# An Architecture for Type-based Repository Systems

Leonardo Candela, Paolo Manghi, and Pasquale Pagano

Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" – CNR
Via G. Moruzzi, 1 - 56124 Pisa - Italy
{candela, manghi, pagano}@isti.cnr.it

**Abstract.** In the last decade, Institutional Repositories played an important role in knowledge dissemination and sharing. However, the requirements of today's User Communities have evolved a lot since their original conception. Repositories are expected to provide access to knowledge that escapes the boundaries of their native Institution assets and whose representation goes beyond that of simple digital counterparts of paper documents. Such a novel scenario poses new challenging requirements for Repository systems. This work presents a reference architecture of a Repository system capable of accommodating such needs by (*i*) empowering the support for managing compound digital objects, (*ii*) providing mechanisms to create digital objects by aggregating already existing objects made available by third party data sources, and (*iii*) enabling explicit sharing of knowledge across various repositories.

## 1 Introduction

The notion of Repository, more precisely *Institutional Repository*, plays a central role in the production and dissemination of knowledge in digital form [1,2,3,4,5]. In particular, this understanding is stressed in [1] where "a university-based institutional Repository is a set of services that a university offers to the members of its community for the management and dissemination of digital materials created by the institution and its community members. It is most essentially an organizational commitment to the stewardship of these digital materials, including long-term preservation where appropriate, as well as organisation and access or distribution". The benefits of replacing traditional knowledge publication and communication channels with digital Web-based applications, combined with world-wide initiatives promoting free and unrestricted access to publications (*Open Access* [6]) and interoperability (*Open Archives Initiative* [7]), resulted in a proliferation of the players willing to act as content providers and introduced the need for implementing new technologies supporting it. Indeed, several Institutional Repository Systems have been developed in the past and are currently adopted by a multitude of Institutions to serve their User Communities needs [8]. However, even if such Systems are still effective and widely in use, in the last few years User Communities became increasingly interested in goals that cannot be fully satisfied by any of the available technology:

**Richness of knowledge** Knowledge may be of arbitrarily complex structure, strictly depending on the Institution information needs. Recently, such an issue has been tackled by means of *compound objects*, which are sets of digital objects interconnected by relationships expressing the nature of their bindings. The Repository should make it easy to both manage and control such kind of objects and enable construction of applications on top of them.

**Federation of knowledge** Sharing is an intrinsic characteristic underlying knowledge production, i.e. Repositories should be able to easily share knowledge with other Repositories they are federated with. The Repository should provide tools for partaking one or more Repository federations and integrate with other Repositories in a transparent way.

**Aggregation of knowledge** Aggregation is another key aspect of knowledge production, i.e. new knowledge is obtained by aggregating existing knowledge. The Repository should make it possible to easily collect and aggregate objects residing out of its federation boundaries and combine them with local objects.

This paper introduces a reference architecture, i.e. an architectural design pattern accommodating a set of known desiderata, for a Type-based Repository System capable of effectively and efficiently deal with the above mentioned requirements. This reference architecture acts as a blueprint for the implementation of future generation Repository software approaching the previous issues in a systematic and organised way. The main constituent parts of such a complex system are identified and their distinguishing characteristics are presented.

The remainder of the paper is organised as follows. Section 2 presents a Repository System reference architecture and discusses the main component parts. Section 3 compares the proposed architecture with respect to current state of the art systems and technologies. Finally, Section 4 concludes the paper and discusses future research issues.

## 2 The Architecture

In what follows we present an *Architecture for Type-based Repository Systems* (T-Repositories) designed to satisfy the evolving needs of modern User Communities. T-Repositories are managed by Institution *Repository Managers*, used by the User Communities, and can partake one or more federations, thereby becoming *federation nodes*. Repository Managers can customize their nodes so as to (*i*) manage compound objects matching the knowledge structure of interest to the Community, (*ii*) share their objects across different nodes within the federation, and (*iii*) import objects residing outside the federation boundaries (e.g. external data sources). For example, a Community can customize a node for managing *e-Thesis* represented as compound objects whose structure is that of a list of *chapter* objects, in turn consisting of PDF files; e-Thesis management includes ingestion of such objects and search. In addition, e-Thesis should feature (*a*) references to a list of *paper* objects relevant to the work, to be locally aggregated from a selection of external Institutional Repositories; and (*b*) references to a list of *data set* objects, representing experimental data and tools used to evidence the thesis result. Finally, the Community can devise "global" search functionality over the union of all e-Thesis objects present, if any, in the federation nodes.

