

DIF8914 Distributed Information Systems

A Framework for Building Open Digital Library

Ding Hao

IDI, Norwegian University of Science and Technology, 7491 Trondheim, NORWAY

haowing@idi.ntnu.no

Abstract

Digital Libraries (DLs) have traditionally been positioned at the intersection of library science, computer science, and networked information systems. The different underlying philosophies of these three fields have had an unsettling influence on the development of DLs. While library science is fairly mature, networked information systems are constantly evolving to keep pace with Internet innovation. DLs are thus expected to demonstrate the careful management of libraries while supporting standards that evolve at an astonishing pace. This architectural moving target is a predicament that all DLs face sooner or later in their lifecycle, and one that few manage to deal with effectively. To exacerbate this problem, there has been a general desire for systems to be interoperable at the levels of data exchange and service collaboration. Such interoperability requirements necessitated the development of standards such as the Dublin Core Metadata Element Set and the Open Archives Initiative's Protocol for Metadata Harvesting (OAI-PMH). These standards have achieved a degree of success in the DL community largely because of their generality and simplicity. Informed by those lessons, this essay is an attempt to introduce interoperability standards and survey the methods to form the basis of a framework for building extensible DLs.

1. Introduction

Digital libraries are far from well-defined [1] Definitional agreement may only extend to notions of accessible collections of information. Because of this, it is hardly surprising to note that the field does not easily converge on standards and technology. Most of the existing systems that are classified as DLs have resulted from custom-built software development projects -- each the product of intensive design, implementation and testing cycles. There are many reasons why this effort is repeated for each project:

- Many DLs are built in isolation as a response to the needs of a particular community, in most cases not involving personnel with prior experience.
- Most modern DLs have WWW interfaces -- thus the user interfaces and process flows are fashioned to resemble the way people use the WWW, which itself changes with time.
- Each DL is aimed at meeting the needs of a particular community -- so the underlying program logic varies vastly among systems.
- Most DLs are intended to be quick solutions to urgent community needs -- so not much

thought goes into planning for future redeployment of the systems.

- DLs, by the very nature of being responses to user needs, can be arbitrarily complex, so new projects sometimes choose to develop from scratch because it is cheaper than adapting what already exists to a different set of scenarios.
- As DL systems get more complex, extensibility becomes more difficult and, as a result, maintainability is compromised. As testimony to this, at the turn of the millennium, Dijkstra wrote that computing central challenge of "how not to make a mess of it" had not been met [2].
- There are very few software toolkits available to build DLs.

A natural solution would be to create software toolkits. A few institutions have investigated this approach. Dienst [3] is a DL system developed at Cornell University with tasks clearly divided and specified by a protocol based on HTTP and eventually using XML. It was developed to support distributed operation of the NCSTRL project and, while technically sound, required an investment in software, methodology, and support that some prospective users were not willing to make. The Repository-in-a-Box [4] software from the University of Tennessee is an alternative, as is the E-Prints software from Southampton University [5]. Both these toolkits avoid many problems related to complexity of DLs by defining workflows that are not easy to change. All of these and other systems have had varying degrees of success among archivists looking for drop-in solutions but they generally suffer from two basic problems:

- The range of possible workflows is restricted by the design of the system.
- The software is either built as a monolithic system or as components that communicate using non-standard protocols -- in both cases making understanding and modification a complex process.

Because it is widely accepted as good software engineering practice, most modern programming environments adopt some form of component model. Even in the DL community, as far back as 1994, early discussions on the future of DLs [6] concluded that components were an integral part of the solution. The University of Michigan Digital Library Project investigated using autonomous agents as the basic components of a DL [7]. Stanford's InfoBus project defined a set of services to support distributed digital libraries, each wrapped in an object, communicating through a remote method invocation interface [8, 9]. Other scientific communities embraced component technology as an aid to rapidly and correctly solving problems -- for example, the Sieve framework at Virginia Tech [10] encapsulates scientific functionality into software components. However, in spite of the widespread use of such technology, for the reasons outlined above, the DL community did not in general adopt a single component framework.

2. Approach to Open Digital Libraries

2.1 What is Open Digital Library?

In October of 1999 the Open Archives Initiative (OAI) [11] was launched in an attempt to address interoperability issues among the many existing and independent DLs. The focus was on high-level communication among systems and simplicity of protocols. The OAI has since received much media attention in the DL community and, primarily because of the simplicity of its standards, has attracted many early adopters.

