Customized OAI-ORE and OAI-PMH Exports of Compound Objects for the Fedora Repository

Alessia Bardi[1,2], Sandro La Bruzzo[1], and Paolo Manghi[1]

[1] Consiglio Nazionale delle Ricerche
Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
name.surname@isti.cnr.it
[2] Dipartimento di Ingegneria dell'Informazione, Università di Pisa
alessia.bardi@for.unipi.it

Abstract. Modern Digital Library Systems (DLSs) are based on document models which surpass the traditional payload-metadata document model to incorporate further entities involved in the research life-cycle. Such DLSs manage graphs of interconnected objects, hence offer tools for the creation, visualization and exports of such graphs. In particular, objects in the graph are exported via standard OAI-ORE and OAI-PMH protocols, encoded as (XML) "packages of interlinked information objects", also known as compound objects. Fedora is a well-known repository platform, designed to support the realization of DLSs implementing modern document models. To date, Fedora does not provide tools to customize compound object exports from DLS object graphs. This paper presents Fedora-OAIzer, an extension of Fedora which allows DLS developers to customize the structure of compound objects to be exported from a given DLS document model – expressed in terms of Fedora Content Models – and to select the OAI protocol of preference. In order to prove the completeness of the approach, Fedora-OAIzer is compared to other solutions for exporting compound objects from Fedora repositories.

1 Introduction

In the past, Digital Library Systems (DLSs) adopted "traditional" document models representing collections of payloads of digitized or born-digital material (e.g., publications, multimedia files) and their digital descriptions (i.e., metadata records). "Modern" document models enhance the traditional document model to incorporate further entities involved in the research life-cycle and semantic relationships. Consequently, modern DLSs manage graphs of interconnected information objects, called *object graph*, rather then flat collections of objects. For example, a "traditional" publication-metadata document model can be enriched with further contextual information, such as the used and generated research data.

The complexity of modern document models introduces a number of challenges concerning the way graphs of information objects are displayed, encoded, and exported across different DLSs. In particular, sub-parts of the object graph are exported, rather than each single information object or the object graph in its whole, in order to disseminate a package containing semantically related information objects. Such packages, called *compound objects*, are usually encoded in

XML or other machine-readable formats and exported via standard protocols. The most used standard protocols are promoted by the Open Archives Initiative [6]: the OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) [4] and OAI-ORE (Open Archives Initiative Object Reuse and Exchange) [5].

The Fedora Repository [3] is a well-known platform supporting the realization of DLS. Its data model is designed to represent modern DLS document models thanks to its graph-oriented primitives. At the time of writing, Fedora features a non-customizable OAI-PMH publisher[1] which exports only Dublin Core metadata records. An additional module supports the OAI-PMH exports of datastreams with different XML formats. Two plugins for OAI-ORE export are also available online: oreprovider[2] and Fedora2ORE[3]. Both plugins do not allow to fully customize the structure of the exported compound object.

This paper presents Fedora-OAIzer, an extension of Fedora for the export of compound objects conforming to a given portion of the underlying DLS document model through the OAI-PMH or OAI-ORE protocols. Fedora-OAIzer implements a mechanism based on the concept of "OAI view" of a Fedora document model. An OAI view is the sub-structure of the document model that developers select to customize the shape of the compound objects to export. OAIzer interprets OAI views to automatically deploy web APIs capable of exporting compound objects compatible with the given structure and according to the preferred OAI protocol.

Outline Section 2 introduces basic concepts and defines the addressed problem. Section 3 describes the Fedora repository platform and the available tools for exporting compound objects via OAI-PMH and OAI-ORE. Section 4 presents Fedora-OAIzer and its approach that enables developers to customize compound objects by selecting sub-parts of the document model at hand. We conclude and compare Fedora-OAIzer to other existing solutions in Section 5.