The T-Repository Architecture is depicted in Figure 1. The picture displays only two T-Repository nodes of a federation, which may include an arbitrary number of nodes. T-Repository nodes are organized into five main Areas:

**Logical Area** The Logical Area enables Repository Managers to define the *typed object sets* matching the Community requirements. An object set is a container of compound digital objects of the same *type*, i.e. respecting the same structure and responding to the same operators; the equivalent, in Relational Database Systems, is a table with a given attribute structure, which implicitly defines the applicable SQL commands. Each T-Repository node defines and exposes through the Logical Area a number of objects sets, characterized by their unique name and their type. As in traditional type systems, object set types can be of various kinds and be combined in an algebra to define more complex types.

**Physical Area** The Physical Area delivers advanced facilities for object set storage and access. The Area is independent from the Logical Area and can therefore adopt and implement different strategies on different nodes. In principle, the best implementations are those exploiting the type information derived from object sets to maximize access performance and minimize the amount of disk to be used.

**Federative Area** The Federative Area enables the inclusion into the Logical Area of objects natively hosted into other federated T-Repository nodes. This can be done in two ways: (*a*) by linking such objects from a local compound object, (*b*) by sharing the object set of another node. In both cases, being part of the same federation, both T-Repositories can potentially add, delete, and modify objects of such a set.

**Mediation Area** The Mediation Area manages special object sets that can be populated with digital objects residing in data sources external to the federation. The Area can either import such objects to *coerce* them into corresponding local compound objects, or, when the data source permits it, keep object sets and relative data sources *synchronized* in both directions.

**Presentation Area** The Presentation Area offers management interfaces for both end-users and applications. Repository Managers can benefit from User interface generation tools, which exploit the type information of the object sets declared into the Logical Area to compose standard GUIs. Application interfaces enable direct access to the individual object sets, through the T-Repository APIs or through standard access interfaces, such as OAI-PMH and SRU/SRW, to be extended with compound object functionalities.

In the following sections we discuss in detail the main features of these Areas and the relationships between them.

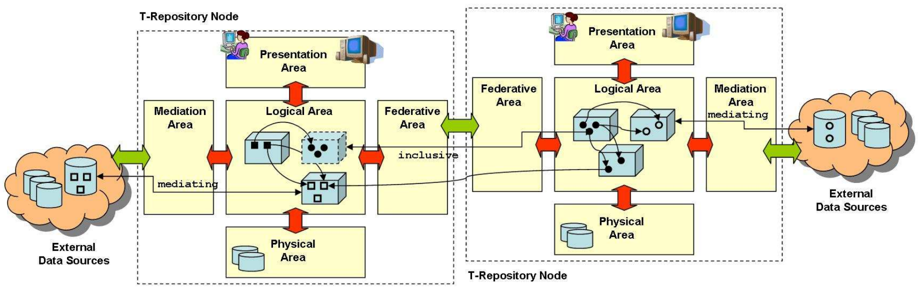


**Fig. 1.** The T-Repository Architecture

### 2.1 The Logical Area

The Logical Area enables Repository Managers to create and manage the object sets required to match the Community knowledge needs. Such sets are defined and created by providing a unique name, an object *type*, and a *metadata format* of preference[1]. A type, as generally intended in programming languages and databases, defines the structure of the objects to be contained in the set and determines the operators that can be applied over them. For example, Figure 2 illustrates two object sets `A` and `B`. `A` is of type `Aggregation(B,MDF)`, which means that an object `obj` in `A` represents an unordered collection of objects belonging to the set `B`, and is described by a metadata record of format `MDF`. `B` is of type `Basic(T,MDF')`, which means that an object `obj'` in `B` contains a digital asset of type `T`, e.g. a file, and is described by a metadata record of format `MDF'`. Due to its specific type structure, `obj` responds to methods such as `obj.addObject(obj')` and `obj.removeObject(obj')`, which respectively add and remove an object `obj'` present in `B` from the aggregation of `obj`; `obj.numberOfObjects()`, which returns the number of objects in `B` currently aggregated by `obj`; `obj.aggregated()`, which returns the set of objects in `B` aggregated by `obj`; and `obj.searchObjects(P)`, which returns the subset of `obj.aggregated()` whose objects have metadata records satisfying the predicate `P` over the format `MDF'`.

