Digital Object Abstraction Layer: A Middleware for Building Federated Digital Libraries

George Pyrounakis
Dept. of Informatics and Telecommunications
University of Athens
Athens, 157 84, Greece
forky@di.uoa.gr

Mara Nikolaidou
Dept. of Informatics and Telematics
Harokopio University of Athens
Athens, 176 71, Greece
mara@hua.gr

Michael Hatzopoulos
Dept. of Informatics and Telecommunications
University of Athens
Athens, 157 84, Greece
mike@di.uoa.gr

Abstract

Federated digital libraries focus on the development of common services over multiple, heterogeneous digital repositories. To offer advanced services for federated digital libraries, there is a need to provide an abstract representation of digital objects stored within a specific repository, maintaining their semantics. In this paper a middleware for the development of federated digital libraries is introduced, named Digital Object Abstraction Layer (DOAL). It aims at (a) providing a unified abstract representation of digital objects stored in heterogeneous Digital Repositories independently of the software and location and (b) facilitating the development of complex reusable digital library service components. Its integration within federated digital library architecture promotes the creation of large scale distributed digital libraries, providing the same functionality as in the case where all digital content was stored in a single digital library system.

Keywords

Federated Digital Library, Digital Repository, Distributed System, Distributed Digital Object

1 Introduction

As extensively described in the literature, digital repositories (DRs) are usually employed for hosting the content produced by academic and research institutes (e.g. papers, thesis, dissertations, lessons etc). Meanwhile, the concept of digital libraries (DLs) has been introduced to identify the set of electronic services provided mainly by memory institutions (as libraries, museums or archives) on their digital content (either digitized or born digital). Though digital libraries and repositories serve the same scope, there is a slide confusion regarding their functionality. According to the notions introduced in [11], DL services are developed over digital repositories in order to build digital library systems. This approach was adopted in Pergamos Digital Library [8], where the Fedora repository software [19] is used for the lower layers of the system architecture, responsible for the storage and retrieval of digital objects. It is also adopted in the following. Building DL services over heterogeneous digital repositories, located in different servers and on various software platforms, results in the formation of federated digital libraries.

The basic entity of repositories is the digital object, as introduced by Kahn and Wilensky in [15]. A digital object can be conceived as a human generated artifact that encapsulates underlying digital content (text documents, images, videos, etc) and related information (metadata, internal structure, relations with other objects). The last few years a great amount of digital objects is aggregated on repositories created by scientific organizations, universities and research institutes worldwide. Although many repositories are members of federations, the functionality offered by federated services (usually searching based on a common subset of metadata) is not as rich as those provided by local repositories hosting digital collections. In order to offer the full functionality of a digital collection, a federated service must be able to retrieve and understand the semantics of the serialized digital objects, hosted on each repository. This means that DL services are dependent on the repository software they are build for, and cannot be reused in other repositories or digital libraries. To offer advanced services for federated digital libraries, a solution must be explored for handling the rich semantics stored on each digital object located in any local repository.

In current distributed platforms, as Web Services [9] and Common Objects Resource Broker Architecture (CORBA) [1] a middleware is used in order to abstract objects to a level that the physical and log-

ical representations are separated. Adopting a similar approach, we propose the creation of a middleware that interprets the structure and semantics of digital objects stored in different repository software located in multiple servers. The proposed middleware should work over current repository software applications, already used by various organizations, while its main target is to facilitate DL services that handle digital objects without possessing information about their physical characteristics.

Digital Object Abstraction Layer (DOAL) is the proposed middleware that allows software developers to build DL services over digital objects, independently of the repositories stored. DOAL uses an abstract form of digital objects that retain their semantics (metadata sets, digital content, relations, behavior) and hides their repository-dependent characteristics including (a) the location of the stored digital object, (b) the repository software used for its storage and (c) the encoding used for its serialization. Such middleware may widen the horizons on the development of advanced services based on the content aggregated on federated digital libraries (e.g. processing digital objects holding structured information emerging from experiments and stored on distributed repositories). Besides typical services offered by the majority of federated digital libraries, many new usages may arise starting from browsing collections or building complex workflow applications.

In the next section, current state of federated digital libraries is described and the motivation for introducing DOAL is explained. In section 3, basic DOAL concepts are introduced, as the resulting Federated Digital Library System architecture and the concept of Distributed Digital Objects (DDOs). In section 4, implementation issues as the Digital Repository Connectors and the Abstraction Mechanism, based on Digital Object Prototype, are explored. An example of the abstraction process of DDOs is also included to demonstrate the feasibility of the proposed concepts. Conclusions and future work reside in section 5.