The OAI Protocol for Metadata Harvesting [11], now the version 2 is available[11], in essence supports a system of interconnected components, where each component is a DL. Also, since the protocol is simple and is becoming widely accepted, it is far from being a custom solution of a single project. The OAI protocol can be thought of as the glue that binds together components of a larger DL. However, since DLs are themselves defined only loosely, this collaborative system could be composed of individual component DLs, and each with different functionality. In the extreme case, each component DL could supply the functionality of exactly one (part of a) service expected by a user. This is the approach taken in this work, where *Digital Libraries are modeled as networks of extended Open Archives, with each extended Open Archive being a source of data and/or a provider of services.* (The "extensions" are necessary since Open Archives are optimized for the provision of data -- but are generalizable to other tasks with a few minor changes.) This network of extended Open Archives, an instance of which is illustrated in Figure 1, is herein called an *Open Digital Library (ODL)*.

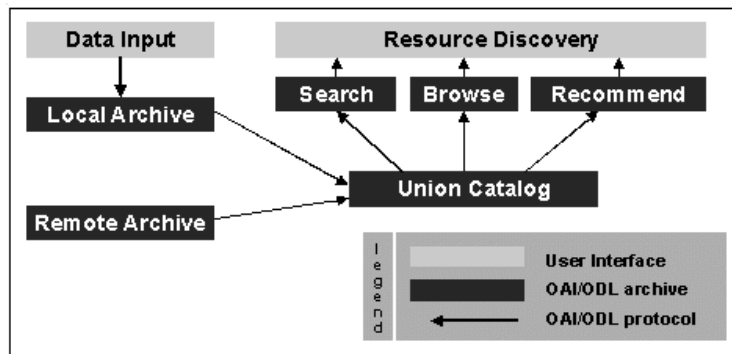


Figure 1. Example networked architecture of an Open Digital Library.

This approach to DL architecture is further motivated by the following factors:

- Componentization and standardization are built into the system by design if every service is delivered by an extended Open Archive. This inherently supports reuse and allows for interoperability at the level of individual services within the DL.
- The ODL approach closely resembles the way that physical libraries work. In a physical library the individual systems interoperate within their own communities. For example, the purchasing department interoperates with the booksellers and the inter-library loan department interoperates with peer departments at other libraries. A retiring head of the acquisitions department can be replaced by a peer at another institution because he or she understands a common protocol for all libraries. Interoperability is achieved at the level of individual services rather than at the level of organizations.
- There is currently a significant difference in technology between a research DL and a production DL. The former focuses on experimental concepts and technology while the latter deals with the real issues of meeting the needs of users. Connecting the two is not usually a simple task, but if both systems subscribe to a common protocol, that would greatly simplify matters -- OAI can be the basis for that protocol.
- The Internet is without a doubt the single most effective information dissemination tool of current times. This was primarily possible because of the simplicity of the protocols it relies on and the hierarchical manner in which protocols such as HTTP [13] build on more fundamental protocols such as TCP/IP. The OAI provides us with a simple protocol to transfer metadata; building simple layered extensions to this protocol would closely follow the proven methodology of the networking community [14].

- While complex system interactions might support complex operations, they also raise the bar on adoption of new technology. A good example would be the hypertext community where the WWW has succeeded well beyond other projects simply because its model was always a simple one [15]. Modeling DL services as Open Archives would enforce such a degree of simplicity.
- Scholarly communication is a rapidly changing field and many people are slow in making the transition to new forms of communication in spite of a growing number of advocates [16]. The success of new DL systems in this arena relies on keeping pace with current thinking on how publications are created, processed, and distributed. A simple component model will greatly simplify changes in the workflow of the DL to support the gradual shift to new and improved processes.
- User interface design and workflow management are complex tasks. But common base-level services -- mediators for connecting resources or middleware in three-tier client-server development [17] -- have emerged in practice, for example, supporting searching and browsing. If an arbitrarily complex user interface could access DL components in a standard manner, it would be easier to interchange components and add new services -- the OAI protocol could be the basis of that standard protocol for components.
- Norman advocates that designs should be visible, understandable and natural in their mappings [18]. The OAI protocol is already establishing itself in those areas so it makes an ideal foundation upon which to build.

2.2 Open Digital Library Design

2.2.1 Open Archives Initiative

ODLs are guided by a set of design principles and operationalized with the aid of a set of the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) extensions.