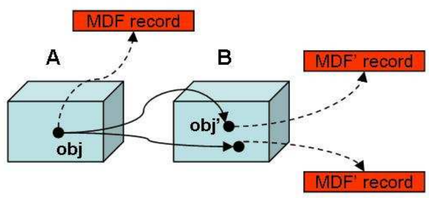


**Fig. 2.** Object sets and Types

Types statically determine the nature of all objects within a set [9]. This means that the Logical Area can ensure that only the right operators will be applied to the objects in a given set and that objects can be stored or linked from other objects only if their structure matches the expected type. T-Repositories offer an algebra of types, which can be combined to define more complex types. Specifically, four kinds of types can be identified: *basic*, defining file objects of specific format, e.g. PDF files, DOC files, generic text files, JPG files, generic multimedia files, but also ground values, such as Integer, Strings, URI, etc.; *complex*, defining compound objects with special operators, such as `Aggregation`, `Relation`, `Version`, etc.; *inclusive*, including and/or sharing objects sets from other Logical Areas or defining sets unifying all object sets in the federation respecting a given type structure; and *mediating*, defining object sets whose content is synchronized with that of data sources residing outside the federation.

Application developers gain great benefits from T-Repository controlled environment, which can be customized to handle the object types needed by a Community while ensuring the static correctness of their applications. In particular, it is important to stress that types can be used to instantiate the objects sets required to represent the *document model* which best fits the conception of digital document perceived by the Community; e.g. the digital e-Thesis above. This approach, inspired by traditional database systems, solves most of the problems arising with traditional Repository Systems, which tend to provide pre-defined document models that do not necessarily match the Community needs. Typically, such models can be either very generic, e.g. Fedora

[1] More than one metadata format can be assigned to the same object. For simplicity of presentation we assume here is only one.

[10], or be quite rigid w.r.t. an idea of digital document, e.g. DSpace [11]. In the former case, building and maintaining applications becomes expensive for Institutions, whose developers must "embed" the notion of Community document model, generally more detailed than the Repository's, within the software components they have to code. In the latter case, either the Community adopts a fixed document model, thereby partly renouncing its User requirements, or developers adapt the Repository document model to the Community's one by integrating other technologies to the Repository and, again, coding complex software components.

T-Repositories support an environment where both the Community and developers of an Institution can realize their knowledge and development expectations, respectively. For instance, the following types define the sets required to create the document model of the e-Thesis described above:

