RIRI 2011

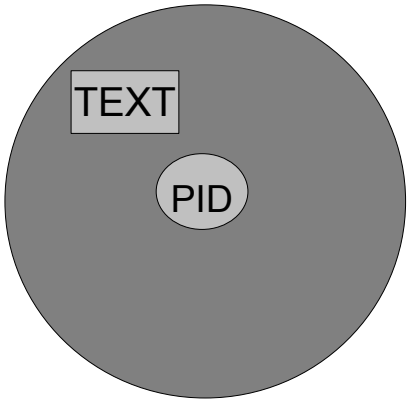
Content Modelling & Disseminators

Steve Bayliss, Acuity Unlimited
Chris Wilper, DuraSpace

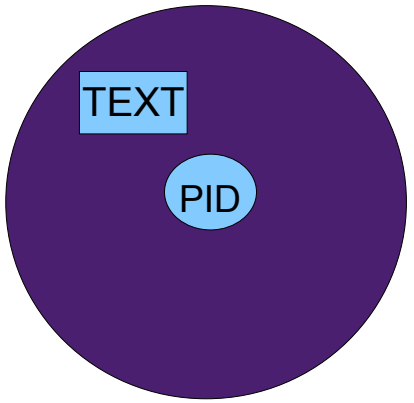
Part I:
Content Modeling

What is a Content Model?

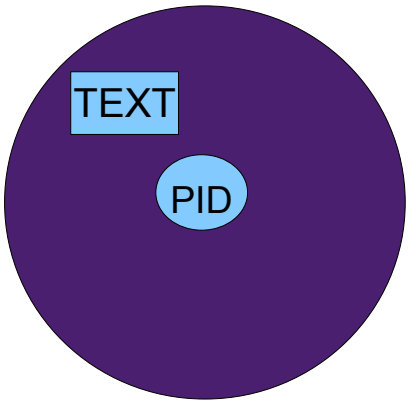
A description of the structure of a group of similar objects.



Content Model



Instance 1



Instance 2

Uses of Content Models

- Construction
  - As a "template"
- Validation
  - Is this thing really an article?
- Discovery
  - List all video objects added this week
- Managability
- Interoperability

Descriptive & Prescriptive Modeling

- Descriptive
  - Describe what exists
  - Look for inherent patterns in the data
- Prescriptive
  - Enforce rules for what should exist
  - Fix or drop non-conforming data

Objects in Fedora

- Four basic classes of objects
  - Data objects
    - The "content" of the repository
  - Content Model (CModel) objects
    - Define the structure of data objects
  - Service Definition (SDef) objects
  - Service Deployment (SDep) objects

Fedora Content Models

- Implemented using Content Model objects (CModels)
  - CModels are Fedora objects
- A CModel defines a "class" of data objects
- Data objects reference the CModels to which they belong
- Every object has a CModel
  - By default: the "system" content model object

CModel Object

- Defines the datastreams that a conforming data object should contain
  - Datastream ID
  - MIME type (optional)
  - FORMAT_URI (optional)
  - XML Schema (optional)
  - Whether it is required or optional

