elpub2002 Proceedings — J. A. Carvalho et. al. (Eds.) — VWF Berlin, 2002

A Flexible Repository Service

The OpenDLib Solution

Donatella Castelli and Pasquale Pagano

ISTI-CNR
Area della Ricerca CNR di Pisa
56126 Pisa, Italy
{castelli, pagano}@iei.pi.cnr.it

Abstract. The Repository Service that we propose is a software module, developed as part of a digital library service system called OpenDLib, that stores and disseminate digital documents. The service is accessible through an http-based protocol, and is generic with respect to the type and contents of information space to be handled. It is thus application-independent and can be implemented as a basic component in many kinds of digital libraries. This paper describes the features that render it broadly applicable.

1 Introduction

Many application sectors (libraries, humanities, museums, local administrations, etc.) are currently heavily engaged in projects for digitalization. The Communities that are involved in this process are increasingly demanding suitable technologies to handle, search and manipulate digital documents. The technology, however, is not yet sufficiently developed to respond fully to these requests. Existing digital library (DL) systems have usually been built as quick solutions to meet the needs of specific user communities [1–4]. Thus, their underlying programming logic covers only very specific requirements and their architecture has not been designed to support future extensions or redeployment [5]. Such systems cannot be considered a solution to the demand from the application communities for easy-to-use systems to handle and disseminate new digital content; similarly the solution does not lie in building other ad-hoc systems. The construction of a DL is very complex; very few communities can afford the economic cost of such a development. It is clear that other technical solutions need to be found. Our research efforts over the last

three years have focussed on this problem. During this period we have worked on designing and developing a general system of digital library services, flexible enough to respond to the requirements of a wide variety of user communities. This system, called OpenDLib [6], can be used to create an application-specific digital library by instantiating the system appropriately and then either loading or harvesting the content to be managed. OpenDLib consists of a federation of services that implement the digital library functionality making few assumptions about the nature of the documents handled. A key service of this system is the Repository Service. This is a flexible service able to store and disseminate different types of documents, metadata formats, and organizations of the information space.

Two main features renders the Repository Service adaptable to different application contexts: the model of the information space, and a configurable behaviour that varies according to the values given to a set of customization parameters.

The model of the information space organizes the digital library content into multiple hierarchies of collections. The documents in the collections must conform to a document model, called the Document Model for Digital Library (DoMDL), able to represent structured, multilingual and multimedia documents. The DoMDL can be customized according to the DL content to be handled. For example, it can be configured to handle documents structured as books or journals, or to manage collections of videos structured as sequences, shots and frames. Multiple metadata records in different formats can be associated with the same document and with its parts. These formats are not embedded in the system but must be indicated at the DL configuration time and can be modified dynamically throughout the DL lifetime.

The behaviour of the Repository Service is configurable along several dimensions. For example, it can be configured to serve certain publishers, to handle particular metadata formats, to automatically generate other metadata formats, or to archive certain sets of documents.

This paper presents these two features of the Repository Service that render it adaptable to different digital library contexts. The DoMDL document model is described in Section 2, the Repository Service configuration features are presented in Section 3 and concluding remarks are given in Section 4.

2 The Repository Information Space

The information space of a digital library is usually structured into sets of documents, often called "collections", which are homogeneous according to some criteria. These can be criteria on the publisher (e.g. the set of documents published by the Italian Central Institute for Union Catalogue, the set of documents published by the PORBASE project consortium, etc.), or criteria on the subject (e.g. all the documents on "Digital Libraries", all the documents on "Metadata", etc.) or other

more complex criteria (e.g. documents on "Digital Libraries" in English published after 1998). The collections may be structured into hierarchies. For example, an information space of documents on Computer Science could be organized into a subject hierarchy that reflects the ACM classification scheme [7]. The hierarchy models a document containment relation. According this relation, for example, all the documents of the collection "Digital Libraries" also belongs to "Information Storage and Retrieval" and to "Information Systems".

The structure of the documents that populate the information space of different digital libraries is usually different. For example, a digital library of conference proceedings may contain documents (the proceedings) that are aggregates of other documents (the preface and the articles). Each article, may be disseminated in different ways, for example it can be disseminated both as a text in postscript format (the readable content of the article) and as a videos in MPEG3 format (the speaker presentation). A digital library of project deliverables is likely to have documents with a completely different structure. For example, these may be textual reports, structured into sections, and demos of the project prototypes. Different digital libraries may also support different metadata formats. For example, a MARC format can be used in a digital library of journals built by library professionals but will not be certainly used in a digital library of geographical information or in a digital library of self-published documents. A digital library service system that wants to be generic with respect to the underlying information space must be able to support different organizations of the information space, and the storage and dissemination of a wide variety of document types and descriptive metadata formats.

