Milos: a Multimedia Content Management System for Digital Library Applications*

Giuseppe Amato, Claudio Gennaro, Fausto Rabitti, Pasquale Savino

ISTI-CNR
Pisa, Italy
{giuseppe.amato, claudio.gennaro, fausto.rabitti,
pasquale.savino}@isti.cnr.it

Abstract. This paper describes the MILOS Multimedia Content Management System: a general purpose software component tailored to support design and effective implementation of digital library applications. MILOS supports the storage and content based retrieval of any multimedia documents whose descriptions are provided by using arbitrary metadata models represented in XML. MILOS is **flexible** in the management of documents containing different types of data and content descriptions; it is **efficient** and **scalable** in the storage and content based retrieval of these documents. The paper illustrates the solutions adopted to support the management of different metadata descriptions of multimedia documents in the same repository, and it illustrates the experiments performed by using the MILOS system to archive documents belonging to four different and heterogenous collections which contain news agencies, scientific papers, and audio/video documentaries.

1 Introduction

Digital Library (DL) technology is today limited to manage specific types of digital objects and specific metadata description models. This implies that existing DL applications can be hardly adapted to different application environments and to different metadata description models. Indeed, many DLs were built having in mind a specific application and, in many cases, a specific document collection, thus resulting in an ad-hoc solution: all components of the DL – the data repository, the metadata manager, the search and retrieval components, etc. – are specific to a given application and cannot be easily used in other environments. Many of these systems guarantee inter-operability with other systems, by adopting standard protocols such as OAI, or Z39.50. However, their inter-operability is limited to the exchange (import/export) of data/metadata. In fact, there is no

* This work was partially supported by the ECD project (Extended Content Delivery) [12], funded by the Italian government, by the VICE project (Virtual Communities for Education), also funded by the Italian government, and by DELOS NoE, funded by the European Commission under FP6 (Sixth Framework Programme). We would like to thank Paolo Bolettieri, Franca Debole, Fabrizio Falchi, Francesco Furfari, and Bertrand Le Saux for their valuable contribution to the MILOS implementation.

chance of reusing software components, to integrate functionality of other DLs, or to use digital contents (documents and metadata) compliant to other standards. This is mainly due to the lack of standard general purpose basic building components tailored to DL application design.

In this paper we propose an approach similar to that applied in the field of traditional database applications. In fact, database applications are generally built relying on a Database Management System (DBMS), a general purpose software module that offers all functions needed to build many different database applications (e.g., banking, corporate management, billing, etc.); these applications will use different types of data, and they will support many different types of retrieval. We intend to demonstrate that the same can be done in the DL field: it is possible to build a general purpose Multimedia Content Management System (MCMS) which offers functionalities specialized for DL applications. Different DL applications, can be built on top of such an MCMS, each supporting the management of documents of any data type, described by using different metadata description models, searchable in many different modes. This MCMS should be able to manage not only formatted data, like in databases, but also textual data, using Information Retrieval technology, semi-structured data, typically in XML, mixed-mode data, like structured presentations, and multimedia data, like images and audio/video.

In this paper we discuss the functionality that the MCMS should provide (Section 2) and we present the MILOS Multimedia Content Management System (Section 3), that we built according to those criteria. Finally we present several significant DL applications that were implemented by using MILOS, and we show the advantages of the proposed approach in building these specific DL applications, resulting in the simplicity of the implementation and in significant system performance (Section 4).

2 Motivations

Digital library applications are document intensive applications where possibly heterogeneous documents and their metadata have to be managed efficiently and effectively. We believe that the main functionalities required by DL applications can be embedded in a general purpose Multimedia Content Management System (MCMS), that is a software tool specialized to support applications where documents, embodied in different digital media, and their metadata are efficiently and effectively handled.

The minimal requirements of a Multimedia Content Management System are *Flexibility*, in structuring both multimedia documents and their metadata, *Scalability*, and *efficiency*. *Flexibility* is required to manage both basic multimedia documents and their metadata. The flexibility required in representing and accessing metadata can be obtained by adopting XML as standard for specifying any metadata (for example MPEG-7 [5] can be used for multimedia objects, or SCORM [11] for e-Learning objects). Requirements of *scalability* and *efficiency* are essential for the deployment of real systems able to satisfy the operational

requirements of a large community of users over a huge amount of multimedia information.