Example

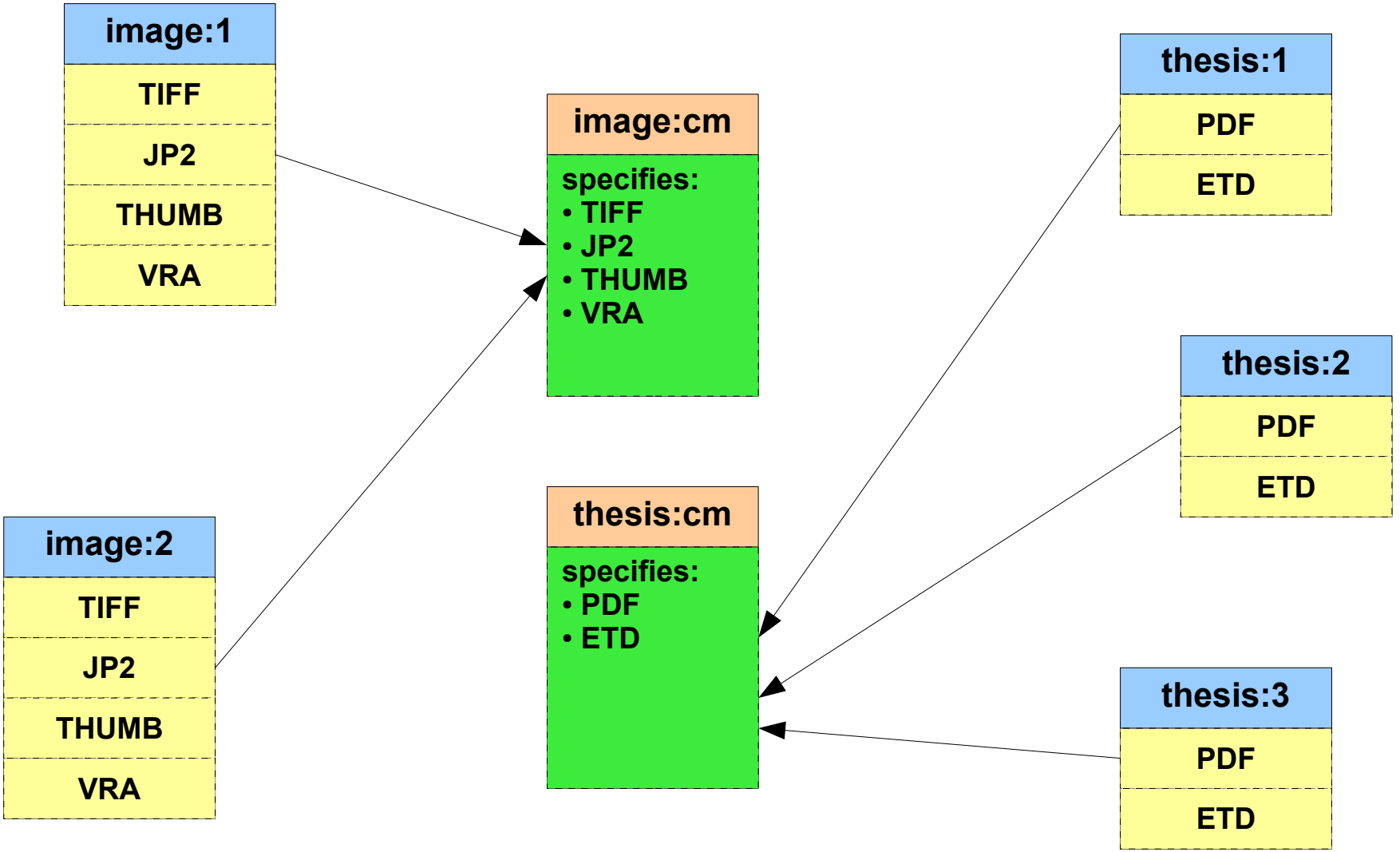


image:1
TIFF
JP2
THUMB
VRA
image:cm
specifies:
• TIFF
• JP2
• THUMB
• VRA
thesis:1
PDF
ETD
thesis:2
PDF
ETD
image:2
TIFF
JP2
THUMB
VRA
thesis:cm
specifies:
• PDF
• ETD
thesis:3
PDF
ETD

CModel Objects

- Descriptive, not Prescriptive
- You can ingest objects that do not conform to their CModel
- But you can check for conformance
  - API method: Validate

CModel object – How?

- Create a CModel Object
- Add a relationship to the system content model object
  - info:fedora/fedora-system:ContentModel-3.0
  - this identifies that this is a CModel object
- Add a DS-COMPOSITE-MODEL datastream
  - defines the datastreams that conforming data objects should contain, and information about them

CModel RELS-EXT

```
<rdf:RDF xmlns:fedora-model="info:fedora/fedora-
system:def/model#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#">

  <rdf:Description
rdf:about="info:fedora/demo:DualResImage">

    <fedora-model:hasModel
rdf:resource="info:fedora/fedora-
system:ContentModel-3.0"></fedora-model:hasModel>

  </rdf:Description>

</rdf:RDF>
```

CModel DS-COMPOSITE-MODEL

```
<dsCompositeModel xmlns="info:fedora/fedora-
system:def/dsCompositeModel#">
  <dsTypeModel ID="MEDIUM_SIZE">
    <form MIME="image/jpeg"></form>
  </dsTypeModel>
  <dsTypeModel ID="FULL_SIZE">
    <form MIME="image/jpeg"></form>
  </dsTypeModel>
</dsCompositeModel>
```

Data object – How?

- Add relationships from data objects to CModel object(s)
- And of course...
  - Add the datastreams that the CModel object has defined

Data Object - RELS-EXT

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-
rdf-syntax-ns#" xmlns:fedora-
model="info:fedora/fedora-system:def/model#">

  <rdf:Description
rdf:about="info:fedora/demo:6">

    <fedora-model:hasModel
rdf:resource="info:fedora/demo:DualResImage"></fe
dora-model:hasModel>

