Developing a Technical Registry of OAI Data Providers

Thomas G. Habing, Timothy W. Cole, and William H. Mischo

University of Illinois at Urbana-Champaign
Grainger Engineering Library Information Center
1301 W. Springfield Ave., Urbana, Illinois 61801, USA
{thabing, t-cole3, w-mischo}@uiuc.edu

Abstract. With the continued growth of the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [1] it has become increasingly difficult for OAI service providers to discover new and keep up-to-date with existing data providers. There are currently several registries of OAI data providers. Most of these registries are incomplete. Most contain minimal information about registered providers – typically a base URL and little if anything else – providing service providers no clue as to repository scope, content, or size. These deficiencies mean significant extra overhead for service providers. This paper describes a more comprehensive registry of OAI data providers (available at http://oai.grainger.uiuc.edu/registry), developed to address some of these issues. While our registry as it presently exists facilitates discovery of data providers, utility is limited by lack of consistent practice for collection-level metadata. To realize the full potential of a better registry, the OAI community needs to develop better practices for collection-level description.

1 Why Another OAI Registry?

We developed our own OAI metadata provider registry to better support a range of OAI-based projects at the University of Illinois Libraries. These projects have included the Mellon funded UIUC Digital Gateway to Cultural Heritage Materials,[1] the Grainger Engineering Library's OAI Search Portal for Engineering, Computer Science, and Physics,[2] the IMLS Digital Collections and Content project,[3] the NSDL Second Generation Digital Mathematic Resources project,[4] and most recently the CIC-OAI Metadata Harvesting Service project.[5] Especially for those projects which are building focused OAI-based services, it has became clear that a significant amount of effort is required to discover relevant data providers and/or relevant sets within a single data provider. This has usually involved manually browsing data providers whose base URLs are listed in one of several existing registries such as that maintained at the Open Archives Initiative web site[6] or at the OAI Repository Explorer

[1] http://oai.grainger.uiuc.edu/
[2] http://g118.grainger.uiuc.edu/engroai/
[3] http://imlsdcc.grainger.uiuc.edu/
[4] http://nsdl.grainger.uiuc.edu/
[5] http://cicharvest.grainger.uiuc.edu/
[6] http://www.openarchives.org/Register/BrowseSites.pl

web site at Virginia Tech.[7] As we employed other discovery methods such as 'Googling,' it became clear that the existing registries were not complete, even taken together. For various reasons there seemed to be a large number of OAI data providers that were not registered with any of the existing lists.

At the same time, various other OAI researchers were reporting similar problems discovering and characterizing OAI repositories. At the JCDL 2003 conference, Kat Hagedorn, manager of the OAIster project,[8] suggested an OAI service that "classifies and tracks repositories." [2] In the conclusions of her report for The Digital Library Federation, Martha Brogan also comments on the lack of any comprehensive registry of data or service providers. [3] To address the issue, the Open Archives Forum (OAForum)[9] project developed a very useful registry of OAI resources, including data providers. However, because their registry was purposefully focused on European repositories and required self-registration its coverage was not as comprehensive as would have been desired. In addition, it lacked some of the functionality that we felt would be useful for a technical registry designed to support service providers in a more automated fashion.

Also, once our registry began to take shape and after the feedback we received following its announcement[10] in the OAI-implementers mailing list we realized that there was potential for many features beyond just discovery that a technical registry could address. This functionality will be described in the next section of this paper.

2 Features and Functionality

This section describes the key features and functions of the technical registry.

2.1 Comprehensive

First, we wanted the registry to be as complete as possible. This meant that we could not be limited to only the repositories listed in existing registries; we could not rely on self-registration, and we wanted the registry maintenance to be as automated as possible. Towards this end we developed several techniques for discovering and processing OAI data providers.

First, we identified all existing registries or lists of OAI data providers that were available. These included the Open Archives Initiative lists (including their list of rejected sites),[11] the list maintained by the Virginia Tech Repository Explorer, and several others.[12] These lists were presented in different formats from XML, to flat ASCII text files, to HTML pages. We developed automated means for processing all of these various lists.