A MCMS mainly supports the *storage and preservation* of digital documents, and their *efficient and effective retrieval* and *management*. This is provided with an appropriate management of documents and related metadata, by:

1. managing different documents embodied in different media and stored with different strategies;
2. supporting the description of document content by way of arbitrary, and possibly heterogeneous, metadata;
3. providing DL applications with custom/personalised views on the metadata schema actually handled.

Point 1) requires that no assumption should be taken on the types of media and encoding used to represent documents, and especially on the specific strategy used to store them. This allows applications to be unaware of the technical details related to multimedia document management. For instance, textual documents can be stored in the file system and served to the users using a normal web server. However, video documents might need to be maintained in a video server that uses various storage devices, as for example digital tapes stored in silos, optical disks, and/or temporary storage space on arrays of hard disks [21]. In addition, video documents might be served exploiting specific real-time continuous media streaming strategies to avoid hiccups during playback. The DL application should be designed independently from these issues, which should be managed transparently by the MCMS. For instance, changes in the storage strategies should be possible without changing the DL application software.

Point 2) states that a content management system should be able to deal with arbitrary metadata. This is required by the fact that different DL applications, according to their specific requirements, might need to use different metadata. Consider that existing archiving organizations have already their own metadata schemas, and hardly want to modify them to be compatible with a specific system. Therefore, a DL management system should be able to support any metadata schema without requiring metadata translation or restrictions on the functionality offered. There are also cases where the same application needs to deal with different metadata at the same time. These different metadata might be needed because the documents have redundant descriptions in terms of different metadata, or because the DL application is dealing with a document collection described with heterogeneous metadata. The last case might occur, for instance, in case of integration/merging of archives managed by different organization.

Point 3) makes it possible that the metadata schema seen by the DL application is different from the metadata schemas actually stored in the repository of the content management system. Suppose that an application was built to deal just with a specific metadata schema. The MCMS should be able to serve requests of such an application even if metadata stored in the repository comply to different schemas. Metadata schema independence can be obtained by exploiting techniques of schema mapping. This feature is especially useful in

case of heterogeneous metadata available at the same time in the repository: the DL application will refer to just one metadata schema, relying on the multiple schema mapping performed on the fly by the MCMS. In addition, this feature allows different DL application, which require different metadata schemas, to share the same MCMS transparently.

3 The MILOS multimedia content management system

We have designed and built MILOS (Multimedia dIgital Library Object Server), a MCMS that satisfies the requirements and offers the functionalities discussed in previous section. The MILOS MCMS has been developed by using the Web Service technology, which in many cases (e.g. .NET, EJB, CORBA, etc.) already provides very complex support for "standard" operations such as authentication, authorization management, encryption, replication, distribution, load balancing, etc. Thus, we do not further elaborate on these topics, but we will mainly concentrate on the aspects discussed above.

MILOS is composed of three main components as depicted in Figure 1: the Metadata Storage and Retrieval (MSR) component, the Multi Media Server (MMS) component, and the Repository Metadata Integrator (RMI) component. All these components are implemented as Web Services and interact by using SOAP. The MSR manages the metadata of the DL. It relies on our technology for native XML databases, and offers the functionality illustrated at point 2) above. The MMS manages the multimedia documents used by the DL applications. MMS offers the functionality of point 1) above. The RMI implements the service logic of the repository providing developers of DL applications with a uniform and integrated way of accessing MMS and MRS. In addition, it supports the mapping of different metadata schemas as described at point 3) above. All these components were built choosing solutions able to guarantee the requirements of flexibility, scalability, and efficiency, as discussed in the next sections.

3.1 Metadata Storage and Retrieval

A typical search in a DL is performed on metadata which describe the document content end their bibliographic information. Three different approaches have been adopted until now to support document retrieval in digital libraries: (a) use of relational databases; (b) use of information retrieval engines; (c) full sequential scan of metadata records. Unfortunately, these approaches did not prove to be effective for DL applications: designers had to face the problem of choosing the right compromise between efficiency of the search systems and complexity of the metadata schema. The result of this compromise is that in mostmany cases DLs use very simple and flat metadata schemas such as Dublin Core [2].