2 Federated Digital Libraries

Most approaches for building federated digital libraries are based on a centralized architecture, using a server that harvests digital objects information from the repositories participating in the federation. Harvested metadata from digital objects are imported on a central repository so they are available to the appropriate DL services developed on top of it. Open Archive Initiative Protocol for Metadata Harvesting (OAI-PMH) [16] is mainly used for the creation of the first generation of federated digital libraries where the provided services are searching and browsing, based on a core metadata set. Updating of digital objects cannot be controlled by OAI-PMH, so local repositories are entirely responsible for the creation and editing of the digital objects that are provided through the federated digital library. In such cases, if a user is navigating in the web interface of the federated digital library and needs more complicated operations than searching and browsing (e.g. navigation in a collection hierarchy or digital content presentation) he is usually redirected to the web interface of the local repository that contains the specific collection. Some of the federated digital library projects that use OAI-PMH are the China Digital Museum Project [20], the National Science Digital Library [12], the European Library [6] and CORDRA [14].

The centralized architecture based on OAI-PMH is defined on Fig. 1. There are also other approaches

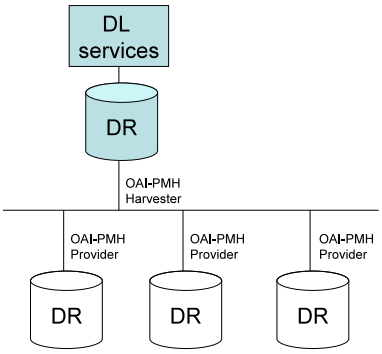

Figure 1: A centralized Federated Digital Library architecture based on OAI-PMH

for the federation of digital repositories, not dependent on OAI-PMH, as MARIAN [13], DRIVER (Digital Repository Infrastructure Vision for European Research) [4] and BRICKS (Building Resources for Integrated Cultural Knowledge Services) [3]. Although, most of them facilitate harvesting of both metadata and digital content, they are also based on a centralized architecture for the provision of federated services. Still in order to homogenize all content harvested by different repositories, the full semantics of digital objects remain on the local repository, encoded and serialized as specified by the repository software. Usually the local repositories are those that handle the serialized digital objects and provide the more advanced DL services.

A basic disadvantage of the current technology DL services is that the digital objects and their serializations are treated as a single entity. Two standards that are used for digital object serializations are the Metadata Encoding and Transmission Standard (METS) [7] and MPEG-21 Digital Item Dec-

laration Language (DIDL) [10]. These standards are used for the persistent storage of a digital object in a repository or for importing/exporting digital objects to/from a repository, that stores them in an internal non-standard format. These formats cover the need for storing and providing digital objects using the notion of Archival Information Package (AIP) and Dissemination Information Package (DIP) as described by the Open Archival Information System (OAIS) reference model [2]. In order for a DL service to access the serialized digital objects must have repository specific information like the physical representation of the digital objects, the location of repository server or the repository Application Programming Interface (API).

Some of the federated services not supported by federated digital libraries, due to the lack of a common interface to the semantic rich digital objects, are:

- workflows for submitting, editing and publishing digital objects,
- management of user access policies applied on digital objects and their components,
- management of digital collections,
- digital content handling (conversion, watermarks, compression etc.)
- preservation policies for digital content.

In order to provide such federated services, there is a need to provide an abstract digital object representation, that should be different from its serialized form.

In current architectural structures like Web Services and CORBA a middleware is used in order to abstract objects to a level that the physical and logical representations are separated. CORBA provides an infrastructure that allows for the cooperation of software components written in different programming languages and run in different servers, solving the problem of interoperability between distributed applications developed using different languages. Adopting a similar approach, we propose the creation of a middleware that interprets the structure and semantics of digital objects stored in different repository software located in multiple servers. Instead of the previous mentioned architecture, where digital objects are imported to a central server in a simplified form and the DL services are built on top of that, in our architecture for federated digital libraries this middleware preserves the role of the "interpreter" between the DL services and the digital objects that remain located in the local repositories. This middleware provides an API to the DL services which use an abstract but semantic-rich form of the digital objects, independent from the repository that are stored, as shown on Fig. 2.