[7] http://jingluo.dlib.vt.edu/~oai/cgi-bin/Explorer/2.0-1.45/testoai
[8] http://www.oaister.org/o/oaister/
[9] http://www.oaforum.org/index.php
[10] http://www.openarchives.org/pipermail/oai-implementers/2003-October/001017.html
[11] http://www.openarchives.org/OAI/RejectedSites
[12] See the http://oai.grainger.uiuc.edu/registry/#repoLists for a complete list

Second, we utilized various optional features of the OAI protocol itself. These include the friends description container, [4] which can be part of the OAI Identify response. The friends container lists other OAI data providers that may be confederates of the current data provider. We also utilized the provenance about container, [5] which can be part of an OAI record. The provenance container describes the origin OAI data provider for records that have been aggregated by another OAI data provider. Our processing scripts will automatically identify repositories from these sources and recursively process any newly identified data providers.

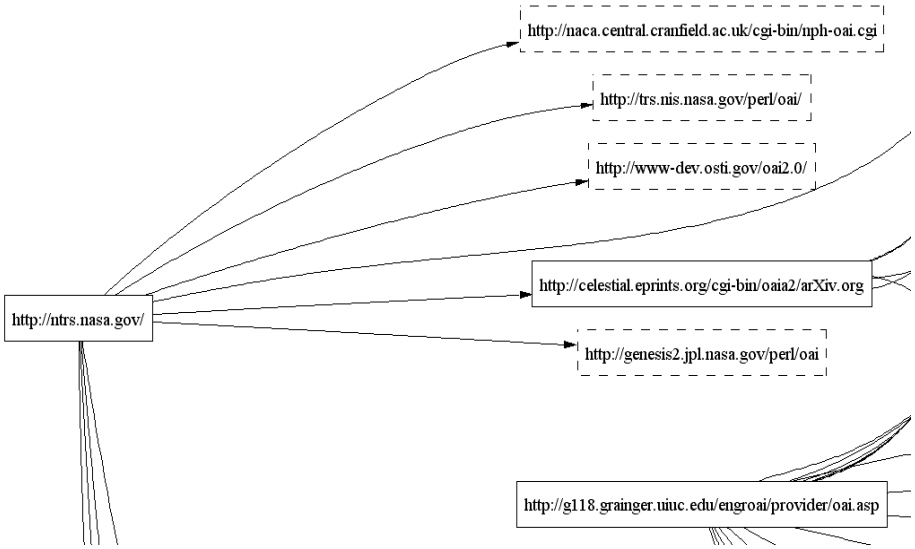

Fig. 1. This is a fragment of a graphic showing the relationships between various repositories which have been established either via the friends or provenance containers. The complete graphic is available from http://gita.grainger.uiuc.edu/registry/FriendsGraph.png.

Third, using the Google SOAP APIs, [6] we developed scripts to periodically search the Google web indexes to discover new OAI data providers. We are able to use various search strategies, from looking for web pages containing OAI related terms and then testing each URL contained in those web pages to determine whether the URL can respond to an OAI Identify request, to using advanced Google queries such as 'allinurl:verb=Identify' which only lists pages whose URLs contain the string 'verb=Identify.' Running these scripts approximately every month has identified numerous new data providers. All of the above techniques are executed on a periodic (currently irregular) basis to keep the registry as current as possible.

Finally, new repositories can be easily added one-at-time as they are identified manually. Numerous repositories have been added at the request of their administrators, and monitoring various mailing lists and web pages that focus on the Open Archives Initiative has identified other providers.

2.2 Searchable and Discoverable

The ability for service providers to discover relevant OAI data providers was another primary goal. However, we wanted to achieve this goal with as little manual cataloging of data providers as possible. To accomplish this we developed processes to automatically harvest and index various data from each data provider.

First for each repository we collect all responses from the Identify, ListSets, and ListMetadataFormats requests. These data are parsed and placed into a relational database (described later). In addition, the complete XML responses are indexed for full-text searching.

Second, for each combination of setSpec and metadataPrefix a minimal record harvest is performed to collect one or more sample records and, if possible, a record count. These data are also parsed into the relational database, and the sample XML records indexed for full-text search.

