Fedora Ontologies

An OWL LITE ontology for Fedora 3.0 data objects based on the CMA

Warning This page is no longer updated. The project is moved to http://ecm.sourceforge.net

In the following these shorthands will refer to the following namespaces

- rdf - http://www.w3.org/1999/02/22-rdf-syntax-ns#
- rdfs - http://www.w3.org/2000/01/rdf-schema#
- owl - http://www.w3.org/2002/07/owl#
- fedora-model - info:fedora/fedora-system:def/model#
- doms-relations - http://doms.statsbiblioteket.dk/relations/default/0/1/#
- doms - Standard prefix for PIDs. It is not a namespace

Fedora is a store for digital objects. The exact way they are stored is not important for this discussion. What is important is that the Fedora digital objects have RDF relations to each other. I.e. the Fedora digital object repository can be modeled as an RDF graph.

There is one critical difference between Fedora digital objects and a normal RDF graph: Each Fedora object contains its own local bit of the graph. You cannot change the number or nature of the relations from object A, without editing object A.

An ontology for the Fedora repository should preserve this characteristic. The ontology must be spread out over the entire repository, and you should not be able to break the ontology in one place by changing it somewhere else. In other words, if you have two separate Fedora repositories, each described by separate ontologies, and you transfer them to the same repository, the combined ontology should describe the combined sets of objects. This leads to the first property on the ontology system:

1. The ontology must not make statements that are global for the entire repository, except for the declaration of the existence of a class or property

Fedora provides a "class" of objects called Content Models. These try to represent the classes of data objects, and if specified, contain the description of the data objects. These are the natural location to place the local ontology bits. But now we reach the first problems. The Content models are both the classes of data objects, but they are also objects themselves. In order to describe such duality of existence, the language needed is OWL FULL, or something with similar expressive power. Since such expressive languages are difficult to reason about by automated systems, we chose to use a more restricted version, called OWL LITE. This imposes the second property on the ontology system.

2. The ontology only describes the data objects. Content models are regarded as classes, never as objects

An ontology with implicit rules, properties or classes, could lead to some potential problems. When part of the ontology is derived from the whole ontology, the effects of changes to the ontology can become difficult to predict. Especially the removal or introduction of a class could affect the nature of other classes. In effect, this means that someone wanting to use the ontology must know the entire ontology, in order to extrapolate anything implicit, which is in conflict with property 1. To make this explicit, the third property is introduced:

3. The ontology must be locally complete, so that every local bit provides the complete description of its local area.

Fedora RDF relations

Fedora does not allow for the FULL RDF specification to be used in the repository. What it basically allows is that each object can have properties relating them to other objects (called relations), and literal properties. There can be no qualifiers on the properties.

There are a number of note-worthy issues about the way Fedora works with RDF. The first is that Fedora objects do not declare a rdf:type property. Instead they use a fedora-model:hasModel property to relate to a Content model. Unfortunately, OWL LITE regards the relations as "owl:ObjectProperty"'s, and "rdf:type" as an "rdf:Property". As they are different, you must use OWL FULL to define: fedora-model:hasModel rdfs:subPropertyOf rdf:type
So, in OWL LITE Content Models cannot be regarded as classes, in violation of property 2. But as this is all that prevents from using OWL LITE for the ontology, there are hackish ways around it. And thus is property 4 defined.

4. In data objects, all "fedora-model:hasModel" relations are to be regarded as "rdf:type" relations to the same subject

In Fedora, it is not required that the relations from an object actually refers to another object. For an ontology, this is problematic, in regards to the 3. property. As the Content Model define the class of an object, using a non-existing content model will mean that the class is implicitly defined. Likewise, if there exist a relationship between two objects, which is not defined in the ontology, the relationship is implicitly defined. As there is to be no implicit definition of properties/relations, all such must be defined in a content model. This impose the properties 5 and 6.

5. All fedora-model:hasModel relations must point to real Content models

6. All allowed relations must be defined in the ontology

Defining ontologies by OWL LITE

When the properties 2 and 6 are expressed in OWL, they become the property 7.