  </rdf:Description>

</rdf:RDF>
```

Take a Look...

- http://riri.islandora.ca:8080/fedora/objects
- Find demo:DualResImage
  - list the datastreams
  - have a look at RELS-EXT
  - have a look at DS-COMPOSITE-MODEL
- Find demo:6
  - list the datastreams
  - have a look at RELS-EXT

Validate

- http://riri.islandora.ca:8080/fedora/objects/demo:6/validate
  - API-M method

Validate

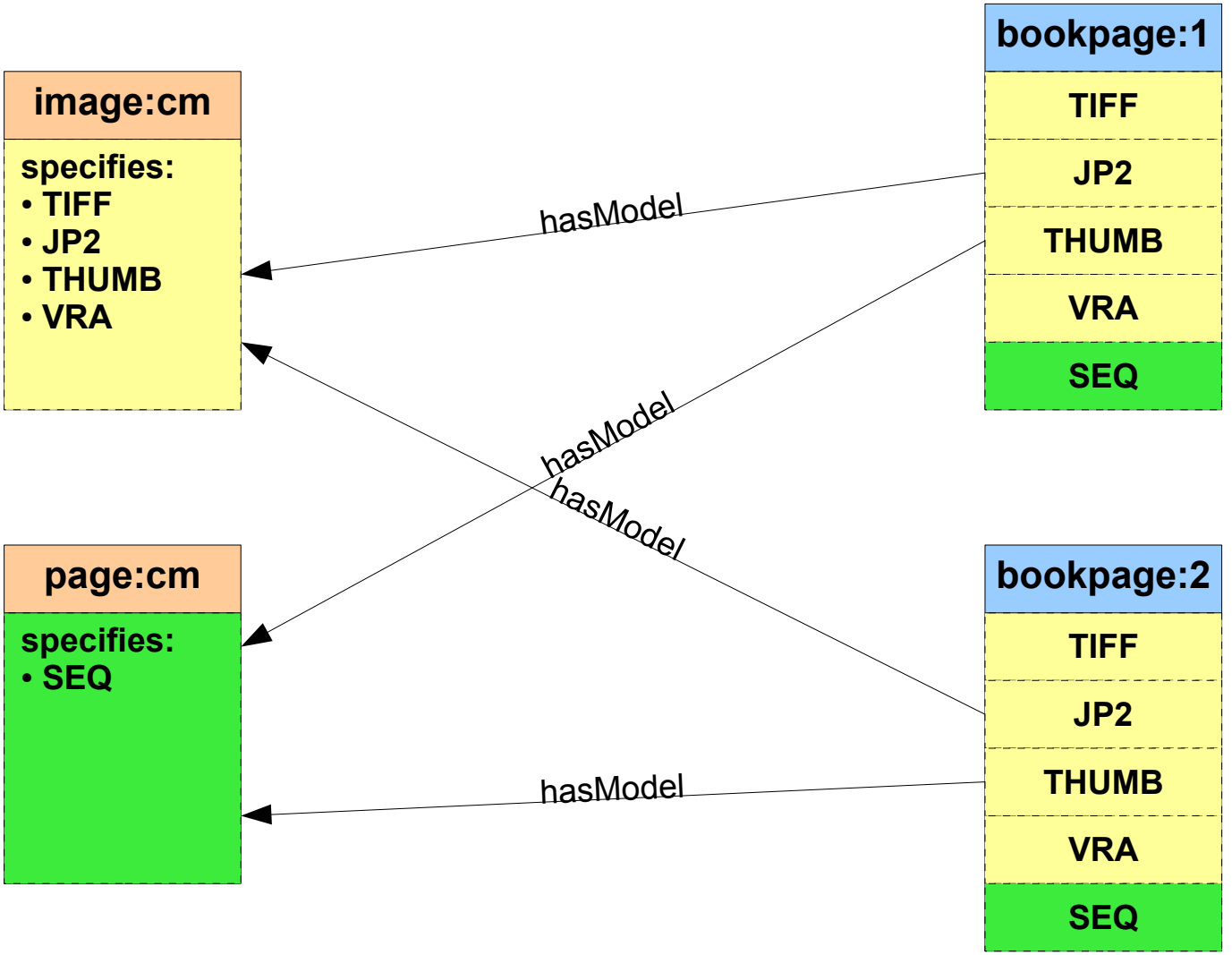
```
<validation pid="demo:6" valid="false">
  <asOfDateTime></asOfDateTime>
  <contentModels>
    <model>info:fedora/fedora-system:FedoraObject-3.0</model>
    <model>info:fedora/demo:DualResImage</model>
  </contentModels>
  <problems>
  </problems>
  <datastreamProblems>
    <datastream datastreamID="THUMBNAIL">
      <problem>Datastream 'THUMBNAIL' is required by the content
model 'demo:DualResImage'</problem>
    </datastream>
    <datastream datastreamID="DC">
      <problem>Datastream 'DC' is does not have the FORMAT_URI and
MIME_TYPE attributes required by 'fedora-system:FedoraObject-
3.0'</problem>
      <problem>Datastream 'DC' is does not have the FORMAT_URI and
MIME_TYPE attributes required by 'demo:DualResImage'</problem>
    </datastream>
  </datastreamProblems>
</validation>
```

RISearch – conforming objects

http://riri.islandora.ca:8080/fedora/risearch