A key observation resulting from our search system is that repositories that include rich collection level metadata either in the optional Identify description containers or the optional ListSets setDescription containers will fare better in terms of discoverability. This suggests the desirability of broader use of collection-level metadata by the OAI community (discussed further below).

Once these data were all indexed, several methods of search and discovery were enabled. First, the full-text of all the OAI protocol responses, including the sample records, can be searched. Second, repositories can be browsed according to various different parameters, such as all repositories which support a specific metadata schema, all repositories which support a specific setSpec, all repositories originating from a particular top-level internet domain (such as a country domain), all repositories which support persistent deleted records, all repositories which are friends of a given repository, and so forth.[13]

In addition to searching and browsing, the registry also provides a detailed HTML view of each individual repository. This view includes, in a human readable format inspired by Jeff Young's work at OCLC with XSLT and OAI, [7] all of the collected responses from the data provider. Plus, the detailed view includes various other summarizations of the repository, such as the number of records that occur in each combination of setSpec and metadataPrefix and a link to a sample record for each combination. The view also lists other repositories that are related to this repository via the friends or provenance containers. The registry also tracks different protocol versions of the same repository and provides links between the newer and older versions.

2.3 Amenable to Machine Processing

In addition to making the registry accessible to human users, we wanted to expose its data in ways that were useful for machine processing.

[13] See the http://oai.grainger.uiuc.edu/registry/ for a complete list

The most obvious way to do this was to make the registry harvestable using the OAI-PMH itself.[14] The details of the implementation, especially the mapping of registry records to Dublin Core (DC) and the choice of OAI and DC Identifiers, was in large part driven by conversations with Jeff Young who needed an OAI registry data provider for his ERRoL system. [8] Essentially, the registry data provider exposes simple DC records (title and identifier) about each registered OAI data provider from one of two sets: ID or URL. The primary difference between these two sets is how the OAI Identifier for the registry record is derived. For the ID set, the OAI Identifier scheme of the repository, if any, is used to derive the OAI Identifier used for the registry record, for example: oai:id-registry.uiuc.edu:lcoa1.loc.gov. The OAI Identifier for the repository is either derived from the optional oai-identifier description container, [9] or it can be explicitly set by request in the registry. For the URL set, the repository baseURL is used to derive the OAI Identifier used for the registry record, for example: oai:url-registry.uiuc.edu:http://memory.loc.gov/cgi-bin/oai2_0.

The base URLs of all the repositories in the registry are also exposed using the standard friends description container, [4] or optionally in the same XML format as used by the Open Archives Initiative ListFriends.pl script.[15] Users also have the option of exporting any repository list that was generated from a search or browse operation in one of these two standard XML formats, either the friends description container or the ListFriends.pl format.

An RDF Site Summary (RSS) [10] feed of the most recently added or modified registry records is also available.[16] This feed can be used by various RSS news readers to automatically keep service providers notified when a repository is added or modified in the registry.

The most recent addition to the registry is an SRU (Search and Retrieve URL)[17] Service. [11] SRW/SRU builds on the Z39.50 protocol along with various web protocols, such as SOAP, HTTP, and XML, to define a service for searching databases across the web. Implementing this service for the OAI registry allows standard SRU clients to perform basic searches against the registry, returning the results as XML records. The service currently supports a subset of the CQL (Common Query Language),[18] allowing either the repository name, repository base URL, repository identifier, or the full text of various OAI responses to be searched. Records are returned using either Dublin Core or ZeeRex (The Explainable “Explain” Service).[19]

Finally, we are exploring methods for service providers to export a list of repositories that they would like to harvest, including information on the sets and metadata formats to harvest from each repository. We are currently calling this feature the Harvest Bag (analogous to the "book bag" feature of many online digital library services). It would allow a user to accumulate a list of base URLs, sets, and metadata formats that they would like to harvest. Once the list is complete they could export it in a new standard XML format which is being developed for the project. The ulti-

[14] http://oai.grainger.uiuc.edu/registry/px/oai.asp
[15] http://www.openarchives.org/Register/ListFriends.pl
[16] http://oai.grainger.uiuc.edu/registry/rss.asp
[17] http://oai.grainger.uiuc.edu/registry/sru/sru.asp
[18] http://www.loc.gov/z3950/agency/zing/cql/
[19] http://explain.z3950.org/

mate goal being that they could then import the XML file into their OAI harvest software so as to initiate or schedule harvests against the selected repositories.

