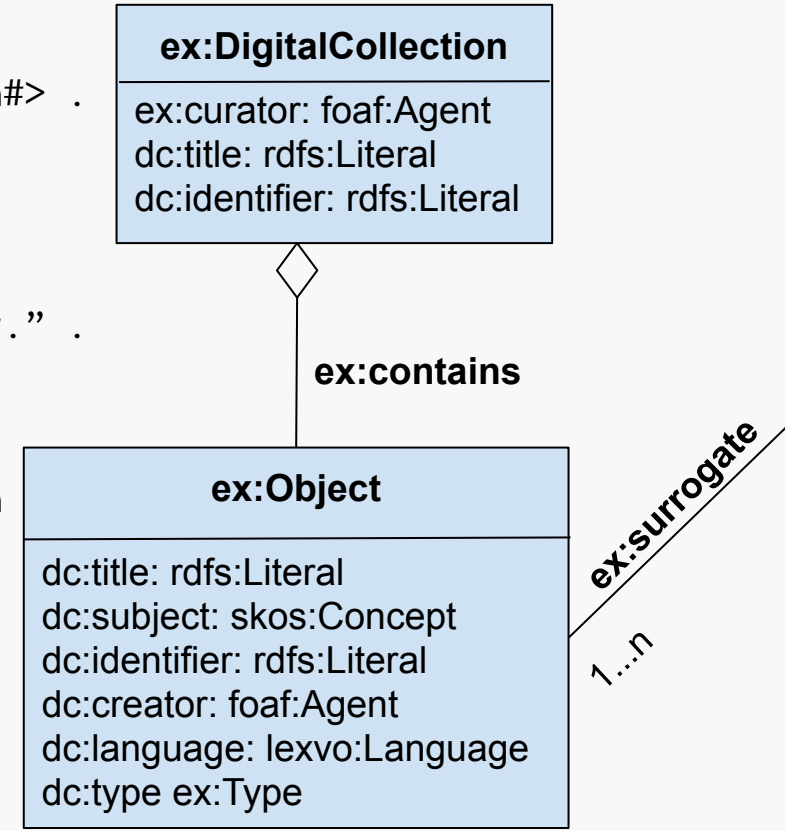
- If, or rather, as applications evolve/migrate, one can still decode the information context from the Model
- Break reliance of understanding information on 1 person, process, database structure, or application
- Capture Information Context for future re-use vis-a-vis the Conceptual Model

Data Models & Database Structures

- Internal in Internal/Conceptual/External Approach
- Alternatively, Database Structures are Subset of Data Models (thinking Entity-Relationship Models primarily)
  - Logical Structure of Database
  - Physical Structure of Database Instance
  - In RDF & Graph Stores, Logical == Physical
- Database Structures detail how Data in Domain Data Model is Stored & Managed

Data Models & Database Structures in RDF

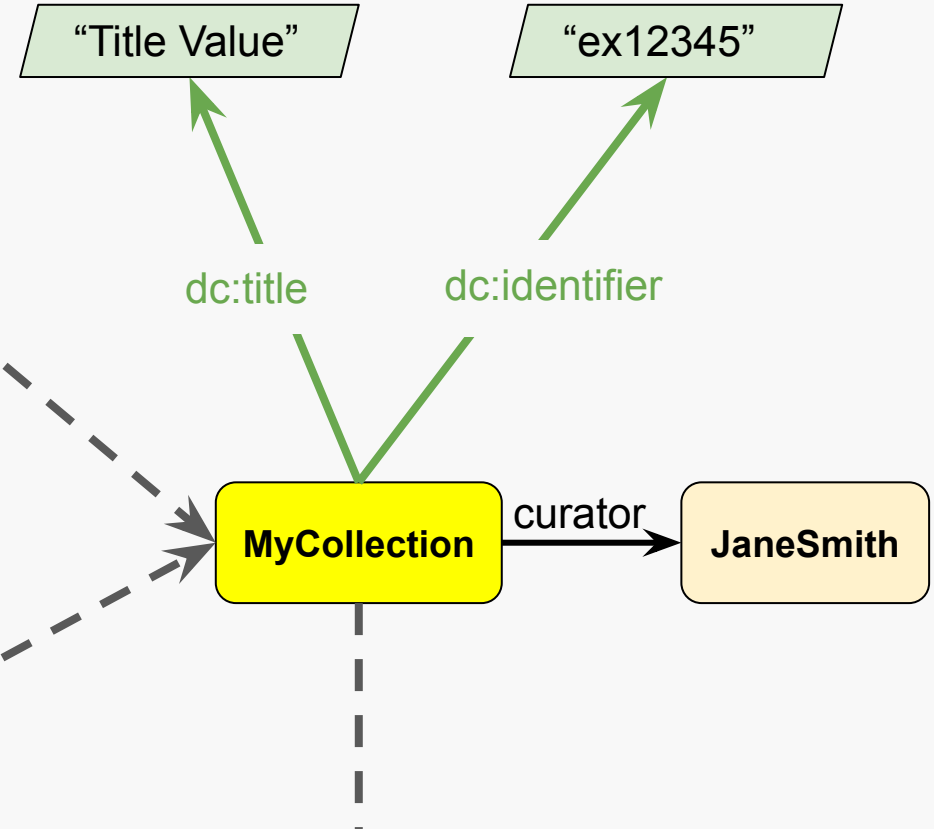
```
@prefix ex: <http://example.edu/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc: <http://purl.org/dc/terms/> .
ex:Object a rdfs:Class ;
  rdfs:label “Object” ;
  rdfs:comment “Digital Object held by a repository.” .
ex:DigitalCollection a rdfs:Class ;
  rdfs:label “Digital Collection” ;
  rdfs:comment “Aggregation of Digital Objects in a
    repository.” .
ex:contains a rdfs:Property ;
  rdfs:label “surrogate of” ;
  rdfs:domain ex:DigitalCollection ;
  …
```