```
select $object
from <#ri>
where
$object <fedora-model:hasModel>
<info:fedora/demo:DualResImage>
```

Mix and match



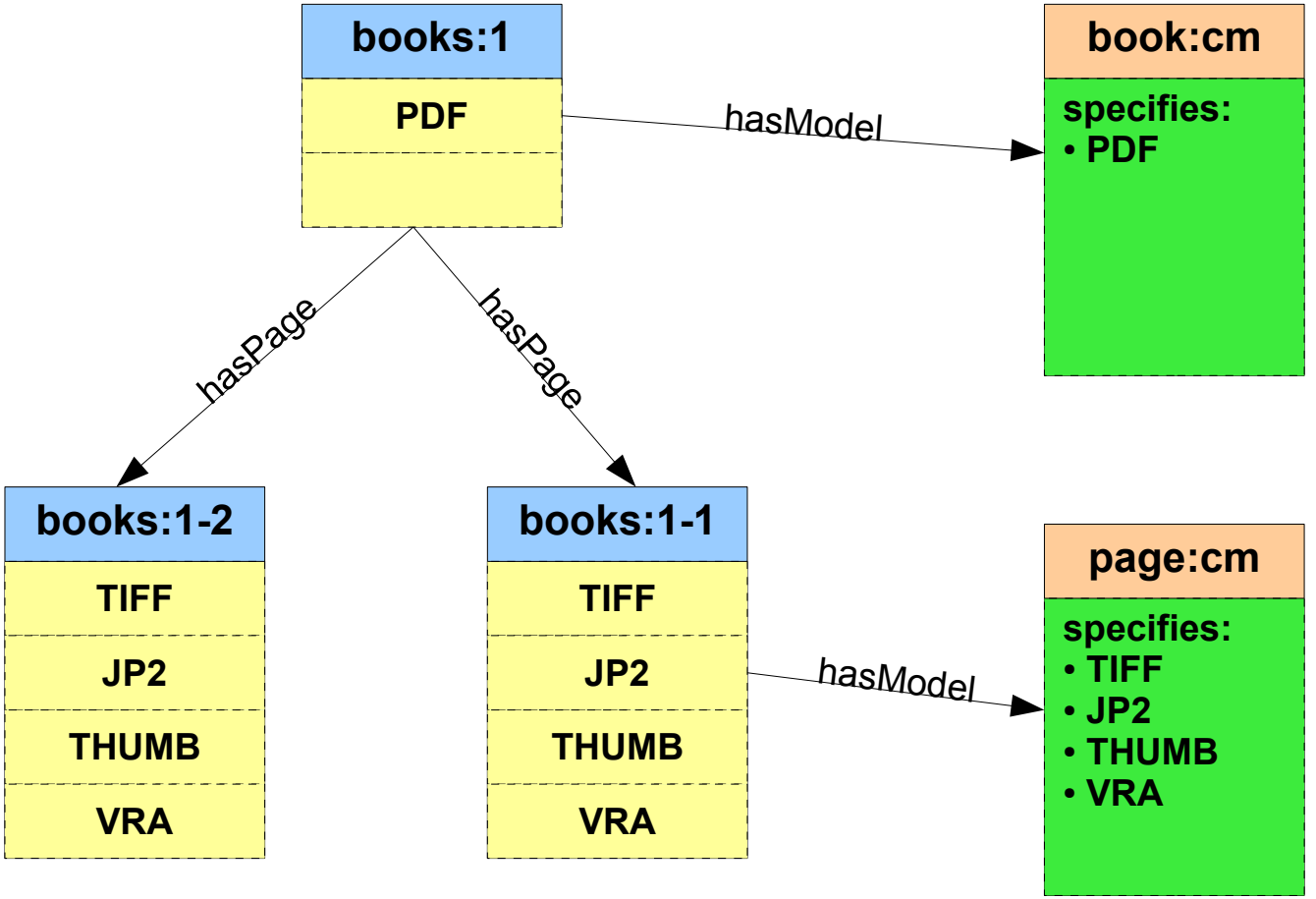
CModels can be "empty"

- CModels are used to identify the overall "class" or "type" of object
  - You don't have to specify any datastreams
  - Can be used just for specifying the type/class

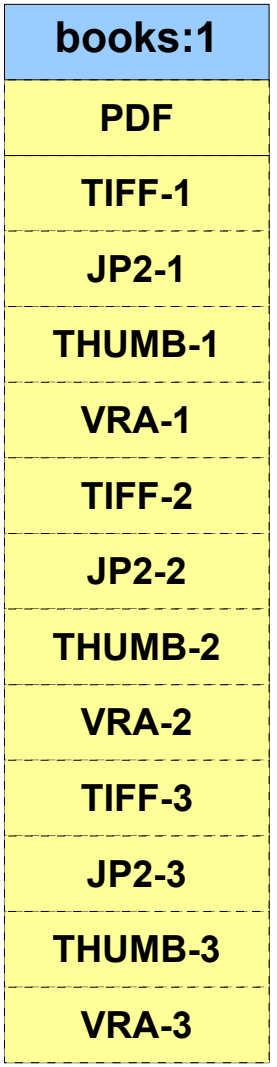
Atomistic and Compound models

- Atomistic and Compound models
  - Compound – single Fedora object for each entity
    - eg image datastreams, metadata datastreams
  - Atomistic – Fedora objects for each part of a composite object
  - Not an either/or choice
    - eg: individual objects for each datastream
    - eg: content and metadata in the same object, but each object belongs to a "parent" object with its own metadata
    - eg: single object for the "thing" with all datastreams in a single object

Atomistic model



Compound model



ECM – Enhanced Content Modeling

- Contributed by Asger Askov Blekinge
  - State & University Library, Denmark
- Added capabilities:
  - Specify optional datastreams
  - Specify & validate vs. XML Schema
  - Specify & validate vs. RDF Ontology
- Available in Fedora 3.4+

ECM – Optional Datastreams