3 Implementation Details

The registry was built on a Windows 2000 Server platform. The web server is Internet Information Server (IIS) using Active Server Pages (ASP). The database is Microsoft SQL Server. The registry maintenance programs were written in VBScript using a harvesting API that was developed for previous OAI projects at the UIUC Library. This API was implemented as an ActiveX dynamic link library (OAIHarvester.dll), which is freely available[20] from the SourceForge[21] open source repository.

3.1 Registry Maintenance Programs

There are two primary programs used for maintaining the registry. The first program is called RegistryPop. The main input for this program is either the base URL to a single repository to be added to the registry or the base URL of a list of repositories all of which are to be added. RegistryPop supports various different formats for these lists. As previously described, RegistryPop performs a minimal harvest of each repository, parsing the various XML responses, and inserting appropriate values into a relational database. RegistryPop is also responsible for recursively harvesting any other data providers that are identified from the friends or provenance containers. The script can also accept a number of optional parameters used for controlling its behavior, such as userid and password for harvesting password protected repositories, flags to indicate whether it should only add new repositories or whether it should only refresh the metadata of previously harvested repositories, or a flag to indicate whether it should ignore invalid repositories.

The second major program is called gOAIglePop. This program uses the Google SOAP API [6] to programmatically query the Google system to find potential OAI data providers. The main input for this program is one or more query phrases that are submitted to the Google search engine. Typical queries would be phrases that would be indicative of an OAI related web site, such as 'OAI,' 'OAI-PMH,' or 'Open Archives Initiative.' The most successful search has been to look for the string 'verb=Identify' directly in the URL using the Google 'allinurl:' special query term. The results of the Google queries are then parsed, and each URL is tested to determine whether it can respond to an OAI Identify request. URLs that lead to an OAI data provider are added to the output list. This output list can then be used as the input for the RegistryPop program.

In addition to the two main programs, there are a number of smaller programs and scripts, some implemented as SQL stored procedures, to perform various other maintenance tasks such as to identify only new repositories given a list of base URLs, extract namespace URIs or xml:lang attributes from XML files, or to add a list of

[20] http://sourceforge.net/project/showfiles.php?group_id=47963&package_id=46165
[21] http://sourceforge.net/

repositories to a virtual collection. All of these scripts are intended to be run from the command line and can be scheduled to run in an automated fashion using the system task scheduler. Most of the programs and scripts also output an activity log using a uniform XML format.

3.2 Registry Database

Data about repositories is stored in a relational database, Figure 2. This database consists of the primary Repositories table along with several tables that are related to this table via primary-foreign key relations. The Repositories table contains data about the OAI data providers, such when they were last harvested, links to previous versions of the same provider, HTTP headers returned by the provider web server, and the full-text of the Identify response.

The MetadataFormats and Sets tables contain the parsed responses to the ListMetadataFormats and ListSets requests to a repository. The Sets table will also contain the full-text of any optional setDescription containers.

The RecordCounts table contains the results of a minimal harvest of the repository. Using every combination of set and metadataPrefix a small harvest is performed, and the results stored in this table, including a count of records, a sample OAI identifier, a sample resumptionToken, and the full-text of a sample record. We are calling this a minimal harvest because only the first resumptionToken's worth of data is retrieved for processing. This means that a record count is only obtained if the repository supports the optional completeListSize attribute on the resumptionToken element or if there is only one resumptionToken's worth of data for the given request.

The Friends table identifies all the repositories that are related to a given repository. The type of the relation is either via the friends container [4] or the provenance container. [5]

There are a number of other tables that contain derived data to support specific queries. The Namespaces table contains a list of every namespace URI that is used by a given repository. The XMLLangs table lists all the xml:lang attribute values that are associated with a given repository. The RepoIdentifiers table lists possible OAI Identifiers that are being used by the repository, either pulled from the optional oai-identifier container [9] or by parsing sample OAI item identifiers.