2 Document models, Digital Library Systems and compound objects

A Digital Library System (DLS) [2] is a DL-oriented software serving a particular DL community. A DLS offers functionalities for the management, access and dissemination of graphs of information objects whose structure is defined by a data model called document model. A document model is a formal definition of types of entities and relationships that a Digital Library (DL) wants to manage. An entity type typically describes properties of objects in terms of name, cardinality and value type. A relationship type usually include a semantic label expressing the nature of the association and the types of entities allowed as sources and targets of the relationship. Figure 1 shows a document model defining three types of entities (Article, PDF, and Data) and the available relationships (HAS_PDF, USES, USED_BY, GENERATES, GENERATED_BY).

[1] Basic OAI-PMH Provider, `https://wiki.duraspace.org/display/FEDORA36/Basic+OAI-PMH+Provider`

[2] oreprovider Fedora module, `http://oreprovider.sourceforge.net/`

[3] Fedora2ORE, `http://trac.eco4r.org/trac/eco4r/wiki/Fedora2ORE`

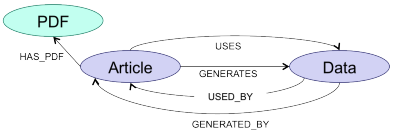

Fig. 1. A modern document model: articles linked with research data

For example, a DLS adopting the document model in fig.1, manages graphs of information objects as that in Fig. 2.

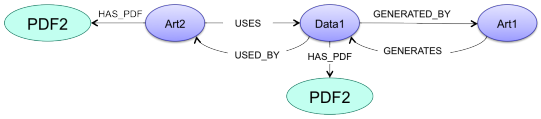

Fig. 2. An example of object graph

To clarify, it is possible to compare the above concepts with similar concepts in the relational database world. An entity-relationship model in the database world corresponds to a DLS document model. DLSs are comparable to applications realized on top of Relational Data Base Management Systems.

Since DLSs manage graphs of information objects, it is important to define the granularity of the data to export. Indeed, exporting each single information object separately (for example, exporting Art1 of Fig. 2 without the generated data) leads to a loss of contextual information. Related information objects should be packaged and exported together as a single compound object, capable of maintaining all semantic relationships involving the packaged objects.

In order to exchange and re-use compound objects, interoperability issues must be tackled. The Open Archives Initiative [6] defines two standard protocols for data interoperability and information reuse: OAI-PMH Open Archives Initiative Protocol for Metadata Harvesting [4] and OAI-ORE Open Archives Initiative Object Reuse and Exchange [5]. Both protocols allow to export compound objects of a DLS, but they adopt two radically different approaches. OAI-PMH is meant for the export of the descriptive metadata in the form of XML files; OAI-ORE is meant for the export of web-interpretable RDF representations of so-called aggregations, which are special web resources encoding compound objects.

OAI-PMH provides an application-independent interoperability framework based on metadata harvesting. Its data model has four main elements:

- Resource: an object described by one or more metadata records.
- Metadata record: XML data describing a resource. Each metadata record has its metadata format, often referred to as the XML Schema.
- Item: container of metadata records describing one resource. Each item must have at least one Dublin Core metadata record.

– Set: optional element used to group items.

In order to export via OAI-PMH, DLSs set up an OAI-PMH publisher capable of exporting a set of XML files encoding compound objects. Well-known metadata formats that can be adopted are METS[9] and XML-DIDL[8].

OAI-ORE defines standards for the description and exchange of Web resources called aggregations. An aggregation is a Web resource with its own identity and it represents a group of related Web resources.

The OAI-ORE protocol does not define an exchange protocol, but only a data model, while suggesting exchange formats such as XML/RDF and ATOM feeds. The OAI-ORE model captures four type of resources:

– Aggregation: resource that groups other resources, called aggregated resources.
– Aggregated resource: resource that belongs to an aggregation, that is the ORE representation of an information object in a compound object.
– Resource map: serializable description of an aggregation. A resource map:
  • lists the aggregated resources;
  • has properties about the aggregation and its aggregated resources, e.g., relationships among aggregated and other external resources.
– Proxy: resource that allows to assert relationships among aggregated resources in the context of one specific aggregation.