```
<dsCompositeModel xmlns="info:fedora/fedora-
system:def/dsCompositeModel#">
  <dsTypeModel ID="MEDIUM_SIZE" optional="true">
    <form MIME="image/jpeg">
  </dsTypeModel>
  <dsTypeModel ID="FULL_SIZE">
    <form MIME="image/jpeg"/>
  </dsTypeModel>
</dsCompositeModel>
```

ECM – XML Schema

```
<dsCompositeModel xmlns="info:fedora/fedora-
system:def/dsCompositeModel#">
  <dsTypeModel ID="MY_METADATA">
    <form MIME="application/xml">
    <extension name="SCHEMA">
      <reference type="datastream"
                 value="MY_METADATA_SCHEMA"/>
    </extension>
  </dsTypeModel>
</dsCompositeModel>
```

ECM – RDF Ontology

(RDF snippet from ONTOLOGY datastream of FedoraObject-3.0)

```
<owl:Class rdf:about="info:fedora/fedora-
system:FedoraObject-3.0#class">

  <rdfs:subClassOf>

    <owl:Restriction>

      <owl:onProperty rdf:resource="info:fedora/fedora-
system:def/model#hasModel"/>

      <owl:allValuesFrom rdf:resource="info:fedora/fedora-
system:ContentModel-3.0#class"/>