The HarvestBag table allows a user to develop a custom list of OAI repositories that she would like to harvest, including which sets. This list can then be exported as an XML file.

The VirtualCollections and RepositoryCollections tables provide a means for creating virtual collections of related repositories. Currently, the only data that are maintained are the names of the virtual collections, such as 'DSpace Collections' or 'Eprints.org Collections.' These data are currently maintained manually, and are mostly a convenience for harvesters who would like to identify all the repositories that fall into a certain category. However, these tables will allow us to attach more diverse collection level metadata to OAI repositories or collections of repositories in the future.

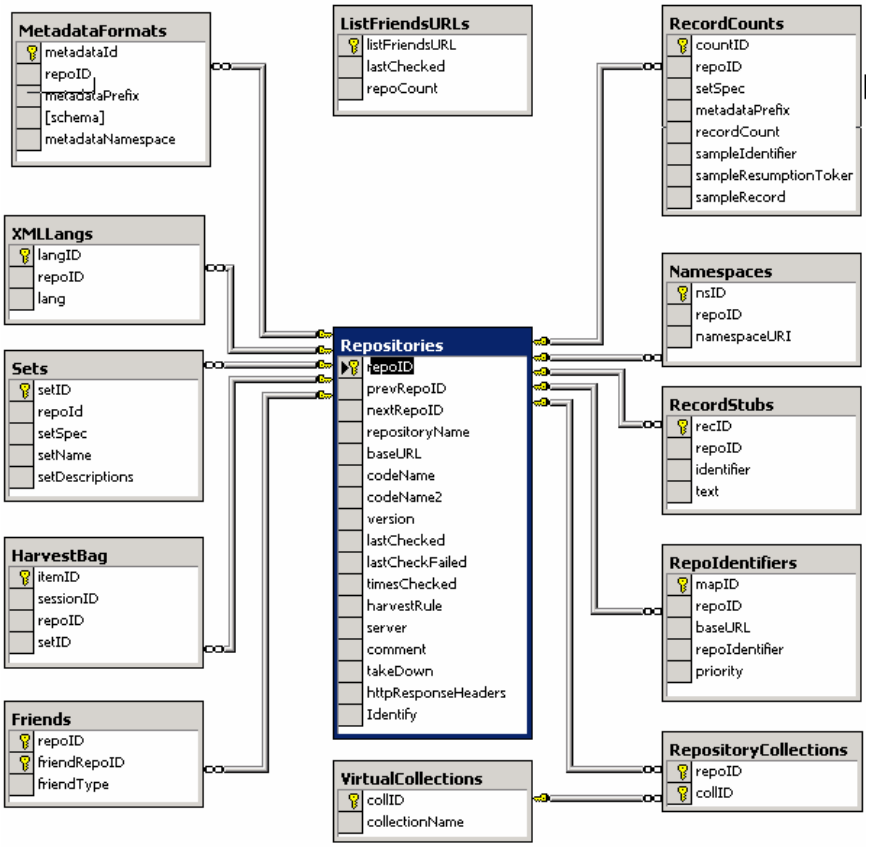

Fig. 2. This is the design of the relation database used for the registry

As already mentioned, various OAI protocol responses are indexed for full-text search. These include the Identify response, the ListSets response, and the sample records for every combination of set and metadataPrefix. The full-text search is implemented using the full-text capabilities of Microsoft SQL Server. [12] The search algorithm currently used is that all words or phrases (strings enclosed in quotes) must be found in the records full-text index to generate a hit. Right-hand truncation of search terms is also enabled by default. We may implement more advanced search capabilities in the future – for example, limiting a search to a particular index, such as sample records or setDescriptions.

3.3 Registry Web Interface

The web interface used for the registry is implemented as an Active Server Page (ASP) written in VBScript running on the Microsoft IIS web server. This includes the pages intended for use by humans, plus the Registry OAI data provider and RSS feeds intended for machine consumption. The top-level page and all of the various reports

are implemented as separate ASP scripts (currently 51 separate scripts). However, there are multiple common functions which are implemented in common include files. Extensive use is made of XSLT for transforming the OAI XML responses into HTML for display. Cascading Style Sheets (CSS) are also used extensively for customizing the display output.

