ABSTRACT

TOWARDS DIGITAL LIBRARY SERVICE INTEGRATION

by
Prateek Shrivastava

Digital Library Service Integration (DLSI) aims to provide a systematic approach in integrating the services and collections of National Science and Digital Library. The National Science and Digital Library collections can share the services among themselves in a totally integrated environment. Collections as such will require no change to plug into the DLSI architecture. Collections will keep using the services of NSDL in the similar manner as before. These services will in turn pass few parameters to the services of DLSI. With the help of these parameters, wrappers will fetch the details and priority of the users. These wrappers will be using the services of Search and Discovery module, Metadata Management services, and Access Management services. Users will see a totally integrated environment. They will see their digital library system just as before. In addition to that, they will find some extra link anchors on the document. These links serve to provide the supplemental information or arrange the information in the user preferred way. For this matter, the DLSI maintains basic user's information and preferences. Other contributions include incorporating collaborative filtering for customizing large sets of links, and advance lexical analysis tool to identify the objects of interest in a document.

TOWARDS DIGITAL LIBRARY SERVICE INTEGRATION

by
Prateek Shrivastava

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer and Information Science

May 2003

APPROVAL PAGE

TOWARDS DIGITAL LIBRARY SERVICE INTEGRATION

Prateek Shrivastava

5/7/2003

Dr. Michael Bieber, Thesis Advisor — Date
Associate Professor, Information Systems Department.
College of Computing Science, NJIT

5/7/2003

Dr. Il Im, Committee Member — Date
Assistant Professor, Information Systems Department.
College of Computing Science, NJIT

5/07/2003

Dr. Y. F Brook Wu, Committee Member — Date
Assistant Professor, Information Systems Department.
College of Computing Science, NJIT

BIOGRAPHICAL SKETCH

Author:	Prateek Shrivastava
Degree:	Master of Science
Date:	May 2003
Date of Birth:	
Place of Birth:	

Undergraduate and Graduate Education:

- Master of Science in Computer and Science,
  New Jersey Institute of Technology, Newark, NJ, 2003
  GPA: 3.91

- Bachelor of Engineering in Electrical and Electronics,
  Karnataka Regional Engineering College, Surathkal.
  Mangalore University, India, 1999
  GPA: 3.5

Major: Computer Science

I dedicate this thesis to my Dad, Mom, Brother, and Sister-in-Law

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to Dr. Michael Bieber, who not only served as my research supervisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance. Special thanks are given to Dr. Il Im, and Dr. Yi-fang. Brook Wu for actively participating on my committee.

I also gratefully acknowledge support from National Science Digital Library for giving us this opportunity, and National Science Foundation for funding the research project of which this thesis is a part.

I would also like to thank Nkechi Nnadi, Xin Chen, Himanshu Shah, and Dong-ho Kim for their special support to the project.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of this thesis is to develop a service level infrastructure for digital libraries for integrating collections and services, and automatically generating links among related elements within them. When the DLSI service is plugged in with NSDL, collection providers just need to provide their content to the NSDL repository. On call, the DLSI will issue some commands through the wrappers to the services or collections, which will return the output in a certain format. The DLSI also automatically fetches the data from the repository and seamlessly integrate it with other collections and services. The developers at the collection provider's end will be responsible for managing the data such as adding new data to the collection, purging irrelevant data, and modifying the existing data. The DLSI will always fetch the most recent records from the collection using date stamp and additional parameters. These developers can also plug in additional services such as user-declared hypermedia links and guided tours referencing any repository element of their choice.

This infrastructure is called as DSLI (Digital Library Service Integration) ([Bieber et. al]). The DLSI when formally launched can form the core of a vibrant virtual educational system by supporting a broad range of services. It will be like "One Library, Many Portals". As a further clarification, it means that over a period of time DLSI will fetch the data from different collections all over the world and present it to the user in a single document. Thus for users, it is one library that they are accessing, but the contents come from a variety of sources. The DLSI as a service also uses other NSDL services. Three services that DLSI will be using are: Search and Discovery module, Access management service, and Metadata management service.

1.2 Background Information

1.2.1 History of NSDL

([http://nsdl.org/tag.4b5181e81f2bd62a.render.userLayoutRootNode.uP?uP_root=root&uP_sparam=activeTab&activeTab=7]):

National Science Digital Library officially launched its portal www.nsdl.org on December 3, 2002. Initial Development of National Science Digital Library (NSDL) program began in late 1995 with an internal concept paper for the National Science Foundation (NSF) Division of Undergraduate Education. In 1996, NSF released a report about ways to improve undergraduate science, technology, engineering, and mathematics (STEM) education. It recommended establishing a national digital library that would constitute an online network of learning environments for improving teaching and learning for STEM education at all levels. Beginning in 1998, two rounds of prototype projects were supported through special initiatives conducted under the auspices of the multi-agency Digital Libraries Initiative - Phase 2 (DLI-2) program. The NSDL program held its first formal funding cycle during fiscal year 2000.