- `chapters = Basic(PDF,DC)`: set of objects, each associated to a PDF file described by a Dublin Core record.
- `e-thesis = Aggregation(chapters,ETD)`: set of objects, each described by the ETD format (Electronic Theses and Dissertations [12]) and associated to a number of objects in `chapters`.
- `papers = Coercion(Basic(PDF,DC),DataSource)`: set of objects of type `Basic(PDF,DC)` obtained by importing objects from the external data source referenced by `DataSource`, here assumed to be an OAI-PMH compliant Repository. Repository Managers will have to provide the parameter settings required to configure the mediation, such as the access point, e.g. the Repository's OAI-PMH URI, and the mapping from the data source object structure, i.e. OAI-Item, onto the destination object type structure, e.g. `Basic(PDF,DC)`. `Coercion` types enforce synchronization in one direction only, e.g. modifications to objects in `papers` are not propagated to corresponding objects in `DataSource`. Other forms of mediating types can be defined, enabling synchronization in both directions; e.g. `Synchro(Basic(PDF,DC), DataSource)` would enforce such strategy. Such types manage only specific typologies of Data Sources; e.g. OAI-PMH would not allow this definition, while FTP sites could.
- `citations = Relation(e-thesis,papers,N:N)`: set of objects, each representing the relationship between an object in `e-thesis` and one object in `papers`. The option `N:N` releases any constraint on the relations between objects in the two sets; other options, such as *1:1*, *N:1*, and *1:N*, do instead enforce relationship constraints. For example, *1:N* would ensure that `citations` cannot contain two objects relating the same object in `papers` to two different objects in `e-thesis`.
- `datasets = Basic(URI,DSF)`: set of objects, each representing the URI through which a data set can be reached and described by a record in `DSF` format. We assume `DSF` to be designed by the Community to contain fields such as `tool` used for elaborating the data, `creator` of the data, and `date` of creation.
- `experiments = Relation(e-thesis,datasets,N:N)`: set of objects, each representing the relationship between an object in `e-thesis` and one object in `datasets`. The same data set can be used in different thesis works.
- `e-thesisFederation = IncludeAll(e-thesis)`: set of objects obtained by unifying all sets of objects in the federation that do respect the type of `e-thesis`. Here `e-thesis` is used to refer to a type definition. Operations over `e-thesisFederation` are performed over all object sets, no matter their name nor where

they are located, that do respect such type structure. Other forms of inclusive types can be defined, sharing one specific object set; e.g. `IncludeSet(R.e-thesis)` includes into the Logical Area the set `e-thesis` from the T-Repository `R`, if it exists.

### 2.2 The Physical Area

The Physical Area is entitled to store the objects sets natively defined by the local Repository Manager. The Area can benefit from the relative set type definitions to allocate the correct disk space and build the structures necessary for efficient access, such as indexes. For example, objects in `e-thesis`, being associated to objects in `chapters`, will be likely stored in such a way that the retrieval of the chapters relative to a thesis will be possible with a minimal number of disk accesses (as may happen in Relational Database systems, given the definition of the tables and the relationships between their primary and foreign keys). Similarly, indexes for ETD format fields may be created for `e-thesisFederation`, in order to perform local efficient queries on local and remote objects; object sets of video files can be stored and exported through streaming interfaces.

Furthermore, the independence from the Logical Area enables different implementations of the Physical Area functionalities, which, for example, may rely on Grid storage systems [13], Relational Databases, file systems, or other technologies.

### 2.3 The Federative Area

The Federative Area provides the functionality enabling a number of T-Repositories, i.e. a federation, to share their Logical Areas. Given a T-Repository federation, it is a responsibility of the individual Repository Managers to include into the local Logical Area the remote object sets they need, by defining the appropriate inclusive types; e.g. `e-thesisFederation = IncludeAll(e-thesis)` and `e-thesis = IncludeAll(R.e-thesis)`. In particular, Repository Managers may establish different federative access policies governing their local objects sets. Therefore, depending on the local Repository Manager needs and remote set policies, remote objects can be moved locally, either synchronized or replicated, or be only **referred from** their original site.

The Federative Area can be implemented in different ways. Due to the distributed and autonomous nature of T-Repositories which might decide to join or leave one federation at any time, *Peer-to-Peer Systems with Overlay Network* seem to be an appropriate solution. In such systems, *peers*, i.e. distributed nodes, can join or leave the system in any moment, and, in order to accomplish their tasks, require some interaction with other peers. The main issue in P2P Systems is that of satisfying requests from peers to discover the peers they need. In some architectures, special peers play the role of *superpeers* and form an overlay network providing advanced facilities for managing peer registration and discovery [14]. T-Repository nodes could constitute a federation by interpreting the role of peers and superpeers of an overlay network. The overlay network would keep track of object sets and relative access policies for all Logical Areas of the T-Repositories registered to the federation. Any local request for accessing remote object sets would then be transformed into a request to the overlay network for discovering matching T-Repository nodes and/or object sets. The Federative Area of a node would be in charge of this data exchange and make it transparent to the local Logical Area.