4 Future Plans and Collection-Level Description

4.1 Registry Enhancements

While the registry is now fully operational, there remain a number of improvements we would like to make to increase its usefulness. Most of these are derived from our own internal use of the registry and the desire to decrease the manual workload associated with its maintenance. Other enhancements have been derived from conversations with the registry's users, such as Kat Hagedorn at Michigan who is using the registry to discover new repositories for her OAIster service or Jeff Young at OCLC who is using the registry for his ERRoL service. [8] Following, in no order, are some plans for future enhancements to the registry:

- We would like to provide more automated maintenance of the registry, including the ability of OAI data providers to securely add or modify their repository's records in the registry, including collection-level descriptive data.
- We would also like to improve the automated discovery of new repositories, such as automatically running the gOAIglePop script.
- We would like the ability to delegate the creation and maintenance of virtual collections of repositories, including collection-level metadata.
- We would like to improve the view of search results, especially the context of the search hit. The current system does not identify the context of a search hit, which could be the Identify or ListSets responses or the sample records.
- We would like to improve the ability of service providers to generate custom lists of repositories that can be added to their HarvestBag, exported, and then used to create harvesting schedules.

We are eager to make this registry a valuable resource to the OAI community, and we are very open to new ideas, suggestions, or collaborations. If you are interested please contact the authors.

4.2 Collection-Level Description in OAI

Our registry is intended especially to facilitate discovery and exploitation of OAI-compliant metadata providers. In this regard the utility of the registry is dependent in large part on the quality and extent of information about repositories and sets made available by providers. Beyond the largely technical enhancements described above, the usefulness of the registry could be further enhanced through the inclusion of richer collection-level description of provider repositories and sets. Current practice varies,

but in general implementers provide at most sparse descriptive information about the topic and scope of content described by their OAI metadata sets. Early on, when the number of metadata providers was small, machine-accessible collection-level description was not emphasized and arguably not needed. More recently however, with over 400 OAI metadata providers up and active, with many more in development, with larger providers becoming more common and making available increasingly diverse content, there is evidence of a renewed interest in collection-level description within the OAI community..

The need for collection-level description was considered by the OAI Technical Committee in developing the 2.0 version of the protocol. This led to the addition in the 2.0 version of the protocol of the optional setDescription container within the ListSets response. [1] The use of setDescription (and also the already existing description element within the Identify response) for collection-level description is explicitly encouraged in section 4.2 of the Implementation Guidelines for Repository Implementers. [13] The interest in collection-level description within the OAI community corresponds to a period of significant progress and interest in collection-level description within the digital library community as a whole. The initial work described here on a more sophisticated OAI metadata provider registry service suggests an opportunity for the OAI community to exploit emerging collection-level description best practices, both to facilitate the discovery and use of data providers and to provide context for item-level metadata records delivered.

Much of the current work in collection-level description has roots in Michael Heaney's "An Analytical Model of Collections and Their Catalogues."[22] This model was used as a foundation for the RSLP collection description schema,[23] indirectly for work now being done by the DC Collection Description Working Group,[24] and by other projects, such as our own project to create a registry for collections developed by or associated with National Leadership Grant projects funded by the U.S. Institute of Museum and Library Services.[25] In the Heaney model, and in associated schemas derived in large part from this model, core collection description is distinguished from description of related entities and agents, such as the 'collector' who assembled the collection, the 'owner' who holds rights over collection content, the 'location' of the collection, and the 'administrator' of the collection who administers it in its location.

A few of these collection-related entities, for instance the repository administrator, are already explicitly mentioned within the OAI protocol. Not specifically defined in OAI, however, are the core collection-level descriptive attributes identified in Heaney's model, and in other emerging models of digital content description. While there remain important issues with these models, and several of the formal XML schemas for expressing collection-level descriptive properties are still under development, there does seem to be an emerging consensus within the broader community about the identity and meaning of the most important, core collection-level description properties and attributes. This work is of a maturity suitable for broader use and experimentation.