Data Models & Serializations

- Converts Data Structures (from Data Models) into Format
- Format is Translation of Data Structure used for Transfer, Storage, Distribution among Systems
- Serialization Contains All Needed Information to (re-)Construct In-Memory Model
  - So Serializations are Logically Equivalent
- Examples:
  - Tabular Data -> CSV, TSV
  - RDF Data -> Turtle, N-Triples, RDF/XML, JSON-LD

Data Models & Serializations in RDF



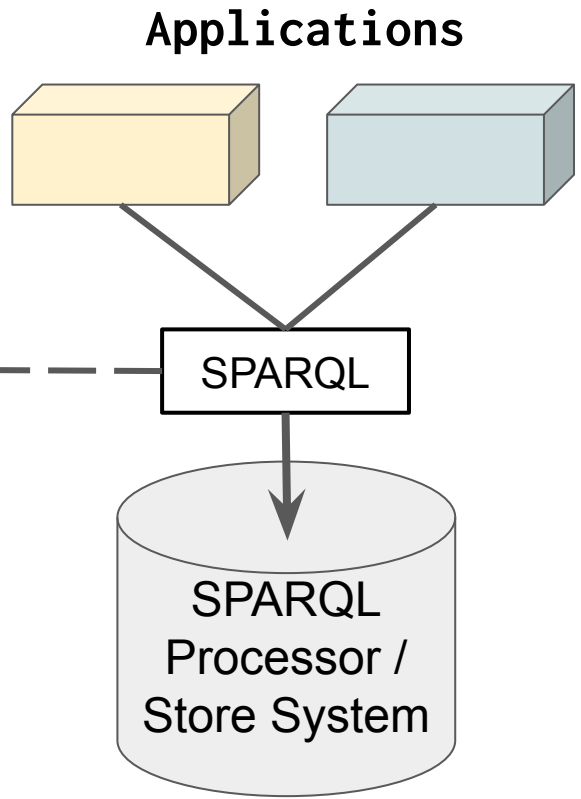

```
@prefix: <http://example.edu/> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/> .
:MyCollection a :DigitalCollection ;
  dc:identifier “ex12345” ;
  dc:title “Title Value” ;
  :curator :JaneSmith .
:JaneSmith a foaf:Agent ;
  foaf:name “Jane Smith” ;
  …
```

Data Models & Data Exchange Protocols

- Protocol State Machines in Olivé is Broader Idea
- Rules (Standards) Outlining Syntax, Semantics & Synchronization of Data or Information across Systems
- Protocols: *SOAP* ; *Z39.50* ; *SRU* ; *OAI-PMH* ; *SPARQL*
  - Also, *HTTP* == Hypertext Transfer Protocol, on which we build the Web & Leverage often for RDF transfer
- Data Models can Surface Needs, Expectations of as well as Data Instances & Formats used in Exchange Protocols

… Now Data Models & Data Protocols in RDF

```
Post
/test?graph-uri=http%3A%2F%2Fexample.edu%2F HTTP/1.1
Host: example.edu/sparql
Accept: */*
Content-Type: application/sparql-update
Content-Length: 136

PREFIX ex: <http://example.edu/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/terms/>
INSERT { ?digitalCollection ex:curator ex:JaneSmith }
WHERE { ?digitalCollection dc:identifier “ex12345” }
```



Application Logic & Data Models

- Application Logic == Logic Specific to an Application
- Fundamentally different from Domain & Infrastructure Logic due to Target & Re-usability:
    - Domain Logic should be reusable in systems or applications addressing the same domain
    - Infrastructure is generic technical capabilities
- Application Logic should leverage the Data Model, may serialize Data into Application-Language Objects

Application Logic & Data Models with RDF

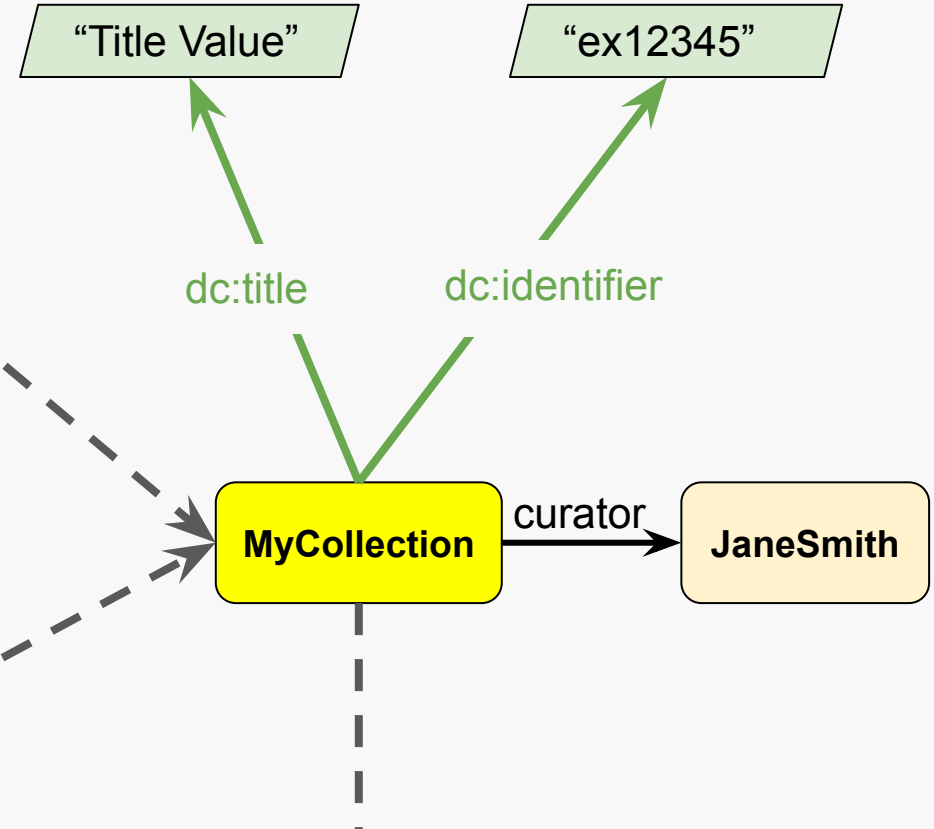
```ruby
module Hydra::Works
  # This module provides all of the Behaviors of a Hydra::Works::GenericFile
  #
  # behavior:
  #   1) Hydra::Works::FileSet can contain (pcdm:hasFile) Hydra::PCDM::File   (inherits from Hydra::PCDM::Object)
  #   2) Hydra::Works::FileSet can contain (pcdm:hasRelatedFile) Hydra::PCDM::File   (inherits from Hydra::PCDM::Object)
  #   3) Hydra::Works::FileSet can aggregate (pcdm:hasMember) Hydra::Works::FileSet
  #   4) Hydra::Works::FileSet can NOT aggregate anything other than Hydra::Works::FileSets
  #   5) Hydra::Works::FileSet can have descriptive metadata
  #   6) Hydra::Works::FileSet can have access metadata
  module FileSetBehavior
    extend ActiveSupport::Concern

    included do
      def self.type_validator
        Hydra::PCDM::Validators::CompositeValidator.new(
          Hydra::Works::NotCollectionValidator,
          super
        )
      end
```

