AIC Use Case: Content Modeling

Title (Goal)	Content modeling
Primary Actor	Developer
Scope	Application
Level	Summary
Author	Stefano Cossu
Story (A paragraph or two describing what happens)	Configuration-based content modeling that allows specifying services and schemata related to specific resource types

I want to be able to CRUD complex resources with a single or a minimal set of requests, without:

- Performing complex and repetitive tasks on the client side, such as querying an UID minter, transforming resources to generate standard derivatives, extract metadata, etc.
- Knowing everything about the schema that the resource metadata are stored in
- Rewriting complex implementation routines for each client connecting to my Fedora repo.

I should specify a "content model" (e.g. an RDF type) for each resource that I CRUD and have predefined metadata schemata for ingestion and retrieval, customizable event hooks related to individual HTTP methods, and validation rules (see related use case)

I want to be able to retrieve some machine-readable documentation for each content type, for individual HTTP methods, e.g. schema information indicating which properties are expected for a specific HTTP method on a specific "content model", data type and cardinality of each property, etc.

Ideally, I would be able to achieve all of the above without using any code, i.e. by wiring content models to services and schemata by means of transparent configuration files.

Example

For a given “myns:Image” content model the service configuration would define the following actions related to distinct HTTP methods:

- GET:
    - Return a representation of the resource and related resources according to a “default” transform program defined for myns:Image and an output format
    - This may include complex networks involving SPARQL or Solr queries on the indexes
    - The transform program name is indicated in a request parameter; the output format as an "Accept" request header
    - Image metadata are retrieved from a Solr or triplestore index, the image content can come straight from the Fedora repo or a server performing a transformation on the original resource (e.g. IIIF-compatible image server).
- POST:
    - Create a new UID from an external UID minter service;
    - create a pcdm:Object resource and populate it with metadata provided by a JSON object in the “metadata” POST field;
    - add the new generated UID;
    - assign the resource the “myns:Image” rdf type;
    - create pcdm:Object resource and assign it a “myns:Instance” rdf type;
    - create a “myns:hasOriginalInstance” relationship between the myns:Image and the myns:Instance resources;
    - Create a LDP-NR resource from the “original” bitstream provided by POST
    - Create a “pcdm:hasFile” relationship between the myns:Instance and the LDP-NR
    - Repeat this for other possible binaries provided by the request
    - Optionally wrap the whole operation in a transaction and roll back if a step fails.