OAI-PMH provides an application-independent interoperability framework based on *metadata harvesting*. There are two classes of participants in the OAI-PMH framework:

- *Data Providers* administer systems that support the OAI-PMH as a means of exposing metadata; and
- *Service Providers* use metadata harvested via the OAI-PMH as a basis for building value-added services.

The main definitions and concepts in OAI-PMH are as follows:

(1) Harvester

A *harvester* is a client application that issues OAI-PMH requests. A harvester is operated by a service provider as a means of collecting metadata from repositories .

(2) Repository

A *repository* is a network accessible server that can process the 6 OAI-PMH requests. A repository is managed by a data provider to expose metadata to harvesters . To allow various repository

configurations, the OAI-PMH distinguishes between three distinct entities related to the metadata made accessible by the OAI-PMH.

resource - A resource is the object or "stuff" that metadata is "about". The nature of a resource, whether it is physical or digital, or whether it is stored in the repository or is a constituent of another database, is outside the scope of the OAI-PMH.

item - An item is a constituent of a repository from which metadata about a resource can be disseminated. That metadata may be disseminated on-the-fly from the associated resource, cross-walked from some canonical form, actually stored in the repository, etc.

record - A record is metadata in a specific metadata format. A record is returned as an XML-encoded byte stream in response to a protocol request to disseminate a specific metadata format from a constituent item.

(3) Item

An *item* is a constituent of a repository from which metadata about a resource can be disseminated. An item is conceptually a container that stores or dynamically generates metadata about a single resource in multiple formats, each of which can be harvested as records via the OAI-PMH. Each item has an identifier that is unique within the scope of the repository of which it is a constituent.

(4) Unique Identifier

A *unique identifier* unambiguously identifies an item within a repository; the unique identifier is used in OAI-PMH requests for extracting metadata from the item. Items may contain metadata in multiple formats. The unique identifier maps to the item and all possible records available from a single item share the same unique identifier.

The format of the unique identifier must correspond to that of the URI (Uniform Resource Identifier) syntax. Individual communities may develop community-specific URI schemes for coordinated use across repositories. The scheme component of the unique identifiers must not correspond to that of a recognized URI scheme unless the identifiers conform to that scheme.

Unique identifiers play two roles in the protocol:

- *Response*: Identifiers are returned by both the *ListIdentifiers* and *ListRecords* requests.
- *Request*: An identifier, in combination with a *metadataPrefix*, is used in the *GetRecord* request as a means of requesting a *record* in a specific metadata format from an item.

Note that the identifier described here is *not* that of a *resource*. The nature of a resource identifier is outside the scope of the OAI-PMH. To facilitate access to the resource associated with harvested metadata, repositories should use an element in metadata records to establish a linkage between the record (and the identifier of its item) and the identifier (URL, URN, DOI, etc.) of the associated resource. The mandatory Dublin Core format provides the identifier element that should be used for this purpose.

(5) Record

A record is metadata expressed in a single format. A record is returned in an XML-encoded byte stream in response to an OAI-PMH request for metadata from an item. A record is identified unambiguously by the combination of the unique identifier of the item from which the record is

available, the *metadataPrefix* identifying the metadata format of the record, and the *timestamp* of the record. The XML-encoding of records is organized into the following parts:

- *header* -- contains the unique identifier of the item and properties necessary for selective harvesting. The header consists of the following parts:
 - ◆ the *unique identifier* -- the unique identifier of an item in a repository;
 - ◆ the *timestamp* -- the date of creation, modification or deletion of the record for the purpose of selective harvesting.
 - ◆ zero or more *setSpec* elements -- the set membership of the item for the purpose of selective harvesting.
 - ◆ an optional *status* attribute with a value of "deleted" -- indicates the withdrawal of availability of the specified metadata format for the item, dependent on the repository support for deletions.
- *metadata* -- a single manifestation of the metadata from an item. The OAI-PMH supports items with multiple manifestations (formats) of metadata. At a minimum, repositories must be able to return records with metadata expressed in the *Dublin Core* format, without any qualification. Optionally, a repository may also disseminate other formats of metadata. The specific metadata format of the record to be disseminated is specified by means of an argument -- the *metadataPrefix* -- in the *GetRecord* or *ListRecords* request that produces the record. The *ListMetadataFormats* request returns the list of all metadata formats available from a repository, or for a specific item (which can be specified as an argument to the *ListMetadataFormats* request).
- *about* -- an optional and repeatable container to hold data about the metadata part of the record. The contents of an about container must conform to an XML Schema. Individual implementation communities may create XML Schema that define specific uses for the contents of about containers. Two common uses of about containers are:
 - ◆ *rights statements*: some repositories may find it desirable to attach terms of use to the metadata they make available through the OAI-PMH. No specific set of XML tags for rights expression is defined by OAI-PMH., but the about container is provided to allow for encapsulating community-defined rights tags.
 - ◆ *provenance statements*: One suggested use of the about container is to indicate the provenance of a metadata record, e.g. whether it has been harvested itself and if so from which repository, and when.