Among the functionalities offered by a DLS, we usually find support for data exchange according to the OAI standard protocols. However most DLSs do not provide means to customize the structure of the compound objects, but they rather fix statically the mapping between the document model and the OAI-PMH and OAI-ORE data models.

3 Fedora and exports of compound objects

Fedora (Flexible Extensible Digital Object Repository Architecture) is a well-known repository platform, designed to support the realization of DLSs. Its data model supports the representation of labelled graphs of information objects. Fedora manages objects of different kinds. "Fedora Data Objects" (FDOs) represent information objects. A FDO is composed by the following parts: an XML Dublin Core metadata record, a list of local or remote files called "datastreams", and a list of relationships to other FDOs. The latter are serialized as RDF/XML[7] into a special datastream called RELS-EXT. "Fedora Content Models" (FCMs) are special objects devised for the definition of a document model in a Fedora instance. FCMs define constraints on the structure of FDOs, declaring which are the mandatory datastreams, relationships and operations (i.e., Web Service methods) of the FDOs compliant to that FCM. Figure 3 shows how the document model in Fig. 1 can be represented in terms of FCMs. CM_article is the content model for the class article and defines one mandatory datastream called ART of mime type PDF. CM_data is the content model for the class data and defines two mandatory datastreams: one for binary content called DATA, the other for the XML

Listing 1.1. Excerpt from the FCM for the Article class: the ONTOLOGY datastream

```
<!-- ONTOLOGY datastream for allowed relationships -->
<foxml:datastream CONTROL_GROUP="X" ID="ONTOLOGY" STATE="A" VERSIONABLE="true">
  . . .
      <rdf:RDF>
        <owl:Class rdf:about="ns:CM_article">
          <!-- Objects of this class can have the following relations -->
           <owl:ObjectProperty rdf:about="uses"/>
          <owl:ObjectProperty rdf:about="generates"/>
          <!--Both relations must have objects compliant to the CM_data content model
               as targets. -->
          <rdfs:subClassOf><owl:Restriction>
              <owl:onProperty rdf:resource="#uses"/>
                <owl:allValuesFrom
                  rdf:resource="ns:CM_data"/>
               </owl:Restriction></rdfs:subClassOf>
          <rdfs:subClassOf><owl:Restriction>
          <owl:onProperty rdf:resource="#generates"/>
                <owl:allValuesFrom
                  rdf:resource="ns:CM_data"/>
           </owl:Restriction></rdfs:subClassOf>
         </owl:Class>
       </rdf:RDF>
. . .
```

description of the data in DDI format[1]. By default, Fedora also includes one mandatory XML datastream called DC for metadata in Dublin Core format. The arrows between the two models represent the available semantic relationships as they are declared in the ONTOLOGY datastream of both content models. Listing 1.1 is an excerpt of the ONTOLOGY datastream for CM_article.

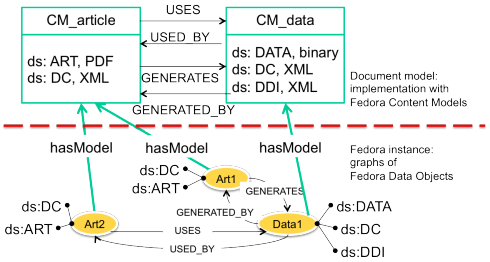

Fig. 3. Fedora content models and data objects example

3.1 Fedora and OAI

The Fedora data model is an expressive data model because its primitives allow to represent graph of information objects without imposing pre-defined structural or semantic constraints. Nevertheless, boundaries of compound objects are fixed, because Fedora's concept of compound object is that of an aggregation of datastreams. This means that in Fedora the notion of compound object matches the

definition of a Fedora Object. Given the instance in Fig. 3, each of Art1, Art2 and Data1 are considered by the system as distinct compound objects. Considering a set of interconnected Fedora Objects as one compound object is not possible in Fedora without the realization of a new logic layer on top of it.