    </owl:Restriction>

  </rdfs:subClassOf>

</owl:Class>
```

Other Approaches

- Hydra?
- Islandora?
- This stuff will evolve..

Content Modelling
Q&A, Discussion

Part II: Disseminators a.k.a. Methods

Preservation Perspective

- Defining how you store your content
- Content in datastreams in Fedora objects
- Describing that content with metadata datastreams
- Defining content models to describe those objects

Access Perspective

- Defining how your content is made available
  - Datastreams are available directly
- Fedora's CMA lets you define "access points" on objects
- Define standard "views" of content
- Add useful services to your objects
  - Dynamic "views"
- Common patterns of methods

Motivations

- Data Hiding
  - Separate the view from the composition of the objects
- Add useful views to existing data
  - Attach behavior at content model level, not at object level

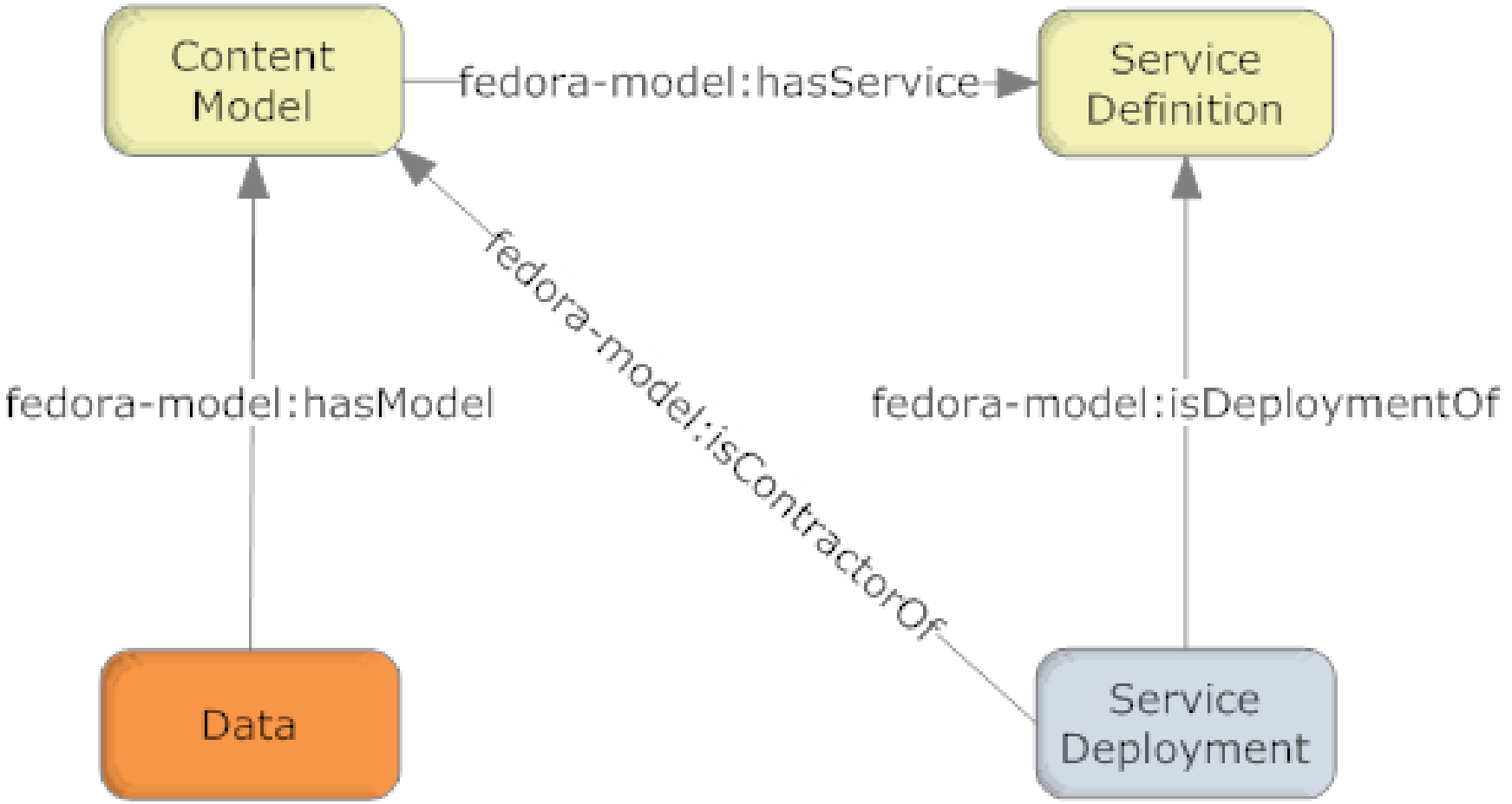
Examples

- Standard Metadata Format from different underlying data
- Provide JPG2000 extraction/selection over all images in repository
- Collection Navigation
  - Links to members can be provided independent of how they are stored.

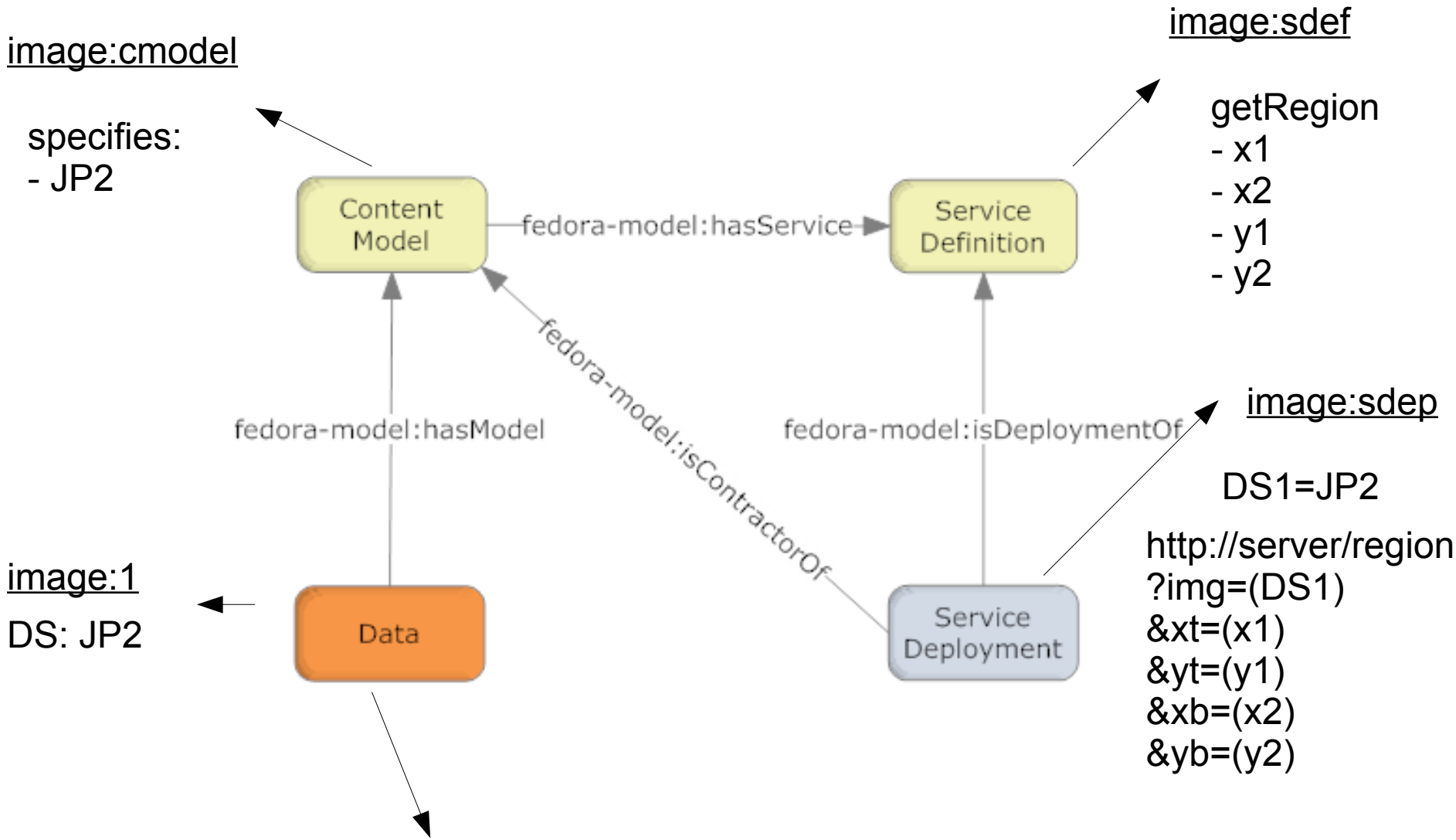
Fedora object model revisted

- Four basic classes of objects
  - Data objects
  - Content Model (CModel) objects
  - Service Definition (SDef) objects
    - Define the "profile" of services on objects
  - Service Deployment (SDep) objects
    - Define the implementation of those services

CMA objects


Content Model
fedora-model:hasService
Service Definition
fedora-model:isContractorOf
fedora-model:hasModel