The following example shows an XML-encoding of a record and its components:

- the *header* part with:
 - ◆ a unique identifier of the item from which the record was disseminated;
 - ◆ the timestamp of the record, i.e. 2002-02-28;
 - ◆ two *setSpecs*, respectively *cs* and *math*, indicating that the item from which the record was disseminated belongs to two sets of the repository;

- the *metadata* part. This consists of a single root tag - in the example the tag `oai_dc:dc` - with the nested tags belonging to the corresponding metadata format -- in the example, Dublin Core elements such as `dc:title`. Note that the root tag within the metadata part includes a number of attributes that are common to all XML documents that use namespaces and schema validity:
 - ◆ *namespace declarations* -- the declarations of the namespaces used within the metadata part, each of which is prefixed with `xmlns` . Namespace declarations within the metadata part fall into two categories:
 - *metadata format specific namespace(s)* - every metadata part must include one or more `xmlns` prefixed attributes that define the correspondence between a metadata format prefix -- e.g. `dc` -- and the namespace URI (as defined by the XML namespace specification) of the respective metadata format. Some metadata formats employ tags from multiple namespaces, requiring multiple `xmlns` prefixed attributes -- in the example, there are declarations for both `oai_dc` and `dc`.
 - *xml schema namespace* - every metadata part must include the attribute `xmlns:xsi`, the value of which must always be the URI shown in the example, which is the namespace URI for XML schema.
 - ◆ *xsi:schemaLocation* -- the value of which is a URI, URL pair; the first is the namespace URI of the metadata that follows in this part, and the second is the URL of the XML schema for validation of the metadata that follows.
- one *about* part of the record which uses the *oai_provenance.xsd* schema, as a means to provide information regarding the origins of the metadata part of the record. Note that the root element within each about part has the same structure as the root element in the metadata part.

(6) Set

A *set* is an optional construct for grouping items for the purpose of *selective harvesting*. Repositories may organize items into sets. Set organization may be flat, i.e. a simple list, or hierarchical. Multiple hierarchies with distinct, independent top-level nodes are allowed. Hierarchical organization of sets is expressed in the syntax of the `setSpec` parameter as described below. When a repository defines a set organization it must include set membership information in the headers of items returned in response to the *ListIdentifiers* , *ListRecords* and *GetRecord* requests.

Each node in a set organization of a repository has:

- a `setSpec` -- a colon `[:]` separated list indicating the path from the root of the set hierarchy to the respective node.
- a `setName` -- a short human-readable string naming the set.
- a `setDescription` -- an optional and repeatable container that may hold community-specific XML-encoded data about the set.

(7) Selective Harvesting

Selective harvesting allows harvesters to limit harvest requests to portions of the metadata available from a repository. The OAI-PMH supports selective harvesting with two types of harvesting criteria

that may be combined in an OAI-PMH request: *datestamps* and *set membership*

2.2.2 Designing ODL

By analyzing some of the emerging aspects of Internet development, a set of basic design principles have been extracted to guide the construction of new protocols so that they are consistent with proven techniques in networked information systems. The observed factors that influence these include:

- simplicity of protocols
- openness of standards
- layering of semantics
- independence of components
- loose coupling of systems
- purposeful orthogonality
- and reuse wherever possible.

Based on these ideas, a generalization of the OAI protocol is possible so that it may be used for purposes that go beyond its original intention, namely to provide higher-level DL services. Formally, these principles are stated as follows:

- (1). All DL services should be encapsulated within components that are extensions of Open Archives.
- (2). All access to the DL services should be through their extended OAI interfaces.
- (3). The semantics of the OAI protocol should be extended or overloaded as allowed by the OAI protocol, but without contradicting the essential meaning.
- (4). All DL services should get access to other data sources using the extended OAI protocol.
- (5). Digital Libraries should be constructed as networks of extended Open Archives.