7. All Content models must contain OWL that define themselves as classes, and list all the allowed relations for their subscribing objects

A Fedora object consists of a number of datastreams. One datastream, RELS-EXT has been reserved for the Fedora rdf statements. We choose to reserve another datastream, ONTOLOGY, to contain the ontology definitions.

Just like Fedora only allows the "rdf:Description" tag in each object, we have chosen to similarly restrict what OWL tags can reside in a Content model. In fact, there are just three allowed elements inside the rdf:RDF tag; "owl:Class", "owl:ObjectProperty" and "owl:DatatypeProperty".

Each Content model must contain one and just one "owl:Class" element, about the Content model itself. In this element the ordinary OWL syntax can be used to place restrictions on the Properties. The allowed restrictions are:

- minCardinality (0-1)
- maxCardinality (0-1)
- cardinality (0-1)
- someValuesFrom
- allValuesFrom

You are not allowed to use the "rdfs:subClassOf" property to make the class a subclass of another Content model.

All relations allowed from a data object should be defined in at least one of its Content models, in the form of "owl:ObjectProperty". This is important, so I will say it again: The complete list of allowed relations in a data object is the set of ObjectProperties defined by it's Content models. A relation can be declared in multiple Content models, but if a Content model place restrictions on a relation, it must declare the relation itself. The reason behind this requirement is just property 3. Even through all the declaration of ObjectProperties are global for the repository, and thereby allowed for all objects in the repository, the demand is that each data object should be described by just the local Content models, ie. those it relates to through the "fedora-model: hasModel" relation.

Fedora will also allow an object to have literal properties. Such properties are defined by the "owl:DatatypeProperty" tag.

Looking at property 1 in the context of "owl:ObjectProperty", it becomes clear that range and domain are not allowed. This is unfortunately required. Since neither OWL nor Fedora provide a way to ensure that the same relation is not defined twice, it is entirely possible for two unrelated Content models in the repository to define the same property. Each part of the repository will be valid viewed locally, but when regarding the repository as a whole, the two different definitions will be combined. Having two domains for a property mean that the source must be of both types, not either, and likewise for range, and the repository as a whole will be invalid. To prevent the risk of such errors the use of domain and range are disallowed.

8. "rdf:range" and "rdf:domain" are not allowed on any properties.

Instead of "rdf:range", one should use the "allValuesFrom" restriction. This restriction defines a range for the property, but only in the given class. As such, the restriction will have no global effect. "rdf:domain" is just not nessesary. The property 3 implies that the ONTOLOGY in a Content Model should describe the local area, i.e. the objects subscribing to that content model. The result of this is that the domain, so to speak, of a property will always be the Content Model in which it was defined. But again, this restriction will have no global effect, the property defined somewhere else will have some other Content Model as its domain.

Example of a simple ontology

```
<!--RELS-EXT from Object_A1-->
<rdf:Description rdf:about="info:fedora/doms:Object_A1">
    <fedora-model:hasModel rdf:resource="info:fedora/doms:CM_A"/>
    <doms-relations:hasB rdf:resource="info:fedora/doms:Object_B1"/>
</rdf:Description>

<!--OWL-SCHEMA from CM_A-->
<owl:Class rdf:about="info:fedora/doms:CM_A">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty
                    rdf:resource="http://www.statsbiblioteket.dk/doms-relations/#hasB"/>
            <owl:cardinality
                    rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#integer">
                1
            </owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty
                    rdf:resource="http://www.statsbiblioteket.dk/doms-relations/#hasB"/>
            <owl:allValuesFrom rdf:resource="info:fedora:/doms:CM_B"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty
        rdf:about="http://www.statsbiblioteket.dk/doms-relations/#hasB"/>
```