The rest of this section describes four existing solutions for the export of compound objects from a Fedora instance.

OAI-PMH Providers Basic OAI-PMH Provider[4] is the built-in OAI-PMH provider for Fedora. It exports the mandatory DC datastream of each FDO. For the export of datastreams with other metadata formats, then the additional OAI Provider Service[5] module is required. OAI Provider Service supports any metadata format available through datastreams and interprets relationships with a given name to set up OAI Sets. Both tools provide a static mapping from the Fedora data model to the PMH data model. A FDO is mapped into an OAI-item, XML datastreams are mapped into metadata records.

OAI-ORE Providers OREprovider[6] enables the export of FDOs as OAI-ORE aggregations with an object-oriented approach. It implements a static mapping from the Fedora data model and the OAI-ORE data model. Datastreams are mapped into ORE aggregated resource. For the generation of ORE aggregations there are two modes available. In "annotation mode" FDOs must be annotated with relationships that assert the identifier of the target ORE aggregation and the datastreams to be mapped into aggregated resources. In "auto-creation" mode each FDO is mapped into one ORE Aggregation, whilst each of its datastreams is mapped into one aggregated resource. In "autocreation mode" the tool is easy to use and no alteration has to be done to an existing repository. However, if a DLS developer wants to have control over the exported objects, FDOs must be appositely annotated, hence the document model must be aware of the special relationships exploited by OREprovider. Furthermore, the static mapping does not include the concept of ORE proxy and the tool does not provide any support for the modelling of relationships among aggregated resources.

The Fedora2ORE[7] tool adopts a navigation-oriented approach for the export of FDOs as OAI-ORE aggregations. ORE aggregations consist of one FDO together with the objects reachable by navigating its relationships up to a given depth. Fedora2ORE traverses the object graph starting from an object with a given identifier according to a variant of the breadth first search. A resource map is created to represent the visited sub-graph. The behaviour of the traversal can be customized by statically specifying in a configuration file which relationships, datastreams, FDOs are to be ignored. Each node of the resulting sub graph is an Aggregated Resource. Fedora2ORE is independent from DLS applications because there is no need to define special relationships as for OREprovider. It generates aggregations by following relationships between objects, but DLS developers can only define the boundaries of aggregations in terms of navigation depth rather

[4] `https://wiki.duraspace.org/display/FEDORA35/Basic+OAI-PMH+Provider`
[5] `https://wiki.duraspace.org/display/FCSVCS/OAI+Provider+Service+1.2.2`
[6] `http://oreprovider.sourceforge.net/index.html`
[7] `http://trac.eco4r.org/trac/eco4r/wiki/Fedora2ORE`

than in terms of their preferred document model sub-structure. Furthermore, relationships among FDOs are not mapped into the OAI-ORE model, hence most of the semantics of the compound object is lost.

4 Fedora-OAIzer

Fedora-OAIzer is an extension for Fedora that allows to customize exports of compound objects via OAI-ORE and OAI-PMH protocols. As shown in Fig. 4, Fedora-OAIzer realizes a new layer on top of Fedora, capable of dynamically deploying OAI-PMH or ORE-ORE interfaces.

Fedora-OAIzer exploits the information about the Fedora Content Models to construct a representation of the document model of the DLS. We denote such a representation as *entity graph*. DLS developers can choose granularity, shape and properties of the compound objects to export by selecting interesting nodes and edges from the entity graph. Since the entity graph is a representation of a Fedora document model, the selected parts are a sub-structure of the document model. That sub-structure is called *OAI view* of a Fedora document model. The OAI view corresponds to the structure of the compound objects to export. OAIzer interprets OAI views to automatically setup OAI-ORE and OAI-PMH repositories. Repositories are here intended as ORE or PMH APIs available at a given URL, dynamically generated after an OAI view interpretation.