The proposed middleware should work over current repository software applications, already used by various organizations, libraries, academic and research institutes. The main target of such middleware is to facilitate DL services to handle digital objects without possessing information about their physical characteristics. This information should be provided to the middleware in order to create this abstract digital object from its serialized form stored in the repository.This feature must be bidirectional allowing to save abstract digital objects that have been updated by the DL services, back to the repository. More specifically, the middleware should offer (a) the manipulation of a digital object as a logical entity without knowing its repository dependent characteristics (location and format) and (b) the ability for grouping and managing the behavior of digital objects.

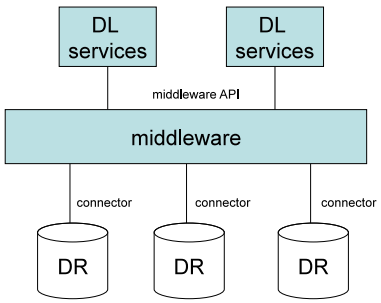

Figure 2: Using a middleware for the communication between DL services and digital repositories

In order for a repository software to cooperate with the proposed middleware a specific connector must be created supporting the basic repository operations. The concept of repository connectors is similar to CORBA architecture that uses stubs and skeletons in order to interpret an object from a server to a format compatible to the client. Similar functionality is offered on Database Management Systems (DBMS) by Open Database Connectivity (ODBC) standard API, where an application can communicate with a database independent of the DBMS used. Database drivers are used as connectors between ODBC Driver Manager and databases. In ODBC and databases, as long as CORBA and software applications, except the protocols, a software designer should be aware of the semantics for each entity used. In CORBA paradigm the structure of an object is needed, while in DBMSs the database schema is necessary in order to access a database.

In digital library systems the semantics of a digital object can be determined by its object type, as defined by the Digital Object Prototype (DOP) concept in [18]. The semantics of a digital object are specified by its components -categorized in metadata sets, re-

lations, digital content and behaviors- which are common for digital objects of the same type. The proposed middleware can use the DOP mechanism for the abstraction of a digital object to a semantic-rich form, according to its type. Because this middleware creates an abstract form of digital objects, can be named as *Digital Object Abstraction Layer (DOAL)*. DOAL offers the proper API for handling a digital object from a DL service, while its semantics are determined by its associated type. The DL services don't need to be bundled to specific repository software, but communicate with the DOAL that interprets methods from the repository software using specified connectors. Specifically, the key benefits of DOAL are the following:

1. Abstracts the serialized digital object in a form manageable by DL services independently of the repository software where is stored.
2. Associates digital objects to their DOP, so they are handled in a common manner by the DL services according to the their type.
3. Homogenizes the digital collections of different repositories that are part of a federated digital library.

3 DOAL Concepts

3.1 Digital Library System Architecture

As presented in Fig. 3 the architecture for a federated digital library system using DOAL comprises of three layers. The lower layer contains the digital repositories storing serialized digital objects. The middle layer consists of DOAL, that supports the instantiation of serialized digital objects to their abstract form according to their type. In order for a repository software to be accessible by DOAL a DR connector must be implemented. The upper layer constitutes of the DL services that provide the functionality needed to the end users, the collection designers, the cataloguers and the other user roles involved in the federated digital library. In Fig. 3, the DL services that are defined for demonstrative purposes, use the DOAL API in order to handle the abstract digital objects. The communication between DL services and abstract digital objects is bidirectional, meaning that, except reading digital objects and their components, the services may create, update or delete them.

Since DOAL is a layer between DL services and repositories it acts as a distributed software -using its classical meaning- in a way that hides to the upper layers the location and physical form of the digital objects. It supports the usage of digital objects stored in any repository, from DL services located in any node,

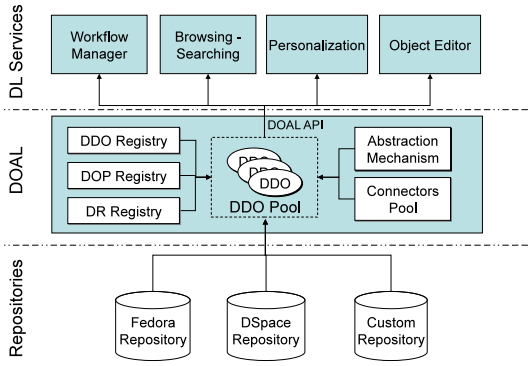

Figure 3: Using DOAL middleware in order to build a federated digital library

in a way transparent to the user and the software developer. A DOAL module exists in each node taking part to the federated digital library either as content provider through the repository or as a DL services provider.