[22] http://www.ukoln.ac.uk/metadata/rslp/model/amcc-v31.pdf
[23] http://www.ukoln.ac.uk/metadata/rslp/schema/
[24] http://dublincore.org/groups/collections/
[25] http://imlsdcc.grainger.uiuc.edu/CDschema_overview.htm

In our opinion it is time for the OAI community to develop initial consensus as to best practices for collection-level description in the context of OAI. This implies also a reconsideration of best practices for how OAI sets are implemented. Note, better collection-level descriptive practices and better use of OAI sets can be implemented effectively without changes to the base OAI protocols. Sufficient flexibility and capacity was included in the protocols by design. Arguably the future work that will lead to the greatest improvements in the utility of ours or any other OAI metadata provider registry will be work that addresses the need for richer collection-level descriptive information about OAI repositories and sets.

References

1. Lagoze, Carl, Van de Sompel, Herbert, Nelson, Michael, and Warner, Simeon: The Open Archives Initiative Protocol for Metadata Harvesting - Version 2.0. (2002) http://www.openarchives.org/OAI/openarchivesprotocol.html
2. Warner, Simeon and Nelson, Michael: Report on the Metadata Harvesting Workshop at JCDL 2003. SIGIR Forum, 37 (2), (Fall 2003) http://www.acm.org/sigir/forum/2003F/jcdl03_warner.pdf
3. Brogan, Martha L.: A Survey of Digital Library Aggregation Services. The Digital Library Federation Council on Library and Information Resources, Washington, DC (2003) http://www.diglib.org/pubs/brogan/brogan2003.htm
4. Lagoze, Carl, Van de Sompel, Herbert, Nelson, Michael, and Warner, Simeon: Implementation Guidelines for the Open Archives Initiative Protocol for Metadata Harvesting - XML Schema for repositories to list confederate repositories. (2002) http://www.openarchives.org/OAI/2.0/guidelines-friends.htm
5. Lagoze, Carl, Van de Sompel, Herbert, Nelson, Michael, and Warner, Simeon: Implementation Guidelines for the Open Archives Initiative Protocol for Metadata Harvesting - XML schema to hold provenance information in the "about" part of a record. (2002) http://www.openarchives.org/OAI/2.0/guidelines-provenance.htm
6. Google Web APIs. Google (2003) http://www.google.com/apis/index.html
7. Van de Sompel, Herbert, Young, Jeffery, A., and Hickey, Thomas, B.: Using the OAI-PMH ... Differently. D-Lib Magazine, 9, (7/8), (July/August 2003) http://www.dlib.org/dlib/july03/young/07young.html
8. Young, Jeffery, A.: Extensible Repository Resource Locators (ERRoLs) for OAI Identifiers. OCLC Online Computer Library Center (2004) http://www.oclc.org/research/projects/oairesolver/default.htm
9. Lagoze, Carl, Van de Sompel, Herbert, Nelson, Michael, and Warner, Simeon: Implementation Guidelines for the Open Archives Initiative Protocol for Metadata Harvesting - Specification and XML Schema for the OAI Identifier Format. (2002) http://www.openarchives.org/OAI/2.0/guidelines-oai-identifier.htm
10. RSS-DEV Working Group: RDF Site Summary (RSS) 1.0. http://web.resource.org/rss/1.0/spec
11. Denenberg, Ray (ed.): SRW/SRU Version 1.1. The Library of Congress, Washington, DC (2004) http://www.loc.gov/z3950/agency/zing/srw/
12. Cencini, Andrew B.: Building Search Applications for the Web Using Microsoft SQL Server 2000 Full-Text Search. Microsoft Corporation. (2002) http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql2k/html/sql_fulltextsearch.asp

13. Lagoze, Carl, Van de Sompel, Herbert, Nelson, Michael, and Warner, Simeon: Implementation Guidelines for the Open Archives Initiative Protocol for Metadata Harvesting – Guidelines for Repository Implementers. (2002)
http://www.openarchives.org/OAI/2.0/guidelines-repository.htm