OAIzer exploits the built-in capabilities of Fedora, namely the REST APIs and the triple store, hence it is possible to plug Fedora-OAIzer in any standard Fedora instance.

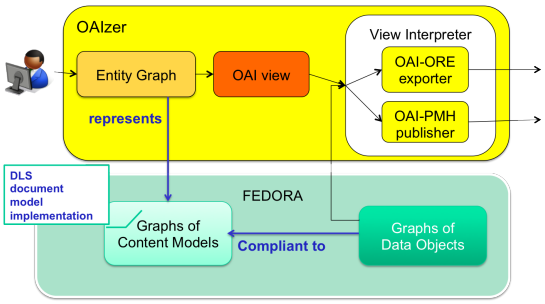

Fig. 4. View mechanism and OAI tools

4.1 Generation of the entity graph

When a DLS is based on Fedora, then developers define the document model in terms of Fedora Content Models (FCMs). Figure 5 shows an example of entity graph representing the document model of Fig.3: nodes represent classes of information objects, that is Fedora Content Models. Nodes are annotated with

information about datastreams. Edges represent allowed relationships between objects of the connected classes.

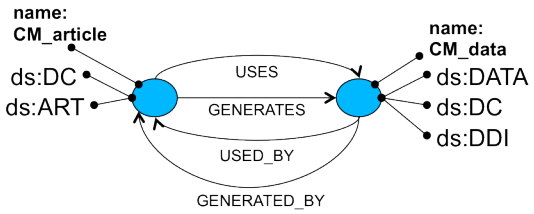

Fig. 5. An example of entity graph

Fedora-OAIzer generates the entity graph by performing the following macro steps:

1. Creation of one node of the entity graph for each FCM in the Fedora instance. The list of existing FCMs can be obtained by querying the triple store [8].
2. Obtain the full XML representation of each FCM using the REST API[9]. Information in that file is used in the next steps.
3. Enrichment of nodes with properties. Properties of a node reflect the declarations of datastreams in the corresponding content model. Such information is extracted from the standard XML datastream named DS-COMPOSITE-MODEL.
4. Generation of edges. If the FCM declares a relationship to another content model, then an edge is created between the nodes corresponding to the content models involved in the relation. The edge is labelled with the name of the relationship as it is declared in the ONTOLOGY datastream (see Listing 1.1).

4.2 Definition of the view

OAIzer provides a graphical user interface where DLS developers can see the generated entity graph and define their OAI view, that is the shape of the compound objects to export by choosing the interesting nodes, properties, and edges.

The DLS developer first chooses the root node of the OAI view. The view interpreter navigates the Fedora object graph starting from every object compliant to the content model represented by the root node. After the selection of the root, the DLS developer performs iteratively the following steps until the OAI view is completed according to requirements:

[8] `select ?o from <#ri> where ?o <fedora-model:hasModel> <info:fedora/fedora-system:ContentModel-3.0>`

[9] `http://<fedoraServer>/objects/<id>/objectXML`

– select one or more properties of the current node. The property names match the names of the corresponding Fedora datastreams: by selecting a property, the developer includes the corresponding datastream in the ORE resource relative to the current node.
– select one or more edges from the forward star of the current node. Each edge is labelled with the name of the corresponding Fedora relationship. If the DLS developer selects an edge, the target node is also included in the OAI view and an ORE relation is added between the ORE resources corresponding to the current and the target node.

The OAI view is eventually serialized into a formal language for the view interpreter.

Figure 6 shows a possible OAI view defined over the entity graph in Fig. 5. Compound objects are rooted in the class CM_article and include the DC datastream of the article, the DC and DDI datastreams of the research data objects reachable through relationships labelled with USES, and the DC datastream of articles reachable from those research data objects via relationships labelled GENERATED_BY.