### 2.4 The Mediation Area

The Mediator Area offers the tools for managing the object sets associated to external data sources through mediating types. For each data source, the Area holds a configuration package, to be set up by the Repository Manager, which establishes the mediation criteria between the source and the destination set. The nature of these packages depends (*a*) on the nature of the data source, e.g. OAI-PMH, FTP, File Systems, Databases, (*b*) by its document models, e.g. OAI-Item structure, special XML files, DB tables, and (*c*) by the kind of access interfaces provided, i.e. read, add, update, delete. The Mediation Area, given a Data Source, provides Repository Managers with semi-automatic tools and interfaces for the management of the relative configuration package.

In particular, compound objects may be coerced from an external data source into a local set or be synchronized between the two. Such operations are based on mappings and transformation rules to be defined by the Repository Manager, and can be performed at different times, either automatically or manually, by the Mediation Area or by the Manager. A typical example of data source coercion is that of harvesting of OAI-PMH Repositories [7] into basic types, but many others are possible, from DIDL-aDORe Repositories, Relational Databases to FTP sites, and others, into complex types of any form. Data Source synchronization can instead be applied to a subset of Data Source typologies, whose access interfaces enable external entities for write/update/delete their content.

### 2.5 The Presentation Area

The Presentation Area builds on top of the Logical Area and provides the interfaces to access the object sets defined by the Repository Manager. Similarly to the Physical Area, the Presentation Area can exploit type information in order to automatically generate three types of view over the Logical Area:

**User-oriented view** The user interface, through which users can ingest, search, and generally manage their objects. Object set types allow for the realization of automatic interface generation tools, e.g. *MS Access*-like wizards. Such tools, given a Logical Area, can guide Repository Managers through the automatic generation of the object set user interfaces. Once generated, such interfaces can be adapted by developers in order to match specific graphical expectations or be modified to satisfy Community specific needs. For example, from the *e-Thesis* set we can implicitly generate Web interfaces to create a new thesis, insert the relative ETD format fields, and insert the required chapters. For each new chapter, the type of `chapters` entails the creation of an interface enabling the upload of a PDF file and the insertion of the relative Dublin Core fields. Of course, developers may decide to interact directly with the object sets APIs and generate their specific applications.

**Web-oriented view** Following the ORE model principles [15], such interfaces enable URI-based navigation of object sets. To this aim, sets and compound objects do correspond to unique URIs, accessible through HTTP and returning an XML representation of the set or the object involved. In particular, the XML of a compound object contains the name of the sets that include the object, its metadata record, and other information that depends on the object type structure; the XML of an object set contains its name, its type, and all references to the objects it contains. For example, objects in `e-thesis` will correspond to an XML file whose structure will contain the name `e-thesis`, the ETD record, and the URIs relative to the XML representations of the associated chapter objects. Objects in `chapters`, will be described by an XML file containing the name `chapters`, the Dublin Core record, and the reference to the PDF document.

**Application-oriented view** The APIs accessible locally or through the network by applications interested in interacting with content from a specific object set. These can be proprietary to the T-Repository object set definition, which strictly depend on the type, or standard access interfaces, such OAI-PMH and SRU/SRW. In the latter case, the OAI-PMH interface is to be extended with new methods for the extraction of the XML representation of compound objects (DIDL format could be used [16]). For example, through such methods it would be possible to harvest full e-thesis objects, i.e. an XML containing the "tree" formed by the thesis object and its associated chapter objects.

## 3 Related Work

Various Repository Systems and related technologies, e.g. [10,11,17,18,19,20,21], have been developed. In this Section we provide a brief overview of few of them focusing on the most widely used systems and discussing how they only partially accommodate the issues addressed by the proposed architecture.

DSpace [11] is an open source digital repository system designed to operate as a centralized system for capturing, storing, indexing, preserving, and redistributing documents in digital formats. It has been designed to satisfy the requirements of a university research faculty for managing its intellectual outputs. Despite the richness of its document model, i.e. it supports *collections* of *items*, each organised into *bundles* of *bitstreams*, the Logical Area does not support native mechanisms for the definition of specialized compound objects. The Presentation Area is hardcoded on the document model and comes with the Repository installation. An interesting feature is the exploitation of SRB [23] as a further implementation of the Physical Area, in addition to the local file system, for storing the repository objects. Federation and Mediation Areas are not regarded.