An object A1 and a Content model CM_A is defined. There is one allowed relation for A1, the #hasB relation. Two restrictions are placed on this relation. There must be one, and just one such relation in A1, and it must refer to an object of class/Content model CM_B. In fact, A1 has one such relation, and it refers to the object B1, which follows below.

```
<!--RELS-EXT from Object_B1-->
<rdf:Description rdf:about="info:Fedora/doms:Object_B1">
    <fedora-model:hasModel rdf:resource="info:fedora/doms:CM_B"/>
</rdf:Description>


<!--OWL-SCHEMA from CM_B-->
<owl:Class rdf:about="info:Fedora/doms:CM_B">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty
                    rdf:resource="http://www.statsbiblioteket.dk/doms-relations/#hasA"/>
            <owl:allValuesFrom rdf:resource="info:fedora:/doms:CM_A"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty
        rdf:about="http://www.statsbiblioteket.dk/doms-relations/#hasA"/>
```

Here is the the object B1, and it's Content model CM_B. There is one allowed relation from a B1, the #hasA relation. There is just one restriction on this relation, that it must refer to something of class/Content model CM_A. No cardinality restriction is defined, so B1 does not need to have the relation, and in fact, it does not have it.

Content Model inheritance

While this model does not require it, it does lend itself readily to a inheritance system for Content models. In this section, such an amendment will be outlined.

There are two apparant ways to specify inheritance. It could be added as "rdfs:subClassOf" (or a subproperty of this) tag inside the "owl:Class" element in the ONTOLOGY datastream or it could be added as a relation in the RELS-EXT datastream for the Content models. Both ways will work, but there is really no advantage in using the ONTOLOGY datastream (to specify the inheritance)over the RELS-EXT datastream. When the relation is defined in RELS-EXT, it is indexed by Fedora, and you can make triple-store queries about it.

The problem about "rdf:type" not being extensible is also relevant for "rdf:subClassOf". There are two ways to model the inheritance relations in RELS-EXT. Firstly, you can use "rdfs:subClassOf" directly. Alternatively, you can define some other property to have the same semantic meaning, just like we did with "fedora-model:hasModel" and "rdf:type". We have chosen the last option, and reserved a property "doms-relations:extendsModel", to represent Content Model inheritance.

Inheritance of Content models does have some implications for data objects. In the standard Fedora worldview, an object does not 'belong to' a Content model, unless it has a "fedora-model:hasModel" relation to it. For now, we want to maintain this invariant. So, in addition to having "doms:extendsModel" between the Content models, a data object must have "fedora-model:hasModel" relations to ALL the Content models inherited by it's primary Content models.

The OWL schema in a Content model will now not reside solely in the ONTOLOGY datastream. As the information about any parent classes is placed in the RELS-EXT datastream, it now has to be included in the ontology. But this provides a new problem; an OWL class is not allowed to have relations to other objects, unlike an OWL individual. So, only the "doms:extendsModel" and the "fedora-model:hasModel" relation from the RELS-EXT datastream should be used when validating, the other relations should be disregarded.

An example using inheritance

To aid in understanding, an example using inheritance have been included below.

```
<!--RELS-EXT from Object_A1-->
<rdf:Description rdf:about="info:fedora/doms:Object_A1">
    <fedora-model:hasModel rdf:resource="info:fedora/doms:CM_A"/>
    <fedora-model:hasModel rdf:resource="info:fedora/doms:CM_C"/>

    <doms-relations:hasB rdf:resource="info:Fedora/doms:Object_B1"/>
</rdf:Description>

<!--RELS-EXT from Object_A2-->
<rdf:Description rdf:about="info:Fedora/doms:Object_A2">
    <fedora-model:hasModel rdf:resource="info:Fedora/doms:CM_A"/>
    <fedora-model:hasModel rdf:resource="info:Fedora/doms:CM_C"/>

    <doms-relations:hasB rdf:resource="info:Fedora/doms:Object_B1"/>
</rdf:Description>
```