Solution (a) requires that metadata should be converted into relational schemas. This is easy for simple flat metadata schemas, such as Dublin Core, but it far more difficult for complex and descriptive metadata schemas, such as ECHO [14],

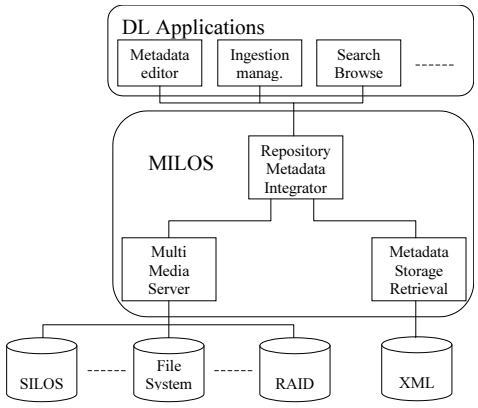

Fig. 1. General Architecture of MILOS

MPEG-7 [5], IFLA-FRBR [10], P/META [13]. Moreover, a query on these metadata must be translated into complex SQL queries at relational level, resulting in many expensive joins to implement tree structure traversals. Thus, the resulting search performance is often unacceptable. However, even with flat metadata schemas, pure relational databases do not offer all functionalities needed for an effective retrieval, such as full text search.

Solution (b) uses full text search engines [22] to index metadata records. In this case the main emphasis is devoted to the textual information contained in metadata fields. Many text search engines offer the fielded indexing capability, where text contained in different fields is independently indexed. However, applications are limited to relatively simple and flat metadata schemas. In addition, it is not possible to search by specifying ranges of values.

Solution (c) is very trivial and inefficient. It is not practicable in applications that pretend to be more than toy systems. In this case no indexing is performed on the metadata and the custom search algorithms always scans the entire metadata set to retrieve searched information.

We successfully attempted a different approach: we have designed and implemented an enhanced native XML database/repository system with special features for DL applications. This is especially justified by the well known and accepted advantages of representing metadata as XML documents. Metadata represented with XML might have arbitrary complex structures, which allows to deal with complex metadata schemas, and might be easily exported and imported. Our XML database can store and retrieve any valid XML document. No metadata schema or XML schema definition is needed before inserting an XML document, except optional index definitions for performance boosting. Once an arbitrary XML document has been inserted in the database it can be immediately retrieved using XQuery. This allows DL applications to use arbitrary (XML encoded) metadata schemas and to deal with heterogeneous metadata, without any constraint on schema design and/or overhead due to metadata translation.

We decided not to use a commercial XML database system (e.g. Tamino [9]) because of our specific operational requirements:

1. Particular attention must be given to the performance of search and insert operations.
2. It is not necessary to enforce a database-like transactional mechanism, since update operations are quite rare compared to search operations. Editing of complex multimedia objects and their metadata, can be based on a sort of check-out/check-in mechanism.

Thus, our native XML database/respository system is simpler than a general purpose XML database system, but offers significant improvements in specific area: it supports standard XML query languages such as XPath [18] and XQuery [19], and offers advanced search and indexing functionality on arbitrary XML documents; it supports high performance search and retrieval on heavily structured XML documents, relying on specific index structures [15,23], as well as full text search [22], automatic classification [20], and feature similarity search [17]. The system administrator can associate an index to a specific XML element. For instance, the tag `<abstract>` can be associated with a full text index and to an automatic topic classifier that automatically indexes it with topics chosen from a controlled vocabulary. On the other hand, the MPEG-7 `<VisualDescriptor>` tag can be associated with a similarity search index structure and with an automatic visual content classifier. The XQuery language has been extended with new operators that deal with approximate match and ranking, in order to deal with these new search functionality.

In our database every XML document is identified by an URN. Therefore, relationships and links among documents - even if they are stored in different repositories - can be easily and unambiguously represented.