Fedora [10] is a repository service for storing and managing complex objects. The Logical Area grounds on a general purpose document model, whose flexibility promotes the system as central to many implementations. The model allows for combining into a single digital object any number and any variety of data streams either residing into the local repository or anywhere on the web and to define objects relationships forming a Repository graph. Interesting, the system is capable of dynamically providing multiple views over the same object through *disseminators*, i.e. web services processing the stored data stream of the object and producing new ones dynamically. However, the Logical Area does not support native mechanisms for the definition of specialized compound objects. Accordingly, the system has a very generic Presentation Area, which fits the graph document model and serves more as a Repository administration tool, i.e. objects and disseminators management. No explicit features supporting a Federative Area have been currently implemented, while some form of indirect Mediation Area are in place, capable of coercing special files into corresponding Fedora objects.

Recently, EPrints [17] version 3 has been released. This novel version of an historically important Repository software presents many improvement w.r.t. the previous versions. The Logical Area offers a pre-defined document model, not supporting compound objects, and a Presentation Area based on it. Federative and Mediation Area are not envisaged at the moment.

OpenDLib [18] is a Digital Library management system developed at ISTI-CNR. Its Logical Area subsumes a typed document model [24] allowing to represent a large set of compound and multimedia information objects. However, the number of type abstractions it offers and the way these can be combined, limits the richness of the document models that can be instantiated in the Repository. Its Presentation Area depends on the specific instantiation of the Logical Area, while its Physical Area is file system based; recently, however, a Grid oriented Physical Area has been implemented [25] resulting in a system exploiting third party storage and computing facilities to manage information objects very demanding with respect to the consumption of these resources. The system realises the Federative Area via a Manager Service, which permits the Repository instances forming the federation to dynamically discover each other.

## 4 Conclusion

Repositories occupy a fundamental role in the knowledge dissemination and sharing process and a number of technologies were developed to address such requirements. However, the novel and challenging requirements of today's Communities can hardly be addressed by current Systems. Indeed, Communities are today interested in building digital knowledge that goes far beyond the mere electronic version of document, but focuses on compound objects of proprietary structure, which aggregate information objects of any media, residing at any location, inside or outside the Community boundaries.

This paper presented a reference architecture for Type-based Repository Systems, with aim to systematise the implementation of systems capable to address the novel community needs. The architecture accommodates into a comprehensive framework the components and the technologies needed to deal with rich knowledge and by promoting exploitation of aggregative and federative mechanisms to reuse and exchange knowledge across the boundaries of single Repositories. Future research activities will address the issues relative to the T-Repository Areas. Special interests will be dedicated to the design and implementation of the Logical and Physical Area, by formally designing the type system to be adopted and proceeding to a first implementation of the T-Repository.

## References


1. Lynch, C.A.: Institutional Repositories: Essential Infrastructure for Scholarship in the Digital Age. ARL **226** (2003) 1–7
2. Crow, R.: The Case for Institutional Repositories: A SPARC Position Paper. ARL Bimonthly Report **223** (2002)
3. Lyon, L.: eBank UK: Building the links between research data, scholarly communication and learning. Ariadne **36** (2003)
4. Ware, M.: Pathfinder research on web-based repositories. Technical report, Publisher and Library/Learning Solutions (2004)
5. Heery, R., Anderson, S.: Digital Repositories Review. Technical report, JISC (2005)
6. Peter Suber: Open Access Overview. `http://www.earlham.edu/~peters/fos/overview.htm` (2007) (accessed July 2007).
7. Lagoze, C., Van de Sompel, H.: The open archives initiative: building a low-barrier interoperability framework. In: Proceedings of the first ACM/IEEE-CS Joint Conference on Digital Libraries, ACM Press (2001) 54–62