DOAL creates the preconditions for the realization of a total distributed environment for federated digital library systems. This is a new concept to the development of DL services, since there is no limit to where and how the digital objects and their components are stored. For this reason an advanced version of digital objects is introduced, in order to utilize the full functionality given by the distributed environment, called *Distributed Digital Object (DDO)*. DDO is described in the next paragraph.

DOAL middleware contains the following modules:

- The *DDO Registry*, *DOP Registry* and *DR Registry* that hold the necessary information for the DDOs, DOPs and digital repositories respectively.
- The *DDO Pool* that holds the DDO instances which are available to be used by the DL services through DOAL API.
- The *Connectors Pool* that contains the available DR connectors for the communication with the registered repository software.
- The *Abstraction Mechanism* that is responsible for the abstraction of digital objects according to their type, as also implemented in Pergamos Digital Library. Abstraction Mechanism implementation is described in paragraph 4.2.

3.2 Handling Distributed Digital Objects

Distributed Digital Object concept is used for the abstract representation of digital objects on the distributed DL environment offered by DOAL. DDO is described by a permanent identifier and belongs to a

specific type (defined by its DOP). DDO constitutes of one or more Serialized Digital Objects (SDOs), located in different repositories. Each SDO contains a different part (one or more digital object components) of the DDO. An example of a DDO of type *photo* is presented in Fig. 4, where *DDO1* constitutes of *SDOx* and *SDOy*, stored on two different repositories (*Node A* and *Node B*). *SDOx* contains descriptive metadata of the photo using the metadata standard suggested by Dublin Core Metadata Initiative (DCMI) [5] and a thumbnail of the contained image (*DC* and *THUMB* in figure). *SDOy* contains two different forms of the digital image: one of low quality for the web and one of high quality for preservation purposes (respectively *WEB* and *HQ*). SDO does not contain information about its type (because it is a part of the whole object), neither a persistent identifier with global scope outside the repository. Nevertheless it must contain the persistent identifier of the DDO it belongs.

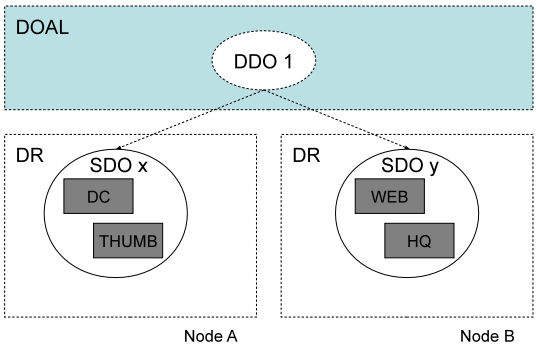

Figure 4: An example of a simple DDO of type *photo* constituted of two SDOs on different repositories

Some of the advantages provided by distributed digital objects are the following:

1. Flexibility in the location and category of storage (repository, filesystem, database, etc).

2. Distribution of serialized digital objects according to the available storage capacities.

3. Usage of repositories that fits best for each type of component (e.g. one repository storing metadata, a second digital images and a third digital videos).

4. Ability to store SDOs according to the network infrastructure.

5. Load balancing of the systems storing and serving digital objects.

4 Implementation Issues

4.1 Digital Repository Connectors

DR connectors are application modules that are registered in DOAL middleware, responsible for interpreting basic repository methods between DOAL and the respective repository software. The methods that need to be implemented in a DR connector are the following:

- Get/Save/Remove a digital object from the repository.
- List the components of a digital object.
- Get/Save/Remove a specific digital object component.
- Search for digital objects using specified metadata (e.g. identifier, date, creator, etc). In order for this method to be available, proper mappings between the metadata fields defined in DOAL and the fields stored on specified metadata set in the digital object must be determined.

DR connectors are available to the Abstraction Mechanism through the Connectors Pool. The Abstraction Mechanism is aware of the DR connector to use for each digital object by the DR Registry. This registry holds for each repository: (a) a unique identifier, (b) a unique name, (c) a persistent URL and (d) a link to the DR connector. In Pergamos DL [8] a connector for Fedora version 2.0 is implemented and bundled in the digital library system. To be used in DOAL this connector is being unbundled from Pergamos DL and reengineered as a separate application module. Connectors in DOAL can also be implemented for data sources other than repositories, like databases or filesystems.