Two data objects, A1 and A2 have been declared. They both have the CM_A and the CM_C Content model.

```
<!--RELS-EXT from CM_C-->
<rdf:Description rdf:about="info:fedora/doms:CM_C">
    <fedora-model:hasModel
            rdf:resource="info:Fedora/fedora-system:ContentModel-3.0"/>
</rdf:Description>


<!--OWL from CM_C-->
<owl:Class rdf:about="info:fedora/doms:CM_C">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty
                    rdf:resource="http://www.statsbiblioteket.dk/doms-relations/#hasB"/>
            <owl:allValuesFrom rdf:resource="info:fedora:/doms:CM_B"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty
        rdf:about="http://www.statsbiblioteket.dk/doms-relations/#hasB"/>
```

Here the CM_C Content model has been declared. It extends no other Content model. It defines the relation doms-relations:hasB. The definition is required in order to impose a restriction on the relation. The restriction is that the relation must refer to objects from Content model CM_B. As could be seen before the A1 and A2 objects both refer to a B1 object, which may or may not be of the class CM_B.

```
<!--RELS-EXT from CM_A-->
<rdf:Description rdf:about="info:fedora/doms:CM_A">
    <hasModel rdf:resource="info:fedora/fedora-system:ContentModel-3.0"
              xmlns="info:fedora/fedora-system:def/model#"/>
    <doms:extendsModel rdf:resource="info:fedora/doms:CM_C" />
</rdf:Description>

<!--OWL from CM_A-->
<owl:Class rdf:about="info:fedora/doms:CM_A">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty
                    rdf:resource="http://www.statsbiblioteket.dk/doms-relations/#hasB"/>
            <owl:cardinality
                    rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#integer">
                1
            </owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty
        rdf:about="http://www.statsbiblioteket.dk/doms-relations/#hasB"/>
```

CM_A declares itself as a subtype of CM_C. CM_A redeclares the relation doms-relations:hasB and defines a cardinality of 1. So, Objects of CM_A must have just one relation doms-relations:hasB, and it must point to an object of class CM_B. Objects just of class CM_C can have 0-* such relations, but they must still all point to objects of class CM_B.

```
<!--RELS-EXT from Object_B1-->
<rdf:Description rdf:about="info:fedora/doms:Object_B1">
    <fedora-model:hasModel rdf:resource="info:Fedora/doms:CM_B"/>
    <fedora-model:hasModel rdf:resource="info:Fedora/doms:CM_C"/>

    <doms-relations:hasA rdf:resource="info:fedora/doms:Object_A1"/>
</rdf:Description>
```

As can be seen, object B1 is of Content models CM_B and CM_C. As such they relations in A1 and A2 are valid, as B1 is of class CM_C. B1 has one relation. Whether or not they are valid should be answered by looking at the Content models CM_B and CM_C, as they describe B1.

```
<!--RELS-EXT from CM_B-->
<rdf:Description rdf:about="info:fedora/doms:CM_B">
    <fedora-model:hasModel
            rdf:resource="info:fedora/fedora-system:ContentModel-3.0"/>
    <doms-relations:extendsModel rdf:resource="info:fedora/doms:CM_C" />
</rdf:Description>

<!--OWL from CM_B-->
<owl:Class rdf:about="info:Fedora/doms:CM_B">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty
                    rdf:resource="http://www.statsbiblioteket.dk/doms-relations/#hasA"/>
            <owl:minCardinality
                    rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#integer">
                1
            </owl:minCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty
                    rdf:resource="http://www.statsbiblioteket.dk/doms-relations/#hasA"/>
            <owl:allValuesFrom rdf:resource="info:Fedora:/doms:CM_A"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty
        rdf:about="http://www.statsbiblioteket.dk/doms-relations/#hasA"/>
```

Here is the definition of CM_B. As can be seen, it extends CM_C. Objects of class CM_B must have one or more doms-relations:hasA relations, and they must all refer to objects of class CM_A.

Since B1 is of CM_C, it may have zero or more doms-relations:hasB relations to objects of CM_B. It has zero such relation, which is allowed. It has one doms-relations:hasA a relation to A1, which is enough to fulfill the restrictions.