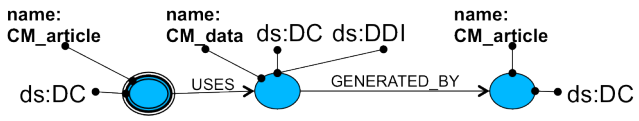

Fig. 6. An example of OAI view

4.3 Interpretation of the view

The View Interpreter is the component of OAIzer that deploys OAI-PMH and OAI-ORE APIs for the dissemination of compound objects whose structure is defined by an OAI view. The OAI view defines a "root" content model: each object compliant to the root content model is the entry point for an *instance of the OAI view*. An instance of the view is a sub-graph of the object graph. It includes all objects and relationships of one compound object.

The interpreter performs a visit on the FDO's graph for each entry point in order to get all FDOs that form an instance of the view. Moreover, the interpreter processes the FDOs of the view instance to keep track of the semantic relationships between them.

Figure 7 shows two instances of the view in Fig. 6 over the Fedora instance in Fig. 3.

For the customization of an OAI-PMH publisher, the interpreter maps the view into an OAI-PMH Set. Each instance of the view is mapped into an OAI-PMH Item with at least two metadata formats: OAIzer-XML and DC. OAIzer-XML is an idiosyncratic format for the representation of compound objects. OAIzer provides a default mapping from OAIzer-XML to DC in order to generate the DC

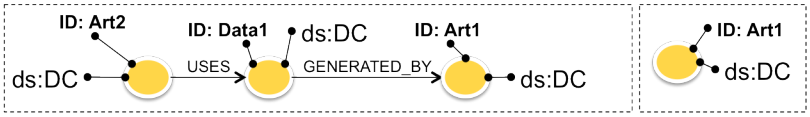

Fig. 7. Instances of the OAI view

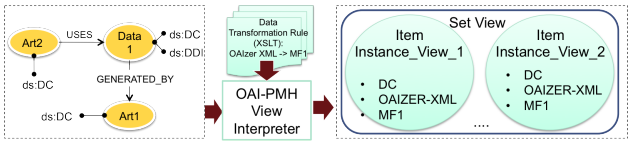

Fig. 8. Examples of OAI-PMH exports

records and be fully compliant to the OAI-PMH protocol. More metadata formats can be added by providing customized XSLT transformations from OAIzer-XML to the target metadata formats (see fig.8).

For the customization of the OAI-ORE exporter, the interpreter creates for each instance of the view one ORE aggregation together with its corresponding resource map. For each FDO of the instance, an aggregated resource and an associated proxy are created. The aggregated resource is the OAIzer-XML representation of the FDO, including the datastreams selected in the view. Finally, the interpreter processes the FDOs of the view instance to add semantic relationships between aggregated resources. At this aim, the interpreter exploits the ORE proxy entities. The interpreter connects two proxies with a relationship `r` if the Fedora triple store contains the triple: `fdo1 r fdo2` where `fdo1` and `fdo2` are the identifiers of the FDOs corresponding to the ORE proxies (see Fig. 9)

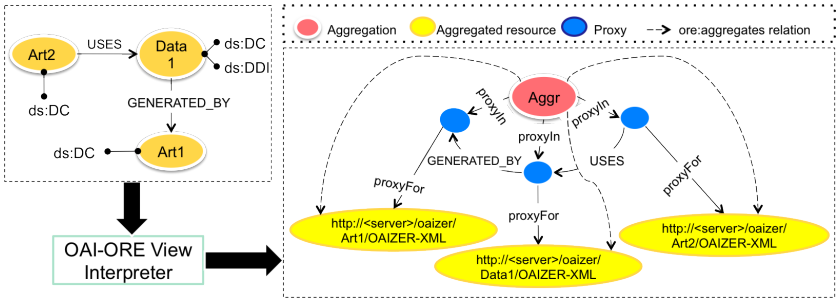

Fig. 9. Examples of OAI-ORE Aggregations

5 Conclusion

We discussed how the evolution of today's Digital Library Systems (DLSs) and the complexity of document models to represent led to data interoperability chal-

lenges. We addressed issues regarding the exchange of compound objects, i.e., packages of interlinked information objects with their own identity, among different DLSs via the standard protocols OAI-ORE and OAI-PMH.