github.com/projecthydra/hydra-works/blob/master/lib/hydra/works/models/concerns/file_set_behavior.rb

Data Models & Metadata Standards

- Rules that Establish Common Semantics for Particular Domain of Data
- In turn supports “Correct” Use & Understanding of Data by Users vis-a-vis Commonly-Defined Attributes
- Data Models leverage Metadata Standards to define their own Semantics & Requirements
- Data Models can reuse, extend multiple Metadata Standards
- Standards can be built off of a common Model

Data Models & Serializations in RDF




```
@prefix: <http://example.edu/> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/> .
:MyCollection a :DigitalCollection ;
  dc:identifier “ex12345” ;
  dc:title “Title Value” ;
  :curator :JaneSmith .
:JaneSmith a foaf:Agent ;
  foaf:name “Jane Smith” ;
  …
```

Vocabularies Are Perspectives

1. DC Terms (generalisms)
2. FOAF & BIBO (concretizations)
3. SKOS (nominalisms (bridged with foaf:focus))
4. FRBR (ideal forms)
5. PROV (entities, activities, agents; qualified roles)
6. OpenAnnotation (descriptions of descriptions)
7. BibFrame (nominal things)
8. Schema.org (a potpurri of perspectives)

What Are We Describing? Niklas Lindström, ELAG 2015 http://goo.gl/49ZCXW

Expressing Data Models

Quick Review of Modeling Languages

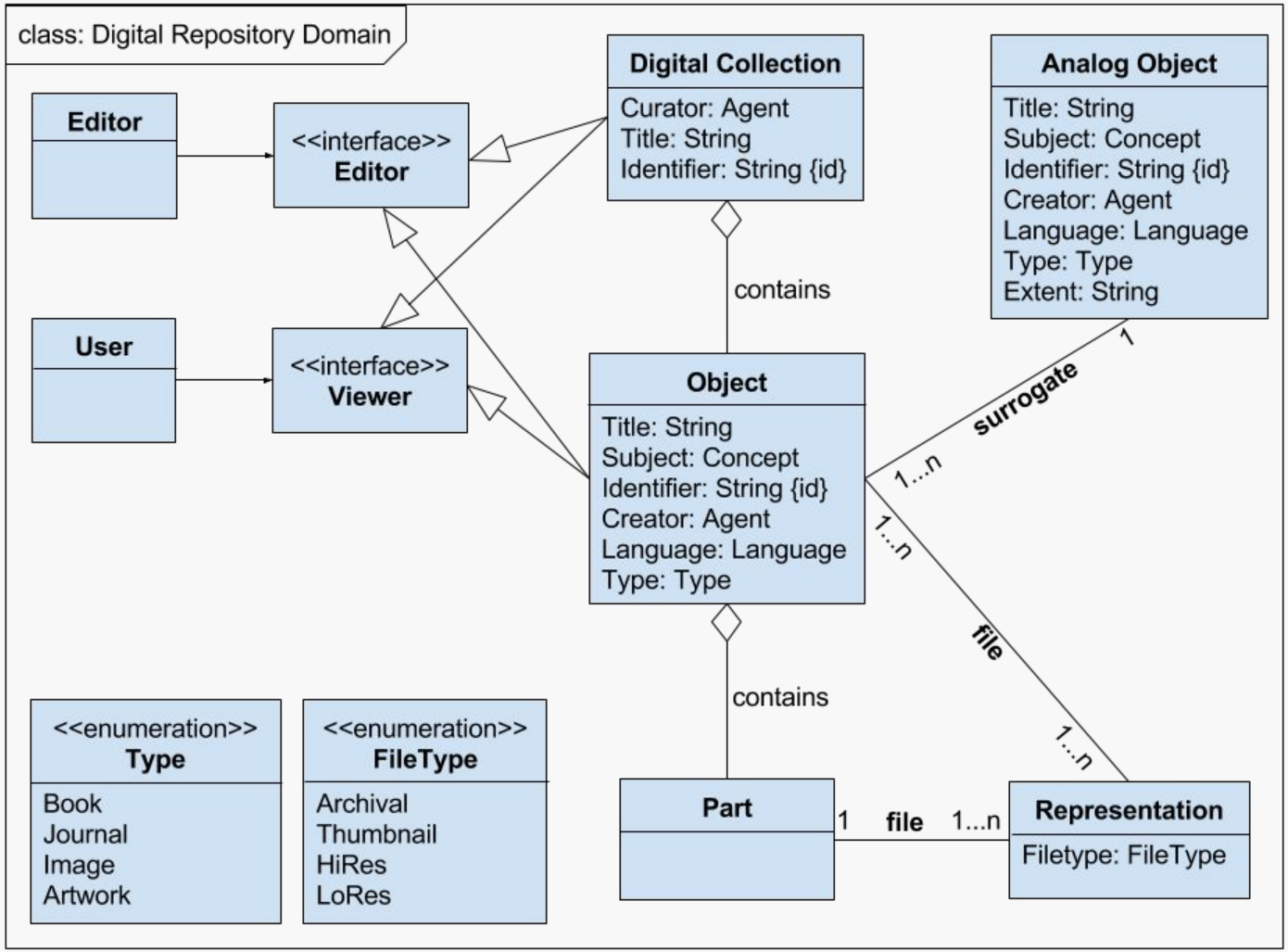
How models get expressed - a limited review of some conceptual modeling languages & methods:

- UML*
- XML*
- Java (Ruby, Python, ... Programming Language) Classes
- RDF
- Diagramming

Universe Modeling Language (UML)

- Standardized Modeling Language
- Complex but General Purpose
- Different Diagram Groups
- User for Software Engineering
- Visualizes designs of Systems from a few approaches:
  - Use Cases
  - Static or Class Diagrams
  - State changes
  - ...

Unified Modeling Language (UML) Simple Class Diagram



uml-diagrams.org/class-diagrams-overview.html

XML

- Document-focused model, with tree structure containing nodes in hierarchies - elements, attributes, values
- Languages that support models in XML:
  - XML Schema
  - RELAX NG
  - Schematron

```
<xs:element name="location" type="locationDefinition"/>
<!--   -->
<xs:complexType name="locationDefinition">
 <xs:sequence>
   <xs:element ref="physicalLocation" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="shelfLocator" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="url" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="holdingSimple" minOccurs="0"/>
   <xs:element ref="holdingExternal" minOccurs="0"/>
  </xs:sequence>
  <xs:attributeGroup ref="languageAttributeGroup"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
  <xs:attribute name="altRepGroup" type="xs:string"/>
</xs:complexType>
<!--

********   Subordinate Elements for <location>
  -->
<!--
********** physicalLocation **********
 -->
<xs:element name="physicalLocation" type="physicalLocationDefinition"/>
<!--   -->
<xs:complexType name="physicalLocationDefinition">
 <xs:simpleContent>
  <xs:extension base="stringPlusLanguagePlusAuthority">
    <xs:attributeGroup ref="xlink:simpleLink"/>
    <xs:attribute name="displayLabel" type="xs:string"/>
    <xs:attribute name="type" type="xs:string"/>
   </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<!--   -->
<xs:element name="shelfLocator" type="stringPlusLanguage"/>
<!--
```

XML data model:

Portion of XML Schema for MODS 3.6

www.loc.gov/standards/mods/v3/mods-3-6.xsd

Java Classes

- Object-Oriented Programming Language:
  - Objects aggregate Data, Attributes & Methods
  - Objects communicate with each other by invoking methods
  - Basic Programming Unit is a Class
  - Object == Instance of a Class
- In Java, all Classes except *Object* inherit information defined in another Class
  - Java's *Object* class defines Common Behaviors
  - Java Classes define Data Representation for Objects
- Class declares variables for storing values of Primitive Types & references to Objects

Data Models Captured in Java Classes

```java
@Scope("request")
@Path("/{path: .*}")
public class FedoraLdp extends ContentExposingResource {

    private static final Logger LOGGER = getLogger(FedoraLdp.class);
```

[SKIPPED / PARAPHRASED CONTENT HERE]

```java
    /**
     * Retrieve the node profile
     *
     * @param rangeValue the range value
     * @return a binary or the triples for the specified node
     * @throws IOException if IO exception occurred
     */
    @GET
    @Produces({TURTLE + ";qs=1.0", JSON_LD + ";qs=0.8",
            N3, N3_ALT2, RDF_XML, NTRIPLES, APPLICATION_XML, TEXT_PLAIN, TURTLE_X,
            TEXT_HTML, APPLICATION_XHTML_XML})
    public Response getResource(@HeaderParam("Range") final String rangeValue) throws IOException {
        checkCacheControlHeaders(request, servletResponse, resource(), session);

        LOGGER.info("GET resource '{}'", externalPath);
```

[…]

https://github.com/fcrepo4/fcrepo4/

Data Models with Ruby Class Using ActiveTriples

```ruby
module DPLA::MAP
  class Aggregation < ActiveTriples::Resource
    configure :type => RDF::ORE.Aggregation

    validates_presence_of :sourceResource, :originalRecord, :isShownAt, :object, :provider

    property :sourceResource, :predicate => RDF::EDM.aggregatedCHO, :class_name => 'DPLA::MAP::SourceResource'
    property :dataProvider, :predicate => RDF::EDM.dataProvider, :class_name => 'DPLA::MAP::Agent'
    property :originalRecord, :predicate => RDF::DPLA.originalRecord
    property :hasView, :predicate => RDF::EDM.hasView, :class_name => 'DPLA::MAP::WebResource'
    property :intermediateProvider, :predicate => RDF::DPLA.intermediateProvider, :class_name => 'DPLA::MAP::Agent'
    property :isShownAt, :predicate => RDF::EDM.isShownAt, :class_name => 'DPLA::MAP::WebResource'
    property :object, :predicate => RDF::EDM.object, :class_name => 'DPLA::MAP::WebResource'
    property :preview, :predicate => RDF::EDM.preview, :class_name => 'DPLA::MAP::WebResource'
    property :provider, :predicate => RDF::EDM.provider, :class_name => 'DPLA::MAP::Agent'
    property :rightsStatement, :predicate => RDF::EDM.rights, :class_name => 'DPLA::MAP::RightsStatement'

    def jsonld_context
      DPLA::MAP::CONTEXT['@context']
    end

    def to_jsonld
      JSON::LD::API.frame(JSON.parse(dump(:jsonld)), DPLA::MAP::FRAME)
    end
  end
end
```