8. Open Society Institute (OSI): A Guide to Institutional Repository Software (2004)
9. Pierce, B.C.: Types and Programming Languages. The MIT Press, Cambridge, Massachusetts, London, England (2002)
10. Lagoze, C., Payette, S., Shin, E., Wilper, C.: Fedora: An Architecture for Complex Objects and their Relationships. International Journal on Digital Libraries **6** (2005) 124 – 138
11. Tansley, R., Bass, M., Stuve, D., Branschofsky, M., Chudnov, D., McClellan, G., Smith, M.: The DSpace Institutional Digital Repository System: current functionality. In: ACM/IEEE 2003 Joint Conference on Digital Libraries (JCDL 2003), 27-31 May 2003, Houston, Texas, USA, Proceedings, IEEE Computer Society (2003) 87–97
12. Variuos Authors: The Guide for Electronic Theses and Dissertations. `http://www.etdguide.org` (accessed August 2007)
13. EGEE: gLite: Lightweight Middleware for Grid Computing. `http://glite.web.cern.ch/glite/` (accessed August 2007)
14. Trunfio, P., Talia, D., Papadakis, H., Fragopoulou, P., Mordacchini, M., Pennanen, M., Popov, K., Vlassov, V., Haridi, S.: Peer-to-Peer resource discovery in Grids: Models and systems. Future Gener. Comput. Syst. **23** (2007) 864–878
15. Open Archives Initiative: Object Reuse and Exchange (OAI-ORE). `http://www.openarchives.org/ore/` (2006)
16. MPEG-21, Information Technology, Multimedia Framework: Part 2: Digital Item Declaration. Technical report, ISO/IEC 21000-2:2003 (2003)
17. Millington, P., Nixon, W.J.: EPrints 3 Pre-Launch Briefing. Ariadne **50** (2007)
18. Castelli, D., Pagano, P.: A System for Building Expandable Digital Libraries. In: ACM/IEEE 2003 Joint Conference on Digital Libraries (JCDL 2003), 27-31 May 2003, Houston, Texas, USA, Proceedings, IEEE Computer Society (2003) 335–345
19. Van de Sompel, H., Bekaert, J., Liu, X., Balakireva, L., Schwander, T.: adore: A modular, standards-based digital object repository. The Compututer Journal **48** (2005) 514–535
20. Saidis, K., Pyrounakis, G., Nikolaidou, M., Delis, A.: Digital object prototypes: An effective realization of digital object types. In: Research and Advanced Technology for Digital Libraries, 10th European Conference, ECDL 2006, Alicante, Spain, September 17-22, 2006, Proceedings. (2006) 123–134
21. Gonçalves, M.A., France, R.K., Fox, E.A.: MARIAN: Flexible Interoperability for Federated Digital Libraries. In: Research and Advanced Technology for Digital Libraries, 5th European Conference, ECDL 2001, Darmstadt, Germany, September 4-8, 2001, Proceedings. Volume 2163 / 2001 of Lecture Notes in Computer Science., Springer-Verlag (2001) 173–186
22. Aloia, N., Concordia, C., Meghini, C.: Implementing BRICKS, a Digital Library Management System. In: Proceedings of the Fifteenth Italian Symposium on Advanced Database Systems, SEBD 2007, 17-20 June 2007, Torre Canne, Fasano, BR, Italy. (2007) 4–15
23. Rajasekar, A., Wan, M., Moore, R., Schroeder, W., Kremenek, G., Jagatheesan, A., Cowart, C., Zhu, B., Chen, S.Y., Olschanowsky, R.: Storage Resource Broker - Managing Distributed Data in a Grid. Computer Society of India Journal, Special Issue on SAN **33** (2003) 42–54
24. Candela, L., Castelli, D., Pagano, P., Simi, M.: From Heterogeneous Information Spaces to Virtual Documents. In Fox, E.A., Neuhold, E.J., Premsmit, P., Wuwongse, V., eds.: Digital Libraries: Implementing Strategies and Sharing Experiences, 8th International Conference on Asian Digital Libraries, ICADL 2005. Lecture Notes in Computer Science, Bangkok, Thailand, Springer (2005) 11–22
25. Candela, L., Castelli, D., Pagano, P., Simi, M.: OpenDLibG: Extending OpenDLib by exploiting a gLite Grid Infrastructure. In: Research and Advanced Technology for Digital Libraries, 10th European Conference, ECDL 2006, Alicante, Spain, September 17-22, 2006, Proceedings. Lecture Notes in Computer Science, Springer-Verlag (2006) 1–13