fedora-model:isDeploymentOf
Data
Service Deployment

Adding methods

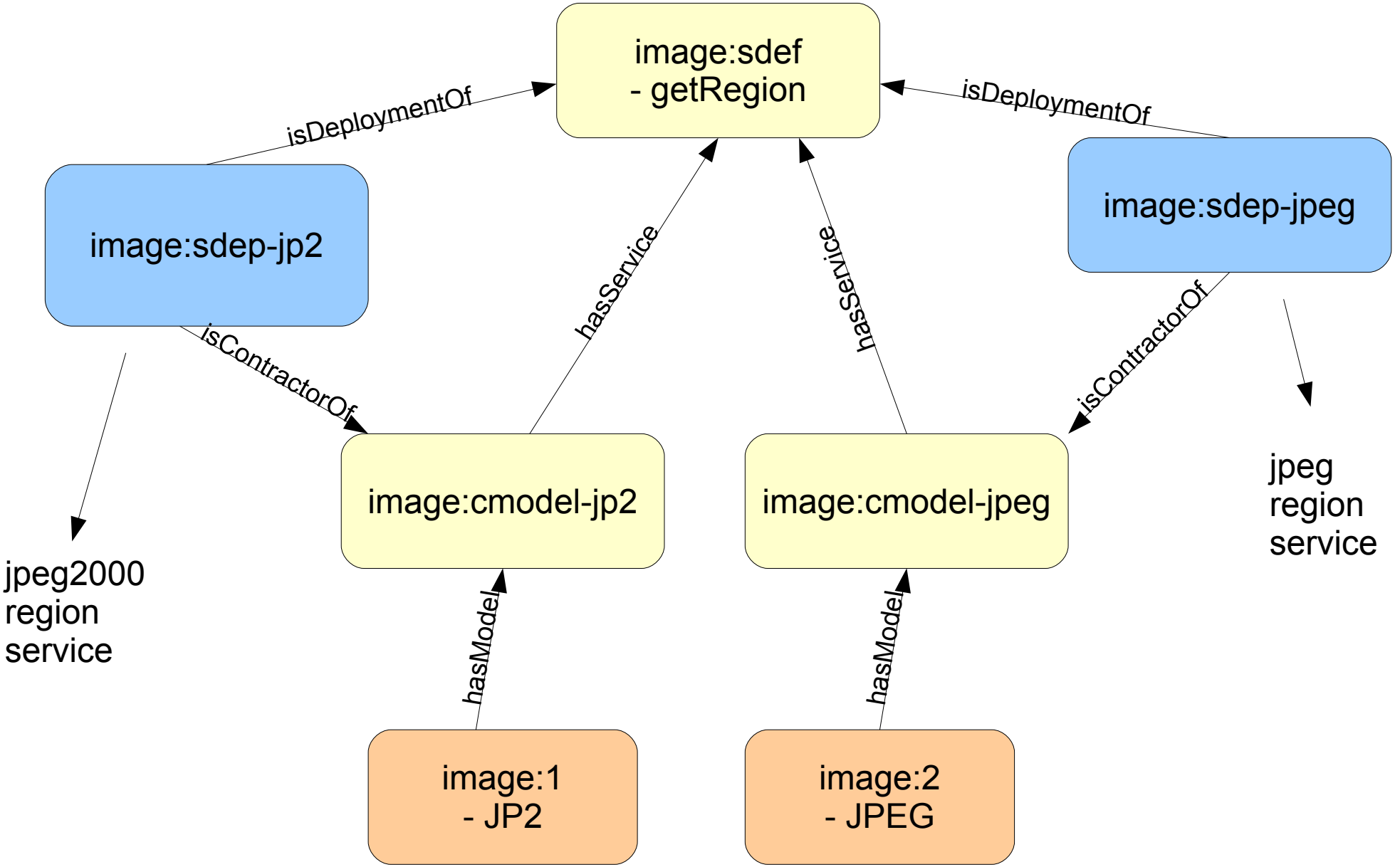


/fedora/objects/image:1/methods/image:sdef/getRegion?x1=10&y1=10&x2=20&y2=20

It's the SDef...

- ... that defines the URL of the method
- /fedora/objects/image:1/methods/**image:sdef**/getRegion?x1=10&y1=10&x2=20&y2=20
- The CModel is nowhere to be seen!
- Different CModels can have different method implementations
  - through different SDeps
- But the "profile" – the data object URL – is the same

Different implementations



Another use case

- "Navigation" services for user interface
- Collection/Member structures
  - eg Image1 -> isMemberOf -> ImageCollection
- Part/whole structures
  - eg Book1 -> hasPage -> Page1
- Standard services for your user interface
  - children/descendents
  - parents/ancestors
- Implemented using different RI queries

(Show Actual Datastreams)

Let's look at an example ...

- imagined:16
- Data object with an image viewer service

Data Object

- imagined:16
- http://riri.islandora.ca:8080/fedora/objects
- Take a look at the object profile
- Identify the content model object
- Take a look at the list of methods
- Identify the SDef from the method list
- Take a look at RELS-EXT
  - the content model relationship
  - relationship to the SDef

SDef

- ilives:viewerSdef
- Take a look at RELS-EXT
  - hasModel relationship – it's an SDef
- Take a look at METHODMAP
  - See the method defined (no parameters for this one)

SDep

- Identify the SDep
  - <sdep> <fedora-model:isDeploymentOf> <sdef>
  - <sdep> <fedora-model:isContractorOf> <cmodel>
- RISearch query