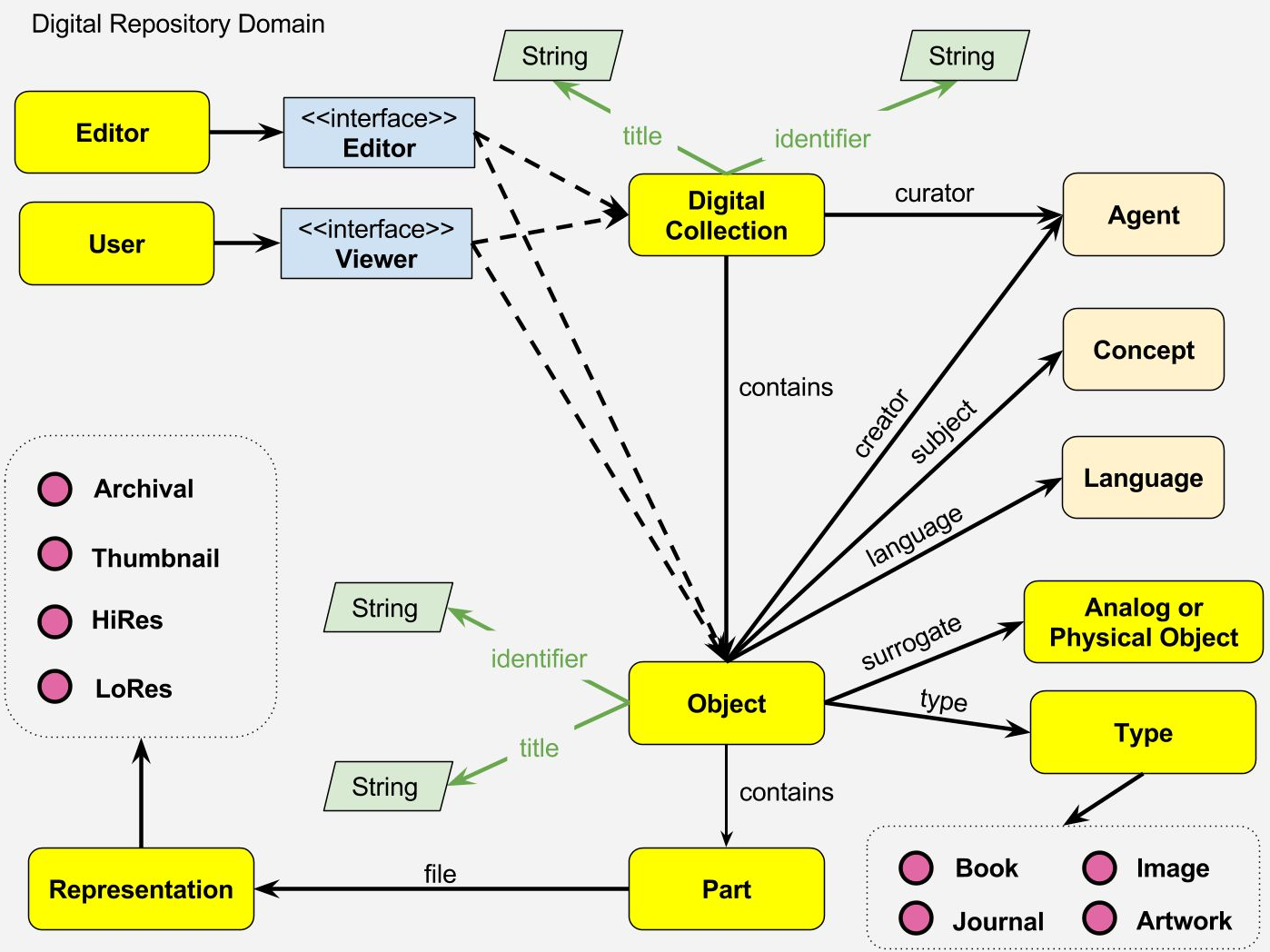
https://github.com/dpla/dpla_map/blob/map-4.0/lib/dpla/map/aggregation.rb

RDF, RDFS, OWL, RDF/XML - here, FOAF

```
  <rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Organization" rdfs:label="Organization"
    rdfs:comment="An organization." vs:term_status="stable">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
<!--    <rdfs:subClassOf><owl:Class rdf:about="http://xmlns.com/wordnet/1.6/Organization"/></rdfs:subClassOf> -->
    <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
    <rdfs:isDefinedBy rdf:resource="http://xmlns.com/foaf/0.1/"/>
    <owl:disjointWith rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
    <owl:disjointWith rdf:resource="http://xmlns.com/foaf/0.1/Document"/>
  </rdfs:Class>
  <rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Group" vs:term_status="stable" rdfs:label="Group"
    rdfs:comment="A class of Agents.">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
  </rdfs:Class>
  <rdfs:Class rdf:about="http://xmlns.com/foaf/0.1/Agent" vs:term_status="stable" rdfs:label="Agent"
    rdfs:comment="An agent (eg. person, group, software or physical artifact).">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <owl:equivalentClass rdf:resource="http://purl.org/dc/terms/Agent"/>
<!--    <rdfs:subClassOf><owl:Class rdf:about="http://xmlns.com/wordnet/1.6/Agent-3"/></rdfs:subClassOf> -->
  </rdfs:Class>
```