4.2 Abstraction Mechanism

Abstraction in DOAL is based on the notion of Digital Object Prototype (DOP), facilitating the description of a type of digital objects (e.g. book, paper, photo, etc) by specifying the common characteristics that define it. These characteristics are specified for each of the four components constituting a digital object that are: (a) the metadata sets, (b) the digital content, (c) the structure and relations with other objects and (d) its behavior. The specifications of each DOP are given using a template written in XML. By using DOPs it is feasible for DL services to handle digital objects of the same type in a unified manner. For example, if the loaded DDO is of type *book*, a DL service is aware of the metadata fields available, the image files for the cover page, the relations with possible DDOs of *page* type and the interfaces that determine

its behavior (e.g. show page, table of contents, book description, etc). This process, as analyzed in [17], is equivalent to the "instance-of" relation between objects and prototypes in Object Oriented Programming. The abstraction process covers the DDO loading from the associated serialized digital objects, while at the same time conformance to its type is ensured. When a DDO is loaded, it is available in the DDO Pool in order to be used by a DL service.

In the distributed environment of DOAL, the abstraction process of a DDO must be realized in the node where the calling DL service is located. In that way multiple instances of a DDO may exist concurrently in different nodes. When DOAL is loading a DDO in a node two cases exist: (a) if the object component is stored in a repository located in the same node as the DOAL module it is loaded immediately from it, (b) in a different case, a request for the required component is forwarded to the node that contains it. The communication between the two nodes is realized on the DOAL layer and not in the repositories layer. Repositories don't possess the proper information to communicate with its other and usually they don't support an appropriate protocol for that purpose.

Abstraction Mechanism is implemented and used in the productive environment of Pergamos Digital Library, hosted by the University of Athens. Existing implementation is currently extended in order to support the functionality of distributed digital objects and the communication between DOAL modules hosted in different nodes.

4.3 An Example of Handling Distributed Digital Objects

The process of DDO abstraction in DOAL is analytically explained using as an example *DDO1*, presented in Fig. 4, which is of type *photo* and comprises of two serialized digital objects. As shown in Fig. 5, the *Browse service* in *Node A* of a federated digital library requests the DC metadata set and the web image of *DDO1*. The SDOs stored in *Node A* contain the Dublin Core metadata describing the photo and the thumbnail image (*DC* and *THUMB* components respectively), while SDOs stored in *Node B* contain the low quality image for the web and the high quality image (*WEB* and *HQ* components).

Using DDO identifier the proper record is retrieved by the DDO Registry and the associated DOP is loaded through the DOP Registry. A record on DDO Registry contains: (a) the persistent identifier of the DDO, (b) the identifier of the associated DOP and (c) the locations of the DDO components stored in specified SDOs. DDO1 instance

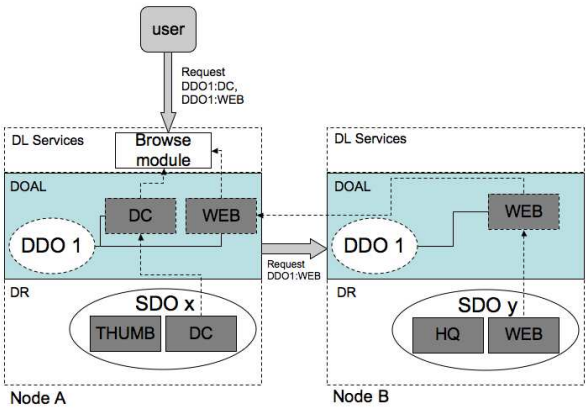

Figure 5: An example of a DDO abstraction in DOAL

is created on *Node A* that holds the whole information provided by the two registries, which contains the DDO identifier, the DOP definition and the full paths of its components (for *DDO1* the full paths are: *nodeA:SDOx:THUMB*, *nodeA:SDOx:DC*, *nodeB:SDOy:WEB* and *nodeB:SDOy:HQ*). When abstracting a DDO, the components are not loaded (each component is loaded on demand), but the DDO instance holds all the data needed to load them. In the case of *DDO1:DC*, where the repository is also on *Node A*, it is loaded using the Connectors Pool of that DOAL module. For the *DDO1:WEB* component the request is redirected to the *DDO1* instance of *Node B* (if it does not exist in the DDO Pool it is loaded). Finally, the requested component is forwarded back to the *DDO1* instance on *Node A*. In both cases, this is realized in a transparent fashion for both the DL services developer and the end-user.