The OpenDLib Repository Service was designed to satisfy this requirement. In particular, it was designed to support a very powerful document model, called DoMDL, that can be customized at the start-up of the system to represent the specific structure of the application documents and their descriptive metadata formats. This model is presented below.

In order to render the presentation more intuitive, we will introduce this model by referring to the represented concepts, rather than speaking of its data structures.

2.1 The Document Model

DoMDL distinguishes four aspects of document modelling, each of them is seen as a different kind of entity: *Document*, *Version*, *View*, and *Manifestation* (see Figure 1).

Document. An Document entity represents the more general aspect of a document, i.e. the document as a distinct intellectual creation. For example, the article "The SOMLib Digital Library System" by Andreas Rauber and Dieter Mekl, the book "Digital Libraries and Electronic Publishing" by William Arms, the lecture

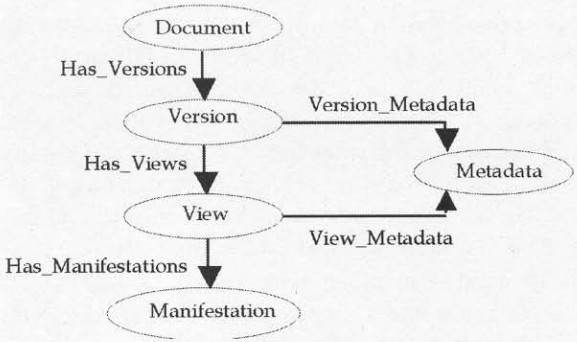

Fig. 1. *The Document Model*

"A dynamic Warehouse for XML Data of the Web" by Serge Abiteboul, the proceeding of the conference ECDL'99, can all be modeled as entities *Document*. Each entity of this type is named by a URN (Unified Resource Name). Unlike a URL, a URN is location independent.

Version. A Version represents a specific edition of the distinct intellectual creation, i.e. an instance along the time dimension. The preliminary version of the paper "The SOMLib Digital Library System", the version submitted to the conference ECDL'99, the version published in the ECDL'99 proceeding, are examples of successive editions of the same document. Versions are linear and numbered. The first version is version 1, subsequent versions are 2, 3, etc.

View. A View models a specific intellectual expression of an edition of a document. A view excludes physical aspects that are not integral to the intellectual perception of a document. A document edition is perceived through one or more views. For example, the original version of the ECDL'99 proceedings might be disseminated under four different views: a) a "structured textual view" containing a "Preface" created by the conference Chair, and the list of papers presented to the conference, b) a "textual view structured into thematic sessions", where each session contains the documents presented during that session, c) a "presentation view", containing the list of the conference presentation slides, and d) a "metadata view", containing a structured description of the proceeding.

Manifestation. A Manifestation models the physical formats under which a document is disseminated. Examples of manifestations are: the MPEG file which maintains the video recording of the presentation made by Andreas Rauber at the ECDL '99 conference, the AVI file of the same video, the poscript file of the paper presented by Andreas Rauber at the ECDL '99, etc.

The entity View is specialized in two sub-entities: *Metadata* and *Content.* The former view perceives a version through the conceptualizations given by its metadata representations. These can be a flat list of pairs (fields, values), as with the Dublin Core metadata records [8], or more complex conceptual structures, such as IFLA-FRBR descriptions [9]. Typically this is the view indexed to support fielded querying and browsing, but it can have other uses. For example, it may be the view disseminated free of charge in an environment in which documents are covered by access rights, or the view disseminated on a mobile device.

The Content is the view of the document content. It has two sub-entities: *Body, and Reference.* Each of them has a different representational semantics characterized by different constraints and relationships (see Figure 2).

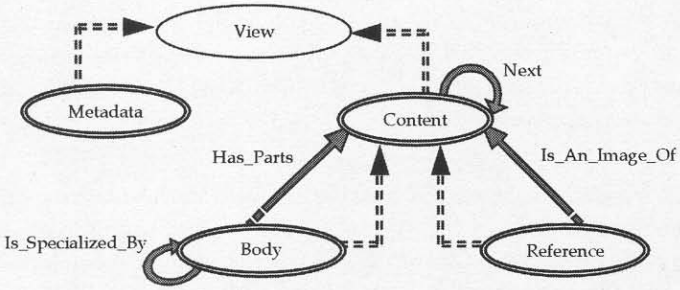

Fig. 2. *The View Entity, its Specialised Entities and its Relationships*

The *Body* entity represents a view of the document content that can be either perceived as a whole or as the aggregation of other views. For example, the ECDL'02 proceedings can have a textual view that is built as the aggregation of the textual views of its components article. Similarly, the "black and white (B&W) view" of the "Herald Tribune" newspaper can be modeled as the aggregation of a certain number of "B&W page views". Each of these pages, in turn, can be represented as the aggregation of particular "B&W" views of articles and pictures, each authored by a different person, and possibly, published elsewhere. The relation *Has_Parts* maintains the link between a Body view and its components views. If the parts are ordered, i.e. they document is a sequence of parts, then two consecutive parts are linked by the *Next* relation.