http://xmlns.com/foaf/spec/

RDF Diagram (Mostly) Using Graffoo



Data Models Breakouts

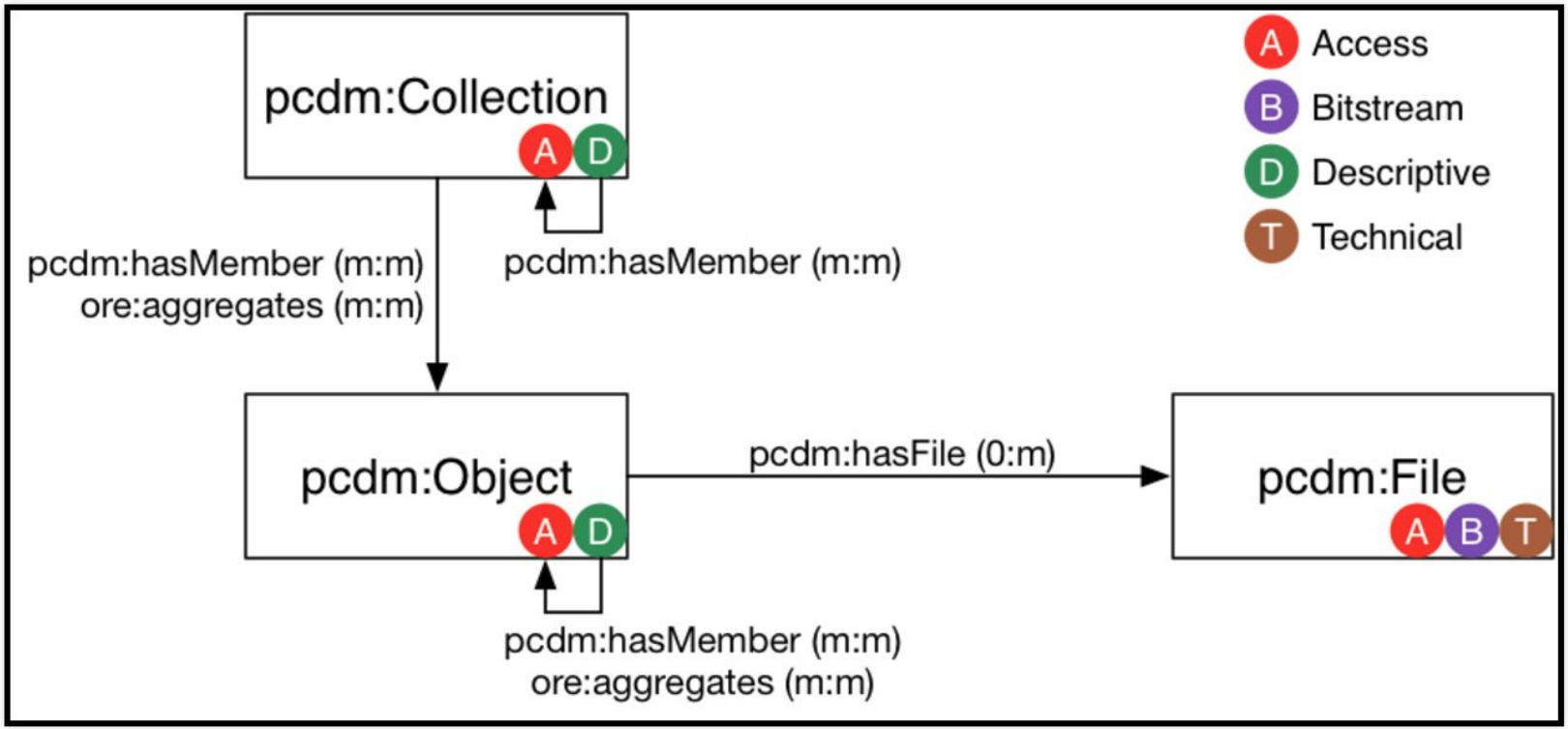
3 Data Models Breakout Groups Options

1. Portland Common Data Model (PCDM)
2. Europeana Data Model (EDM) / Digital Public Library of America (DPLA)
3. International Image Interoperability Framework (IIIF)

Portland Common Data Model (PCDM)

- Started Officially in 2015
- Community Effort for Interoperable Modeling of Digital Repository Objects
- PCDM extends Object Reuse & Exchange (ORE) Data Model
- Interacts with Other Specifications (like LDP), Platforms (like Fedora 4), but Modeled to be Neutral
- Hydra works with PCDM Data from Datastore (Fedora 4) by Translating into Ruby Objects leveraging ActiveFedora

PCDM “Core”