Each DL service is then designed as a self-contained component that communicates with other services using a protocol that is an extension of the OAI-PMH. A typical example of how this works in practice is illustrated in Figure 2, which shows a search engine's relationships to its data source and the interface that uses it as a service component.

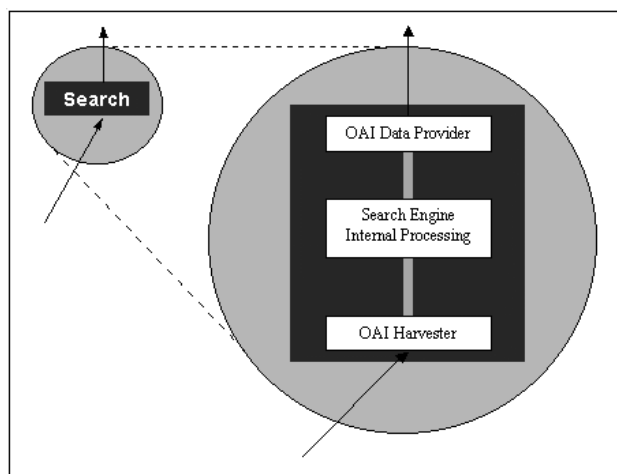


Figure 2. Internal structure of a typical ODL search component.

In this case, the OAI Harvester is used to obtain a stream of data which in turn is used to create indices for searching. Queries are then submitted through the OAI Data Provider interface. These queries overload the semantics of the OAI-PMH, by using the OAI notion of sets to correspond to the dynamically generated result sets of a search engine. Analogously, when such a request is submitted, the query is mapped to the name of a set. Thus, without making any changes, the OAI-PMH can be used to serve as the interface to a search engine. With a few minor additions to the OAI-PMH, information such as cardinality of result sets also can be returned.

An example of such an OAI request, with overloaded semantics for a search component is:

verb=ListIdentifiers&set=odlsearch1/computer%20science/1/10

This query specifies that the response should contain the *identifiers* of the first 10 most relevant documents with respect to the query "computer science". The response generated by the component is in the standard format returned by OAI-compliant archives.

To use this component in a typical ODL network, it must be connected to a source of metadata and a user interface, much in the same manner as UNIX pipes and filters are used to connect cooperating processes together. This is illustrated in Figure 3.

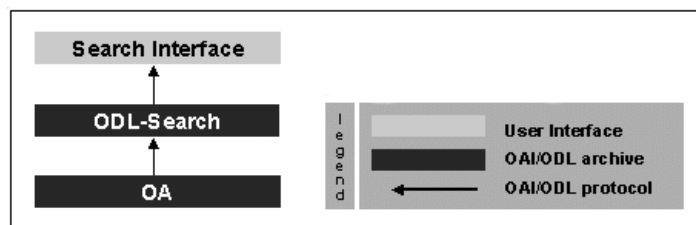


Figure 3. Simple ODL network using Search component.

In the practical development stage, it is mandatory to collect the source archives from which data was harvested through OAI-PMH interfaces, one of which required a filter because of minor differences in implementation from the others. The data was aggregated into a central archive for use by local services. Three high-level services were provided using this data: Search, Browse, and Recent. Search indexed the data and exposed an OAI-like interface for specifying keyword queries. Browse sorted the data and exposed a slightly different OAI-like interface for accessing items by controlled vocabulary

elements. Recent stored items and upon request returned a random sample of those. The main components and their functions are shown in Table 1.

Component	Function
ODL-Union	Combine metadata from multiple sources
ODL-Filter	Reformat metadata from (non-OAI-Conforming) data sources
ODL-Search	Provide search engine functionality
ODL-Browse	Provide category-driven browsing functionality
ODL-Recent	Provide a sample of recently-added items

Table1. Main Components in prototype system

The design method mentioned here for ODL system conformed to the design principles stated earlier and thus provide an extensible framework for future expansion. And furthermore, the future resulting system should still abide by the principles of software engineering as applied to distributed information systems, and the design and development of Digital Libraries in particular.

3. Conclusion and Future Work

It is hoped that the framework devised by component technology will change the way people build ODLs. Building upon a foundation of extensibility, ODLs then will be possible to work on providing more interesting services to users, thus bridging the wide gap between current research and production systems, and ultimately making information more accessible to more people.

In the future work, we will attempt to integrate our work with emerging standards in web-based services such as SOAP[19] and WSDL[20], which are expected to provide a general syntactic layer for high-level application protocols.