3.2 Multi Media Server

Different DL applications may have different storage and access needs. For example, very small DLs might store documents on standard hard disks, while more mission critical applications might need to store documents on arrays of disks, possibly duplicating and distributing content on several sites. Digital libraries dealing with huge archives of video documents, might need to store them on digital tapes maintained in silos, and to have arrays of disks used as temporary storage for frequently used documents. In addition, we must consider that a DL may scale over time, when the number of documents grows over a certain limit or faster access is needed.

DL applications might also use different delivery strategies. For example, a small DL might serve documents using a normal web server, while heavily accessed DLs might need to use replication and load balancing strategies to guarantee high performance access to content. A video DL might use high performance video servers to stream videos in real time to users [21].

The MMS allows the programmers of the DL applications to be unaware of all these issues. The key idea is that the DL application should deal with documents

in a uniform way, independently of the specific strategy used to manage them. Thus, the MMS identifies all documents with an URN and maintains a mapping table to associate URNs with actual storage locations. Applications use the URN to get or store documents from the MMS, which behaves as a gateway to the actual repository that stores the document. The system administrator can define rules that make use of MIME types, to specify how the MMS has to store a document of a specific type. For example, the rule may specify that an MPEG-2 video has to be be stored in a tape of a silos, while an image will be stored in an array of disks.

A special care is taken to deal with the actual access protocols offered to retrieve the documents. An application will refer a specific document always using its URN. However, the retrieval of the document should be done using an access protocol compatible with the storage and delivery strategy associated with the document. For instance, when the document is stored in a web server it will be retrieved with an HTTP request. On the other hand, suppose that a video document is served through a commercial video server such as the Helix Universal Server [4]; in this case the real time streaming of the video will be obtained using RTSP [6]. When an application requires to retrieve a document, the MMS will translate the given URN into a specific handle (for instance an RTSP URL) that the application will use to access the document.

3.3 Repository Metadata Integrator

The RMI manages the accesses to the document and metadata repositories and supports metadata mapping to guarantee metadata independence. The mapping of application requests into requests compatible to the metadata schema actually managed by the MCMS is accomplished by defining a set of schema mapping rules. The main purpose of this mapping is to translate application requests into XQuery queries compliant to the stored metadata. This mechanism allows the RMI to translate names of fields (such as Title, Author, etc.) known to the DL application, into requests to the MSR without the need of knowing the specific schema model adopted. When a new XML schema is introduced, the system administrator must specify the mappings for the new metadata.

Each mapping rule specifies how to translate the name of a metadata field, known to the application, into an XPath expression that specifies the XML path names that should be used to access that metadata field in the target metadata schema. A generic mapping rule has the following structure:

metadataType[.Name] = <RE_XPath>,<SE_XPath>* where

1. The *metadataType* field identifies the metadata model used by the application e.g. DublinCore, SCORM, MPEG-7 etc;
2. *Name* is the name of a metadata field requested by the application e.g., Title, Author, etc. If empty, it means that the rule applies to all metadata fields of the specified *metadataType*;
3. *<RE_XPath>* (Retrieved Element XPath) is the XPath corresponding to the XML element that will be retrieved with this field;

4. <*SE_XPath*> (Searched Element XPath) is the XPath, under <*RE_XPath*>, of the element that contains the value of the metadata field used for searching.

As an example, let us consider a DL for e-Learning applications, where the Learning Objects in the repository have a complex metadata structure, based on SCORM [11]. Suppose that we want to search SCORM metadata trough Dublin Core. We can use the following mapping rules:

```
dc.title = /lom, general/title/langstring
dc.description = /lom, general/description/langstring
```

They specify that the Dublin Core metadata fields 'dc.title' and 'dc.description' can be searched in SCORM respectively by means of the XPath string `lom/general/title/langstring`, and `lom/genaral/description/langstring`. The whole `<lom>` element will be retrieved when `<langstring>` contains the desired value. Note that, the `<title>` and `<description>` SCORM XML elements do not contain the title text of the document, but the element `<langstring>`, which in turn contains the real text.