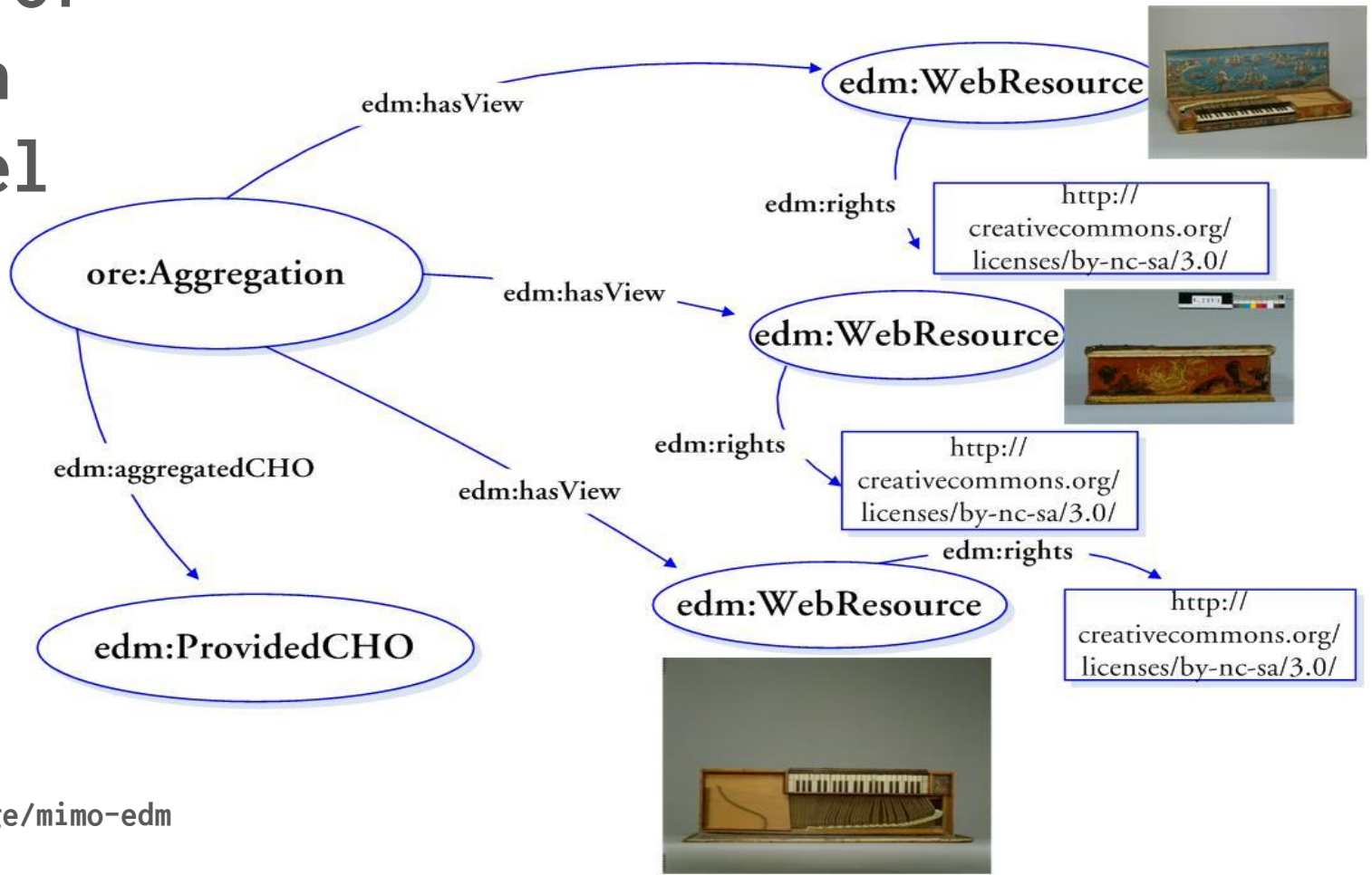


github.com/duraspace/pcdm/wiki

Europeana Data Model (EDM) / Digital Public Library of America (DPLA) MAP

- RDF-based, Descriptive Models
- Focus is Cultural Heritage Resource Aggregation
- Extends CIDOC-CRM as well as Other Models, Vocabularies
- DPLA is Subset of EDM
- Basis for Profiles, i.e. not Designed to be Exhaustive
- More Concerned with Resource Description than System Functions

Fragment of Europeana Data Model Diagram

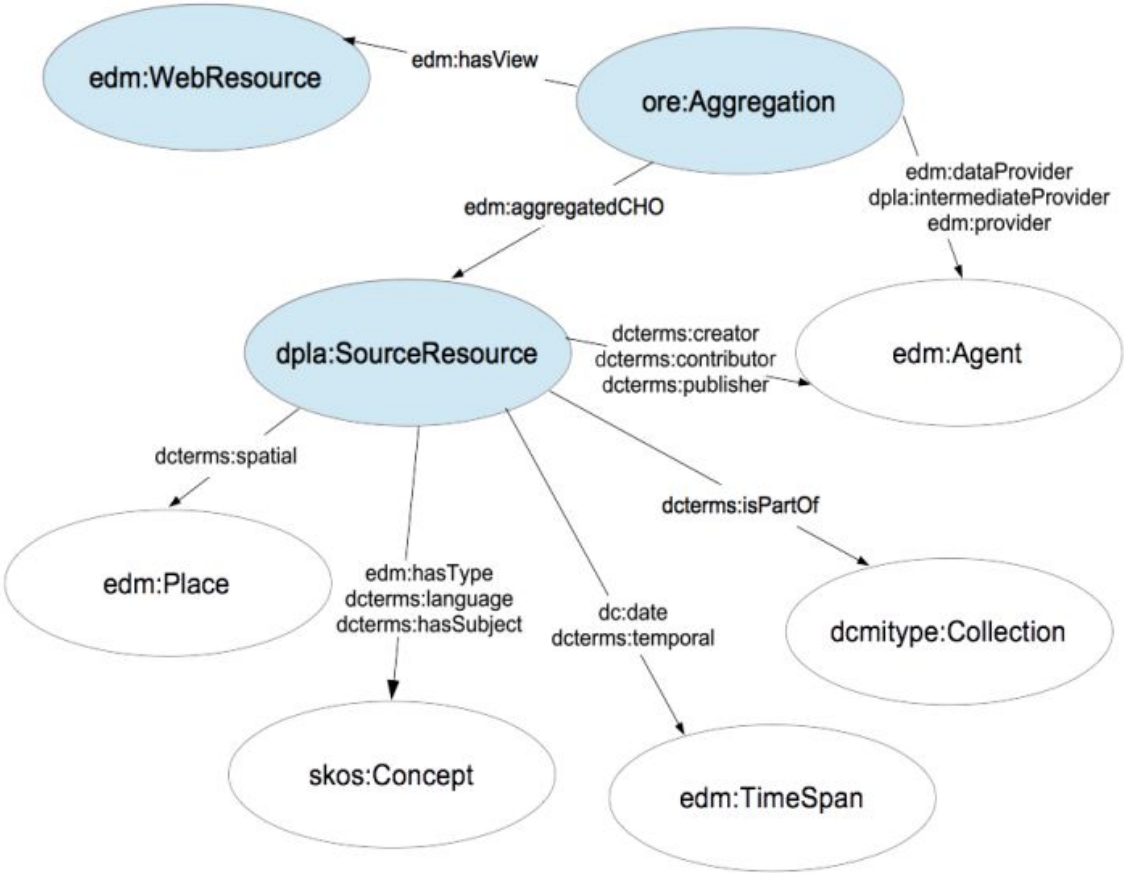


pro.europeana.eu/page/mimo-edm

DPLA v.4 Model Diagram

2.0 Digital Public Library of America Domain Model, v4

Core classes highlighted in blue.