References

- [1]. Borgman, C. L. (1999). "What are digital libraries? Competing visions.", in *Information Processing and Management*, Vol 35, No 3, pp. 227-243.
- [2]. Dijkstra, Edsger. (2001). "The End of Computing Science", in *Communications of the ACM*, Vol 44, No 3, March 2001, p. 92.
- [3]. Lagoze, C. and J. R. Davis. (1995). "Dienst - An Architecture for Distributed Document Libraries", in *Communications of the ACM*, Vol 38, No 4, p. 47.
- [4]. NHSE. (2001). *Repository-in-a-Box*. Website <<http://www.nhse.org/RIB/>>.
- [5]. OpCit. (2001). *E-Prints*. Website <<http://www.eprints.org/>>.
- [6]. Gladney, H., Z. Ahmed, R. Ashany, N. J. Belkin, E. A. Fox, and M. Zemankova. (1994). *Digital Library: Gross Structure and Requirements (Report from a Workshop)*, IBM Almaden Research Center, Virginia Tech Dept. of Computer Science, June 1994.

- [7]. Birmingham, William P. (1995). "An Agent-Based Architecture for Digital Libraries", in *D-Lib Magazine*, July 1995. Available <<http://www.dlib.org/dlib/July95/07birmingham.html>>.
- [8]. Baldonado, Michelle, Chen-Chuan K. Chang, Luis Gravano, and Andreas Paepcke. (1997). "The Stanford Digital Library Metadata Architecture", in *International Journal on Digital Libraries*, Vol 1, No 2, pp. 108-121. Available. <<http://www-diglib.stanford.edu/cgi-bin/get/SIDL-WP-1996-0051>>.
- [9]. Roscheisen, M., M. Baldonado, C. Chang, L. Gravano, S. Ketchpel, and A. Paepcke. (1998). "The Stanford InfoBus and Its Service Layers: Augmenting the Internet with Higher-Level Information Management Protocols", in *Digital Libraries in Computer Science: The MeDoc Approach, Lecture Notes in Computer Science*, No. 1392, Springer, 8 August 1998. Available <<http://dbpubs.stanford.edu:8090/pub/1998-25>>.
- [10]. Sieve. (2001). Sieve. Website <<http://simon.cs.vt.edu/sieve/>>.
- [11]. Lagoze, Carl and Herbert Van de Sompel. (2002). *The Open Archives Initiative Protocol for Metadata Harvesting*, Open Archives Initiative, July 2002. Available <<http://www.openarchives.org/OAI/openarchivesprotocol.html>>
- [12]. Van de Sompel, Herbert and Carl Lagoze. (2000). "The Santa Fe Convention of the Open Archives Initiative" in *D-Lib Magazine*, Vol 6, No 2, February 2000. Available <<http://www.dlib.org/dlib/february00/vandesompel-oai/02vandesompel-oai.html>>.
- [13]. Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. (1999). *RFC2616: Hypertext Transfer Protocol - HTTP 1.1*, Network Working Group, June 1999. Available <<ftp://ftp.isi.edu/in-notes/rfc2616.txt>>.
- [14]. ISO. (1994). *ISO/IEC 7498-1:1994, Open Systems Interconnection Basic Reference Model: The Basic Model*, International Organization for Standardization.
- [15]. Berners-Lee, Tim and Mark Fischetti. (1999). *Weaving the Web*, Harper, San Francisco.
- [16]. Harnad, S. (1999). "Free at Last: The Future of Peer-Reviewed Journals", in *D-Lib Magazine*, Vol 5, No 12, December 1999. Available <<http://www.dlib.org/dlib/december99/12harnad.html>>.
- [17]. Umar, Amjad. (1997). *Object-Oriented Client/Server Internet Environments*, Prentice Hall, New Jersey.
- [18]. Norman, Donald. (1990). *The Design of Everyday Things*, Currency/Doubleday, New York.
- [19]. Simple Object Access Protocol (SOAP) 1.1, Available <<http://www.w3.org/TR/SOAP>>
- [20]. Web Services Description Language (WSDL) 1.1, Available <<http://www.w3.org/TR/wsdl>>

Additional resources

- [1]. <http://www.openarchives.org/>
- [2]. <http://www.dbulincore.org/documents/dces/>
- [3]. <http://www.eprints.org/>
- [4]. <http://www.dlib.org/>
- [5]. <http://www.dlib.org/dlib/september01/suleman/09suleman-pt1.html>
- [6]. <http://www.dlib.org/dlib/september01/suleman/09suleman-pt2.html>>