```
select $s
from <#ri>
where
$s <fedora-model:isDeploymentOf> <info:fedora/ilives:viewerSdef>
and
$s <fedora-model:isContractorOf> <info:fedora/islandora:mapCModel>
```

SDep

- islandora:viewerSdep-slideCModel
- Take a look at RELS-EXT
  - hasModel – it's an SDep
  - isDeploymentOf – the SDef
  - isContractorOf – used by a few CModels
- Take a look at METHODMAP
  - method
  - parameters

SDep

- islandora:viewerSdep-slideCModel
- Take a look at DSINPUTSPEC
  - One datastream input
- Take a look at WSDL
  - find the service URL
  - and the operation definition
    - with the parameters
    - including the datastream

Fedora Methods
Tips and Tricks - Steve

Tips and Tricks

- At least one datastream binding must be specified
  - (this may change)
  - So specify one, even if it is not used (eg dummy binding to DC)
- Supplying the PID to the external service
  - Default input parameter in SDep, with the value of "PID"
-

Tips and Tricks

- Datastream Mediation
  - Can be turned on and off
  - If it is on, URL of datastream is not passed directly, instead a "proxy" temporary URL is minted
  - Service gets unauthenticated access (temporarily)
  - If it is turned off, service will need to authenticate

Fedora Methods
Q&A, Use cases, discussions