A body view can be specialized, i.e. there may be other views that represent more detailed perceptions of the same content. For example, the view "slides of the Andreas Rauber presentation" might be specialized by: "B&W view of the slides", and by "Colored view of the slides". A view is related to all its specializations though the relation *Is_Specialised_By.*

A *Reference* entity represents a view that, being equal to an already registered one, does not need to be explicitly stored. It has been introduced in the model to avoid duplication of information and it has no conceptual counterpart. This view can be used, for example, when a new version of a composite document is submitted. All the views that have not been changed with respect to the previous version can be specified as reference and linked to the original through the relation *Is_An_Image_Of*. There is thus no need to load and store them again. As a consequence of its definition, this type of view has no manifestation and no assigned explicit property, except the link to the existing view. This link is sufficient to derive all the information that is required to retrieve and disseminate the complete description of the view modeled as reference.

Note that Body and Reference do not partition the Content view. For example, the views Summary and Keyframe are content views that do not belong to any of these subclasses.

The relations *Has_Versions*, *Has_Views*, and *Has_Manifestations* link the different aspects of a document. Note that these relations are multiple, i.e. there can be several objects in the range associated with the same object in the domain. This means that there can be multiple versions of the same documents, multiple views of the same version and multiple manifestations of the same view.

Versions and views may be described by one or more metadata records in different formats. This records are accessible through the relations *Version_Metadata* and *View_Metadata*.

Each of the entities described above has a set of attributes which model properties of the corresponding aspect of the document. For space reasons we cannot list them here. The complete list can be found in [10]. We only want to mention that each entity has an attribute that specifies the rights on the modeled document aspect. This makes it possible, for example, to model possibly different rights on different editions, or different rights on different parts of the same view, and so on.

By combining the rich document model with the possibility of having multiple metadata descriptions for each aspect, the DoMDL document model provides a very powerful representational mechanism that can satisfy the needs of many different information spaces. This is one of the key aspects that enables the Repository to serve different application contexts. In the next section we will describe the other specific feature of the Repository Service that make it widely applicable.

3 The Repository Service

The Repository Service stores and disseminates DoMDL documents and metadata descriptions in different formats. It is implemented as a software module whose

functionality is accessible through a number of service requests that must be formulated according to an established HTTP-based protocol. This protocol is a superset of the OAI-MPH protocol [11]. This means that the Repository Service is able to provide its own advanced functionality and also act, if required by interoperability matters, as a simple OAI-compliant archive.

In the OpenDLib architecture, the Repository is a distributed service, i.e. each publishing institution can host an instance of the Repository Service on its local server, even if the other services, like the Query Mediator or the Registry, are located elsewhere. The choice of distributing the Repository Service is a consequence of the request, formulated by many institutions, to maintain control over their documents. This does not mean that there must be one Repository Service for each organization, actually an institution may decide to host on its own server documents published by other institutions.

In the rest of this section we present the range of functionality that is covered by the Repository Service by listing some of its configuration parameters and discussing how they influence its behaviour. All these parameters must be specified at the start-up of the system (following an XML syntax) and can change along the DL lifetime.

- *Collections.* The documents stored in a Repository instance may belong to one or more collections. The collection is a cross-repository instance notion, i.e. documents belonging to the same collection can be distributed over several repository instances. This storage policy assumes an agreement among the publishing institutions on the semantics of the collections and of their hierarchical structure. The choice as to which collections are handled by a Repository instance is made at the DL configuration time and can change during the DL lifetime.
- *Association between repository instances and publishing institutions.* Each Repository instance stores documents published by one or more publishing institutions. The association between a repository instance and its publishing institutions is established when an OpenDLib digital library is set up and can be changed during the digital library lifetime to support the dynamic addition of new publishing institutions to the federation.
- *Metadata formats.* The Repository is capable of storing multiple metadata formats. The set of these formats is chosen when the Repository is started-up, but can also be updated later, for example when a new publishing institution is hosted. In order to simplify the configuration, a simple XML configuration file maintains (for each metadata format) the name and description, plus references to its DTD and to the list of used namespaces.
- *Derived metadata formats.* The Repository service, if appropriately configured, can automatically derive metadata records from other existing metadata format. For example, it can be configured to generate a Dublin Core record each