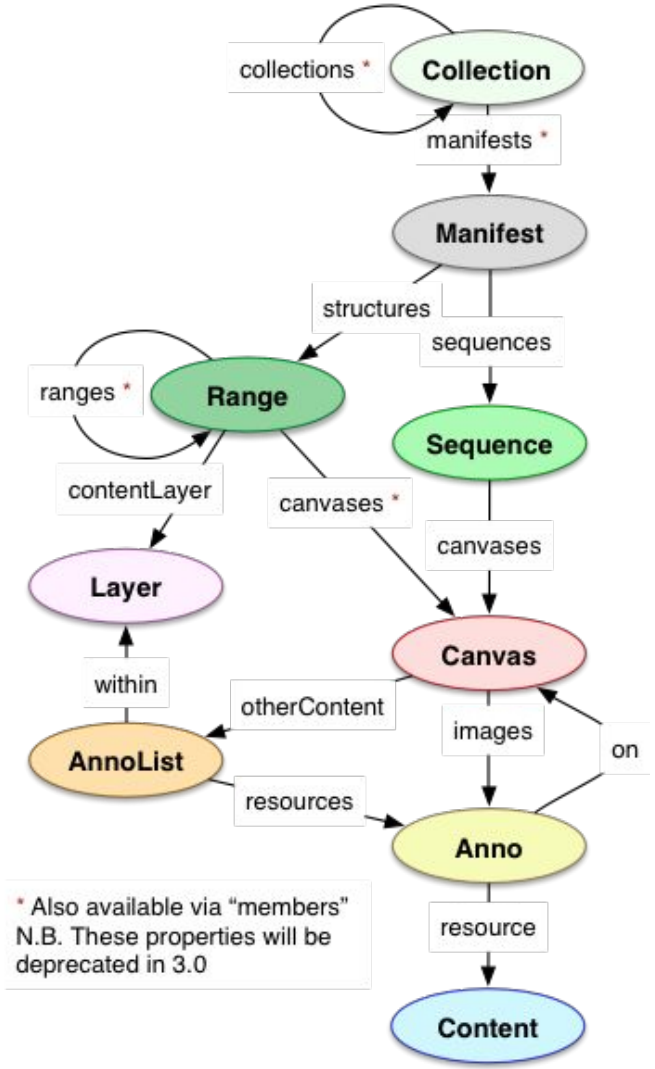


dp.la/info/developers/map/

International Image Interoperability Framework (IIIF)

- Set of 4 APIs & Models Working Together to Share Images:
  - Presentation (the focus of the break-out)
  - Image
  - Authentication
  - Search
- Growing Range of IIIF-Compatible Image Viewer Systems
  - Mirador
  - Universal Viewer
  - ...

IIIF Presentation API Model Diagram



iiif.io/api/presentation/2.1/

Data Models Breakouts Activity

Learn about Your Data Model & Discuss:

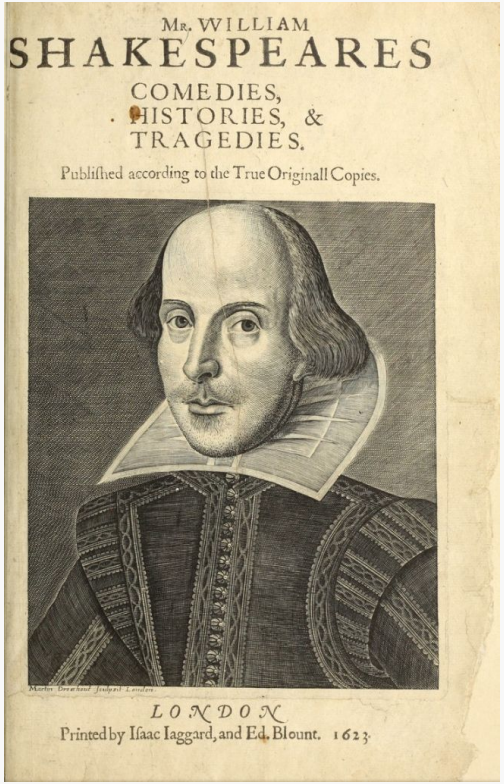
1. Entity Types ;
2. Relationships Types ;
3. Attributes .
4. Contextualize within Domain, Applications, Standards ;
5. Diagram an Instance of the Data Model with a Shared Example.

*** Please Capture Work & Diagrams in the Shared Notes ***

Breakouts' Shared Example

http://bit.ly/HC16DataModelExample

Digital Surrogate of *"This, the first edition of the collected plays of William Shakespeare, is commonly known as the First Folio. Printed in London and issued in 1623..."* in the Boston Public Library Digital Collections.



Breakouts until 11:50

PCDM - Stage
IIIF - Lobby
EDM/DPLA - Back of auditorium
Use the Shared Drive for Notes

10 Minute Break
Reconvene at 12:05 ish

Breakouts Recap

Volunteer from each Breakout to Report Back on Your Group's Modeling Work

Different Models, Different Data?

- How do our breakout group responses, ideas, approaches compare?
- How do we use modeling to build interoperability + shared understanding?
- Where do we see differences in what data modeling accomplishes across these breakouts?

Building Data Modeling in the Hydra Community

Maintaining Hydra Data Modeling Momentum

- Hydra Metadata Interest Group
  - #metadata on project-hydra Slack
- Project-hydra Slack channels that discuss modeling:
  - #architecture
  - #geomodeling
  - #hylandora
- Hydra-Tech Mailing List
- Notes & Shared Resources from Workshop- Keep Adding to this!

Broader or Related Communities Working on Modeling

- PCDM
- IIIF
- Fedora 4 (& Fedora broadly)
- Islandora
  - CLAW / Islandora & Fedora 4 Architecture
  - Islandora Interest Groups (Includes Metadata)
- Europeana Data Modeling Work
- W3C Efforts: LDP, Shapes, …
- DCMI Work

Data Modeling Tools & Resources

- Data Modeling Resources Going Forward
  - Starter List of Tools
  - Some Links to Modeling Work in Cultural Heritage Institutions
- Data Modeling Needs in Hydra & Related (PCDM, Fedora) Communities:
  - Location for Open Discussions & Issues

Your ideas to continue support for data modeling in communities?

Thank you!

bit.ly/HC16DataModels