Let us now explain how the mapping directives are used by RMI to generate the XQuery query. The RMI allows applications to search on metadata by using the *findExactMatch* method:

findExactMatch(**string** MetadaType, **vector of string** fields, **vector of string** values, **string** returnFields),

This method searches for a set of metadata records of the specified *MetadaType*. The *fields* parameter is a vector of (application known) names of metadata fields, of the *MetadaType*, to search for. The *values* parameter specifies the values that the fields must match (the different fields are searched by using the boolean connective **AND**). Finally, *returnFields* specifies the fields of the retrieved records (i.e. RE_XPath) that the application wants to know. The method translates the request into an XQuery query as follows:

1. for each triple <*MetadaType,* $value_i$*,* $field_i$>, specified in the *findExactMatch*, RMI searches the mapping rules matching $MetadaType.field_i$ to fetch the corresponding XPath strings RE_XPath_i and SE_XPath_i;
2. for each pair <*MetadaType,* $returnField_i$ >, specified in the *findExactMatch*, RMI searches the mapping rules matching $MetadaType.returnField_j$ to fetch the corresponding XPath strings $RE\_XPath_{ret_j}$ and $SE\_XPath_{ret_j}$.
3. check that all the strings RE_XPath_i and $RE\_XPath_{ret_j}$ are the same string and call that string RE_XPath, otherwise fail and stop;
4. finally, combine the XPath strings RE_XPath, SE_XPath_i, and $SE\_XPath_{ret_j}$ to generate the XQuery query, as follows:

for $\$a$ **in** RE_XPath
where $\$a/SE\_XPath_1 = value_1$ **and** ... **and** $\$a/SE\_XPath_n = value_n$
return $\$a/SE\_XPath_{ret_1}$ ... $\$a/SE\_XPath_{ret_m}$

Example: Suppose that an application wants to use Dublin Core to search SCORM metadata having a specific title, and wants to have back the corresponding descriptions. In this case we have $MetadataType = dc$, $field_1 = title$, $returnField_1 = description$. Applying the previous mapping rules we obtain:

for *$a* **in** /lom
where $a/general/title/langstring = $value_1$
return $a/general/description/langstring

4 Field Trials

In order to verify and demonstrate the flexibility and efficiency of MILOS in managing different heterogeneous DL applications, we took four data sets used by four different existing DLs and we built the corresponding DL applications on top of MILOS. The data sets that we considered consist of documents and metadata of very different nature: the Reuters data set [7], the ACM Sigmod Record dataset [8], the DBLP data set [1], and the ECHO data set [14].

The DL applications that we built use the same MILOS installation and all data sets were stored together. The functionality of MILOS allows individual applications to selectively access data and metadata of their interest or to perform cross-library search. Each DL application consists of a specific search and browsing interface (built according to the data managed) and a bulk import tool. The search and browse interfaces were built as web applications using Java Server Pages (JSP). The bulk import tool was a simple Java application. On average, the effort required to build each application from scratch was one week of work of a single skilled person. This, we believe, is really a little effort compared to the cost that would have been required to build from scratch a DL, without general purpose tools, or the cost that would have been required to translate and adapt the data and metadata to cope with the requirements and restrictions of an existing DL system.

We built the browse and retrieval interface from scratch. However, we are currently working to develop a tool supporting the automatic generation of the browsing and retrieval interface according to data and metadata fields. This will contribute to a further reduction of the cost of building DL applications.

All applications resulted to be very efficient. We installed the system, the applications, and the data on a single computer equipped with a Pentium 1.8 GHz and 1 Gb of RAM, running Windows 2000 server. We have used JAX-RPC as SOAP application server to run MILOS. Applications have been tested by 30 users operating at the same time from remote workstations, and executing a predefined search intensive job. On average the response time of the system was below 1 second. Notice also that for more intensive uses of the system, the underlying Web Service technologies offer plenty of solutions to guarantee scalability exploiting techniques of replication, load balancing, resource/connection pooling etc.

The **Reuters data set** [7] contains text news agencies and the corresponding metadata composed of Reuters specific metadata including titles, authors, topic categories, and extended Dublin Core metadata. The data set contains 810,000 news agencies (2.6 Gb) with text and metadata both encoded in XML. We associated the full text index and the automatic topic classifier to the elements containing the body, the title, and the headline of the news. Other value indexes were associated with elements corresponding to frequently searched metadata, such as location, date, country. The search interface allows the user to perform integrated text, category, and exact match search.