time a MARC record is submitted. This automatic generation is carried out by a generic procedure whose input is a tuple indicating the source metadata format name, the target name, and a reference to an XML file that maintains the corresponding mapping. This configuration file, called the mapping table, is very easily to define because it maintains the relation between source and target attributes plus a function, or a reference to it, to map source values into target ones. The possibility to modify them is very useful, for example, to automatically translate date values, personal names, etc. If the function is not specified, the values will simply be copied as they are. The Repository supports a number of standard metadata format mapping tables, but it can also accepts other procedures as illustrated in the section below on derived manifestation types.

- *Manifestation type.* Any view of a document can have several different manifestations. The range of possible manifestation types is a configuration parameter.
- *Derived manifestation type.* The Repository service, if appropriately configured, can automatically derive manifestation types from others. For example, it can be configured to generate a PDF manifestation each time a Postscript one is submitted. This automatic generation is carried out using appropriate procedures pre-loaded by the service. Other derivations can easily be added specifying the source manifestation name, the target one, plus a reference to an internal procedure, new or existing, or to an external program.
- *External manifestations.* The Repository Service admits external manifestations, i.e. manifestations that are not physically stored in the Repository but are handled by other specialised services. The Repository maintains these as dummy manifestations that report, as one of their attributes, the URL of the real manifestation. This facility is useful for those information spaces which contain documents that need a special treatment. For example, this permits an organisation in which the Repository stores the structure and the metadata description of video documents whereas a specialised video server that provides video specific functionality, like streaming, handles the manifestations of these documents.
- *Archived documents.* A facility can be activated at the start up of the Repository Service to store documents that are rarely requested in a compressed format. This make it possible to reduce the storage space required. These documents are automatically decompressed when the Repository serves a request for their dissemination.

The Repository Service can be customised by selecting values for each of the dimensions listed above. The values assigned are not independent, but they are constrained by consistency rules that establish the legal configurations of the Repository. For example, any derived metadata formats must be one of the metadata formats supported. The Repository Service checks these rules both at the start-up and each time the configuration is modified.

4 Conclusions

This paper has presented the features that render the OpenDLib Repository Service able to support the storage and the dissemination of documents in different digital library systems.

The functionality of the Repository Service is accessible through a set of service requests formulated according to a HTTP-based protocol. Its configuration parameters are defined precisely and so are the consistency rules among these parameters. These features render the Repository Service a generic service for handling and disseminating digital documents that can be used outside of the OpenDLib digital library service system. For example, it can be used to easily implement an archive that has to be OAI-PMH compliant. The only requirement for its use is that the host system must communicate its requests and its configuration parameters through the established protocol. The Repository Service is currently employed in the Scholnet project [12]. This is a digital library project which aims at building a digital library system able to support the communication and collaboration within scholarly communities. The experience made in Scholnet has validated the flexibility of the Repository Service and its adaptability to different application contexts.

References

1. NCSTRL: Networked Computer Science Technical Reference Library `http://www.ncstrl.org`
2. Networked digital library of theses and dissertations. `http://www.theses.org`
3. RePEc: Research Papers in Economics, `http://repec.org/`
4. arXiv.org e-Print archive. `http://arxiv.org/`
5. Hussein Suleman and Edward A. Fox, A Framework for Building Open Digital Libraries, in D-Lib Magazine, vol. 7, n. 12, Dec. 2001, `http://www.dlib.org/dlib/december01/suleman/12suleman.html`
6. Donatella Castelli and Pasquale Pagano, "OpenDLib: A Digital Library Service System", in Proc. ECDL'02 Conference, Maristella Agosti and Costantino Thanos eds., Roma, September, 2002.
7. ACM Computing Classification System. `http://www.acm.org/class/`
8. Dublin core metadata element set, version 1.1.: Reference description. `http://dublincore.org/doceumnts/dcse`
9. K.G. Saur. Functional requirements for bibliographic records. Final report. Technical Report, Munchen, 1998. `http://www.ifla.org/VII7s13/frbr/frbr.pdf`
10. OpenDLib: a Digital Library Service System. `www.opendlib.com`
11. The Open Archives Initiative, `http://openarchives.org`
12. Scholnet: A Digital Library Testbed to Support Networked Scholarly Communities. `http://www.ercim.org/scholnet`