DDOs and corresponding loaded components, after loaded as instances on DOAL, are retained for a specified time interval, since it is most likely they will be requested again. Though, in a distributed environment, corresponding SDOs may be updated. Thus, digital object consistency should be maintained and a proper synchronization policy should be applied in case of updating the SDO containing the loaded components. When loading a component in a node, DDO holds a timestamp specific for that node (e.g. *NodeA:DDO1:WEB.TIMESTAMP*). Each time this component is requested from that node, the DDO instance in that node communicates with the relative DDO instance on the node that contains the respective SDO, and the timestamps are compared. In case the component has been updated at that time interval, it is forwarded to the proper node and the timestamp is renewed. If there is no change the loaded component is provided as is.

5 Conclusion - Future Work

DOAL middleware enables DL service developer and collection designer to create reusable services without worrying about where and how the digital objects are stored. The first develops generic DL services using the methods offered by the DOAL API, while the second designs collections for a federated digital library just by defining the different types of digital objects. End-users have also a great benefit by the advanced services provided on the distributed environment of federated digital libraries.

We are currently working on the implementation of a full operational DOAL module. This includes the communication channels between the DOAL nodes, the caching for the DDO instances and components, and the DOAL API specification. Implementation will contain the synchronization of registries on different nodes, methods for avoiding save conflicts on serialized digital objects and DR connectors for basic open source repository software, like Fedora, DSpace and EPrints.

References

[1] Common object request broker architecture (corba / iiop), version 3.0.2. Available from: `http://www.omg.org/technology/documents/formal/corba\_iiop.htm`.

[2] Reference model for an open archival information system (oais). Technical Report Blue Book, Issue 1, Consultative Committee for Space Data Systems, 2002.

[3] Bricks web site, 2009. Available from: `http://www.brickscommunity.org/`.

[4] Driver web site, 2009. Available from: `http://www.driver-repository.eu/`.

[5] Dublin core metadata initiative (dcmi) metadata terms, 2009. Available from: `http://www.dublincore.org/documents/dcmi-terms`.

[6] The european library, 2009. Available from: `http://www.theeuropeanlibrary.org`.

[7] Metadata encoding & transmission standard (mets), 2009. Available from: `http://www.loc.gov/standards/mets/`.

[8] Pergamos digital library, 2009. Available from: `http://pergamos.lib.uoa.gr/`.

[9] Web services, 2009. Available from: `http://www.w3.org/2002/ws/`.

[10] J. Bekaert, L. Balakireva, P. Hochstenbach, and H. Van de Sompel. Using mpeg-21 dip and niso openurl for the dynamic dissemination of complex digital objects in the los alamos national laboratory digital library. *D-Lib Magazine*, 10(2), February 2004.

[11] L. Candela, D. Castelli, N. Fuhr, and Y. Yoannidis. Current digital library systems: User requirements vs provided functionality. Technical Report DELOS Deliverable D1.4.1, 2006.

[12] C. Lagoze et al. Core services in the architecture of the national science digital library (nsdl). In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, 2002.

[13] M. A. Goncalves, R. K. France, and E. A. Fox. Marian: Flexible interoperability for federated digital libraries. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*, 4 - 9 September 2001.

[14] H. Jerez, G. Manepalli, C. Blanchi, and L. Lannom. Adl-r: The first instance of a cordra registry. *D-Lib Magazine*, 12(2), February 2006.

[15] R. Kahn and R. Wilensky. A framework for distributed digital object services. Technical report, Corporation of National Research Initiative - Reston USA, 1995.

[16] C. Lagoze and H. Van de Sompel. The open archives initiative: Building a low-barrier interoperability framework. In *First ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'01)*, 2001.

[17] K. Saidis, G. Pyrounakis, and M. Nikolaidou. On the effective manipulation of digital objects: A prototype-based instantiation approach. In *9th European Conference on Digital Libraries*, 2005.

[18] K. Saidis, G. Pyrounakis, M. Nikolaidou, and A. Delis. Digital object prototypes: An effective realization of digital object types. In *10th European Conference on Research and Advanced Technology for Digital Libraries*, 2006.

[19] T. Staples, R. Wayland, and S. Payette. The fedora project: An open-source digital object repository management system. *D-Lib Magazine*, 9(4), April 2003.

[20] R. Tansley. Building a distributed, standards-based repository federation: The china digital museum project. *D-Lib Magazine*, 12(7/8), July/August 2006.