Both the **ACM Sigmod Record** [8] and the **DBLP** data-sets [1] consist of metadata corresponding to the description of scientific publications in the computer science domain. The ACM Sigmod record is composed of 46 XML files (1Mb), while the DBLP data-set is composed of a sinlge large (187Mb) XML file. Their structure is completely different even if they contain information describing similar objects. For these two datasets we built one single DL application that allows one to access both. The MILOS mapping is used to translate application requests in the two schemas. We associated a full text index to the elements containing the titles of the articles, and we associated other value indexes to other frequently searched elements, such as the authors, the dates, the years, etc. The search and browse interface allows users to search for articles by various combinations full text and exact/partial match of elements. In addition it allows user to browse results by navigating trough links (and implicitly submitting new queries to MILOS) related to the author, journal, conference, etc.

The **ECHO data set** [14] includes historical audio/visual documents and the corresponding metadata. ECHO is a significant example of the capability of MILOS to support the management of arbitrary metadata schemas. The metadata model adopted in ECHO, based on IFLA/FRBR [10] model, is rather complex and strongly structured. It is used for representing the audio-visual content of the archive and includes among others, the description of videos in English and in the original language, specific metadata fields such as Title, Producer, year, etc., the boundaries of scenes detected (associated with a textual descriptions), the audio segmentation (distinguishing among noise, music, speech, etc.), the Speech Transcripts, and visual features for supporting similarity search on key-frames. The collection is composed of about 8,000 documents for 50 hours of video described by 43,000 XML files (36 Mb). Each scene detected is associated with a JPEG encoded key frame for a total of 21GB of MPEG-1 and JPEG files. Full text indexes where associated to textual descriptive fields, similarity search index where associated with elements containing MPEG-7 image (key frames) features, and other value indexes where associated with frequently searched elements. The search and retrieval interface (Figure 2) allows users to find videos by combining full text, image similarity, and exact/partial match search. Users can browse among scenes, and corresponding metadata. The original ECHO DL application [3], was built using a relational database, and translating all metadata in a relational schema. Even simple searches required several (up to 10 or more) seconds to be processed. With MILOS we had a dramatic improvement

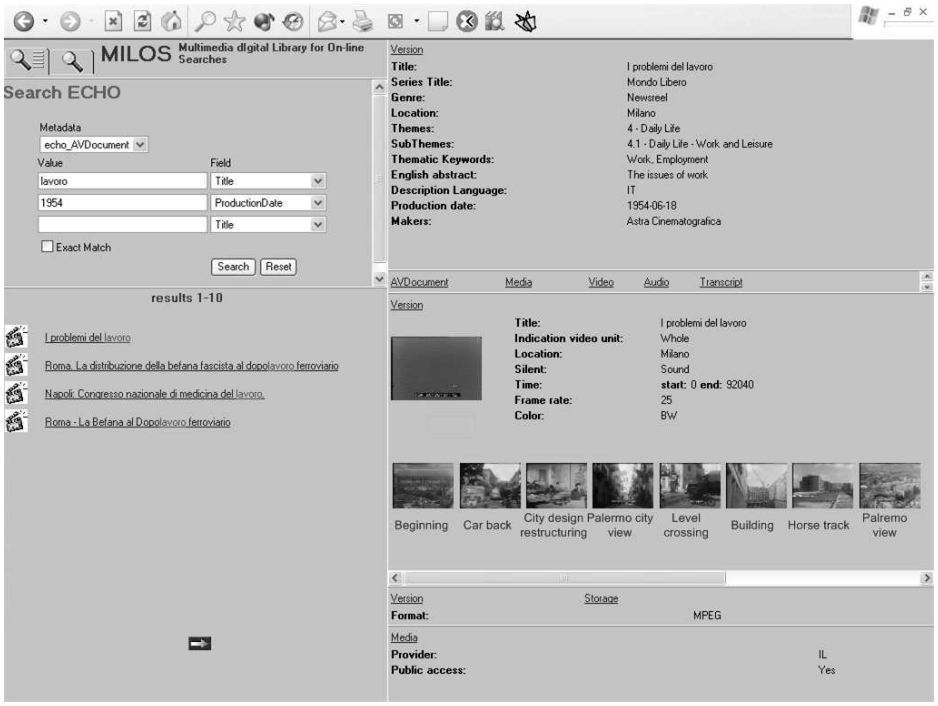

Fig. 2. The ECHO retrieval interface implemented in MILOS

of performance, being able to serve requests in less than one second even with several users accessing the system.

