Digital Repositories and Data Models

DORSDL2 Workshop
ECDL Conference
Aarhus, 2008

Asger Blekinge-Rasmussen
Kåre Fiedler Christiansen
Program

- Introduction
- A look at Fedora objects
- Uses for formalized Content Models
- Schema languages
- Introducing our Content Model language
- Extending Fedora Content models
- Describing Ontologies
- Inheritance
- Rounding up
Introduction

- Shift of paradigm
  - Catalog cards -> interlinked content
  - Heterogeneous collections in the same repository
- A single metadata model will not do
- Need for a Data Model Description Language
- Extension of Fedora 3.0 Content Models
  - Precise description of XML datastreams
  - Ontology for relations
A look at Fedora Objects

- Interrelated objects
- Datastreams in objects
- Content Models
Uses for formalized Content Models

- Validation
- Integrity Checking
- Preservation
- Automated tools
  - Think of Content Models as Java interfaces
- Exchange of data models
- Adherence to Metadata Standards
  - Formalized Content Models as Metadata Standards
Schema Languages

- Datastreams
  - XML
    - XMLSchema
  - Non-XML
    - ?
- Relations
  - RDF
    - RDFS schema
    - OWL
Introducing our Content Model Language

- Do not restrict the expressibility of Content Models
- Keep with the Fedora paradigm
  - XMLSchema is stored with datastream definitions
  - Ontology is stored in a Content Model datastream
  - Content Models are local
    - Describe only properties about conforming objects
    - Describe all properties about conforming objects
  - Use a subset of OWL to describe subset of RDF
Extending Fedora Content models

- Current Fedora content models

```xml
<dsCompositeModel>
  <dsTypeModel ID="DC">
    <form MIME="text/xml"/>
  </dsTypeModel>
</dsCompositeModel>
```
Extending Fedora Content models

- New Fedora content models

```xml
<dsCompositeModel>
  <dsTypeModel ID="DC">
    <form MIME="text/xml"/>
    <extensions name="DOMS">
      <schema:schema type="xsd" datastream="DC_SCHEMA"/>
    </extensions>
  </dsTypeModel>
</dsCompositeModel>
```
Describing Ontologies

<owl:ObjectProperty rdf:about="fedora:hasPart"/>
<owl:Class rdf:about="info:fedora/demo:cm1">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="fedora:hasPart"/>
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="info:fedora/demo:cm2">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="fedora:hasPart"/>
      <owl:allValuesFrom rdf:resource="info:fedora/demo:cm2"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
Describing Ontologies

<owl:ObjectProperty rdf:about="fedora:hasPart"/>
<owl:Class rdf:about="info:fedora/demo:cm1">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="fedora:hasPart"/>
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="info:fedora/demo:cm2">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="fedora:hasPart"/>
      <owl:allValuesFrom rdf:resource="info:fedora/demo:cm2"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
Describing Ontologies

<owl:ObjectProperty rdf:about="fedora:hasPart"/>
<owl:Class rdf:about="info:fedora/demo:cm1">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="fedora:hasPart"/>
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="fedora:hasPart"/>
      <owl:allValuesFrom rdf:resource="info:fedora/demo:cm2"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
Describing Ontologies

- **OWL subset**
  - Subset of OWL Lite *if*
    - fedora:hasModel is understood to mean rdf:type
    - Relations between content models are not considered
      - allValuesFrom/someValuesFrom
      - cardinality/minCardinality/maxCardinality
Inheritance

- Inheritance might be defined as
  - \( CM\ A \text{ extends } CM\ B \)
  means
  - \( \text{Objects with } CM\ A \text{ also have } CM\ B \)

- Simply require that if
  - \(<CM\ A \ rdfs\:subClassOf \ CM\ B>\)
  then
  - \(<\text{Object } fedora\:hasModel \ CM\ A>\) and
    \(<\text{Object } fedora\:hasModel \ CM\ B>\)
Rounding Up

- We are using this data model language for
  - Auto-generated user interface (being built)
    - May require we restrict the XML schema language as well
  - Validation of data integrity
    - By ingest
    - Periodically

- Project homepage
  - http://wiki.statsbiblioteket.dk/domswiki
Rounding Up

- Thank you for listening

- Questions?

- This work has been funded by:
  - DEFF, Denmark's Electronic Research Library
  - State and University Library, Denmark