We presented Fedora-OAIzer, a tool for the customization of OAI-PMH and OAI-ORE exports of compound object from Fedora repositories. Fedora-OAIzer creates a layer on top of Fedora in order to allow DLS developers to select the shape and boundaries of the compound objects to export. Fedora-OAIzer analyses a Fedora instance and infers the document model at hand from the Fedora content models. The document model is represented as an entity graph from which developers can select a subset of nodes and edges to define the OAI view. The OAI view defines the structure of the compound objects to export. Fedora-OAIzer then interprets the OAI view and, on demand, deploys OAI-ORE and OAI-PMH APIs delivering compound objects whose structure complies with the view definition.

Figure 10 summarizes the comparison among Fedora-OAIzer and the other existing solutions for OAI-PMH and OAI-ORE exports we described in Sect. 3.1.

	OAizer	Basic OAI-PMH Provider	OAI Provider	OREProvider	Fedora2ORE
Fedora Built-in	NO	YES	NO	NO	NO
Supported Protocols	PMH - ORE	PMH	PMH	ORE	ORE
PMH Item	OAI View Instance	FDO	FDO	n.a	n.a
PMH-Metadata Records	Generated from OAI View Instance	Datastream	Datastream	n.a	n.a
PMH Metadata Format	DC, OAIzer-XML	DC	any format existing datastream	n.a	n.a
ORE Aggregation	OAI view instance	n.a	n.a	definded by annotation	subgraph visited starting from a given FDO
ORE Aggregated Resources	FDOs in the OAI view instance	n.a	n.a	datastreams with a given name	FDOs in the visited subgraph
ORE Proxy	FDOs in the OAI view instance	n.a	n.a	not supported	not supported
Relationships between Aggregated Resources	YES	n.a	n.a	NO	NO
Compound object boundaries	OAI view (properties and navigational criteria)	FDO	FDO	FDOs annotated with the same tag	navigation depth

Fig. 10. Comparing Fedora-OAIzer to other OAI solutions for Fedora

6 Acknowledgements

This work is partly funded by the European Commission as part of the projects OpenAIRE (FP7-INFRASTRUCTURES-2009-1, Grant Agreement no. 246686) and OpenAIREplus (FP7-INFRA-2011-2, Grant Agreement no. 283595).

References

1. D. D. I. Alliance. Data Documentation Initiative. `http://www.ddialliance.org/`.
2. L. Candela, D. Castelli, P. Pagano, C. Thanos, Y. E. Ioannidis, G. Koutrika, S. Ross, H.-J. Schek, and H. Schuldt. Setting the foundations of digital libraries: The delos manifesto. *D-Lib Magazine*, 13(3/4), 2007.
3. C. Lagoze, S. Payette, E. Shin, and C. Wilper. Fedora: An Architecture for Complex Objects and their Relationships. *Journal of Digital Libraries, Special Issue on Complex Objects*, 2005.
4. C. Lagoze and H. Van de Sompel. The OAI Protocol for Metadata Harvesting. `http://www.openarchives.org/OAI/openarchivesprotocol.html`.
5. C. Lagoze and H. Van de Sompel. The OAI Protocol for Object Reuse and Exchange. `http://www.openarchives.org/ore/`.
6. C. Lagoze and H. Van de Sompel. The open archives initiative: building a low-barrier interoperability framework. In *Proceedings of the first ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 54–62. ACM Press, 2001.
7. F. Manola and E. Miller. RDF primer. Technical report, W3C Recommendation, 2004. `http://www.w3.org/TR/rdf-primer/`.
8. MPEG-21, Information Technology, Multimedia Framework. Part 2: Digital Item Declaration. Technical report, ISO/IEC 21000-2:2003, 2003.
9. The Library of Congress. Metadata Encoding and Transmission Standard. `http://www.loc.gov/standards/mets/`, February 2002.