5 Conclusion

This paper described the architecture of the MILOS Content Management System and the solutions adopted to obtain a system that is flexible in the management of documents with different types of content and descriptions, and that is efficient and scalable in the storage and content based retrieval of these documents. In particular, we described the approach adopted to support the management of different metadata descriptions of multimedia documents in the same repository. This goes towards the solution of the challenging problems of interoperability among different metadata descriptions. The proposed solution, based on the use of a mapping mechanism among the metadata fields of the different models, has been practically experimented by using the MILOS system to archive documents belonging to four different and heterogenous collections which contain news agencies, scientific papers, and audiovideo documentaries. The archiving of these documents was straightforward and it only required the creation of the mapping file and the development of the user interfaces to archive and to search the documents.

An evolution of this activity is foreseen in several directions: on one side we are working to improve the retrieval capabilities of the Metadata Storage and Retrieval component; on the other side, we are working with partners of the ECD [12] project on the automatization of the mapping between different metadata schemas, by using thesaurus and cross-language vocabularies [16].

References

1. DBLP computer science bibliography. http://www.informatik.uni-trier.de/ ley/db/.
2. Dublin Core Metadata Initiative. http://dublincore.org/.
3. Echo: European CHronicles On-line. http://pc-erato2.iei.pi.cnr.it/echo/.
4. Helix Universal Server. http://www.realnetworks.com/products/server/index.html.
5. Motion picture experts group. http://mpeg.cselt.it.
6. Real Time Streaming Protocol. http://www.rtsp.org/.
7. Reuters corpus. http://about.reuters.com/researchandstandards/corpus/.
8. Sigmod record, xml edition. http://www.acm.org/sigs/sigmod/record/xml/.
9. Tamino XML Server. http://www.softwareag.com/tamino/.
10. IFLA study on the functional requirements for bibliographic records, 1998. http://www.ifla.org/VII/s13/frbr/frbr.pdf.
11. Shareable content object reference model initiative (scorm), the xml cover pages, October 2001. http://xml.coverpages.org/scorm.html.
12. ECD - Enhanced Content Delivery, 2002. http://ecd.isti.cnr.it/.
13. P_META, the EBU metadata exchange scheme, 2003. http://www.ebu.ch/en/technical/publications/Tech_3000_series/tech3295/.
14. G. Amato, D. Castelli, and S. Pisani. A metadata model for historical documentary films. In J. L. Borbinha and T. Baker, editors, *Proc. of the 4th European Conference ECDL*, pages 328–331. Springer, 2000.
15. G. Amato, F. Debole, F. Rabitti, and P. Zezula. YAPI: Yet another path index for XML searching. In *ECDL 2003, 7th ECDL Conference, Trondheim, Norway, August 17-22, 2003*, 2003.
16. D. Beneventano and al. Semantic integration and query optimization of heterogeneous data sources. In *OOIS Workshops*, pages 154–165, 2002.
17. C. Böhm, S. Berchtold, and D. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, September 2001.
18. W. W. W. Consortium. XML path language (XPath), version 1.0, W3C. Recommendation, November 1999.
19. W. W. W. Consortium. XQuery 1.0: An XML query language. W3C Working Draft, November 2002. http://www.w3.org/TR/xquery.
20. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambidge University Press, 2000.
21. D. J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe. Multimedia storage servers: A tutorial. *IEEE Computer*, 28(5):40–49, May 1995.
22. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1983.
23. P. Zezula, G. Amato, F. Debole, and F. Rabitti. Tree signatures for xml querying and navigation. In *Database and XML Technologies, XSym 2003*, volume 2824 of *LNCS*, pages 149–163. Springer, 2003.