To date, 119 projects have been funded to create collections and services for teacher and learners at all levels, and perform targeted research in digital libraries and their application to education. The NSDL program is an unusual program for NSF in that its projects are engaged in building an enterprise much larger than the object of any one grant. Most of these projects focus on collection and service development, while the Core Integration project is providing organizational and managerial functions of the digital library. The NSDL is emerging as a center of innovation in digital library as applied to education, and community center for groups focused on digital-library-enabled science education. Mission of NSDL is to build a comprehensive online source for science, technology, engineering, and mathematics education.

The National Science Foundation predicts that over the time National Science Digital Library will become the world's largest library of science, technology, engineering, and mathematics information resources and services, as well as an online network of learning environments and resources.

1.2.1 History of Digital Library Service Integration:

The project started with building of Dynamic Hypermedia Engine in the Collaborative and Hypermedia laboratory at NJIT. The third version of Dynamic Hypermedia Engine was released during the NSDL initial launch conference in December 2002 at Washington D.C. The concept of DLSI started even before this engine was launched. The concept of wrappers, which form an integral part of the DLSI project, was initiated in Fall 2002. This wrapper forms the core of all the utilities provided by the DLSI. The DHE is another integral part of the DLSI. DHE renders the anchors, also known as links, to the web page on World Wide Web on the fly. It is also capable of generating other hypermedia services within the application if basic material is provided by the resource owner.

CHAPTER 2

DIGITAL LIBRARY SERVICE INTEGRATION

2.1 Core Concepts

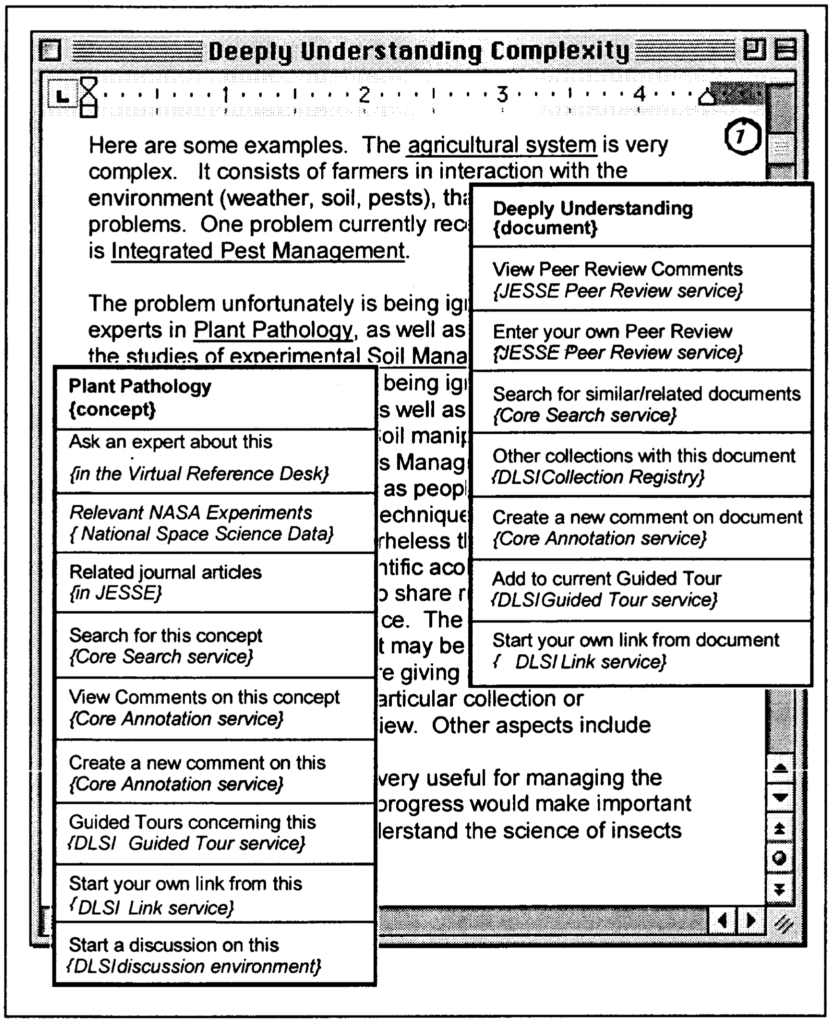
The entire functionality of Digital Library Service Integration revolves around four modules. They are:

1. Dynamic Hypermedia Engine
2. Core Integration Wrapper
3. Lexical Analysis Tool
4. Collaborative Filtering

This chapter will describe how users will interact with DLSI environment. It will also explain how and where each of the modules is used in the user's navigation as he/she continues through the journey within the DLSI environment. It will also describe the DLSI infrastructure as a whole. It also explains the use of text analysis both to supplement the links generated based on the system structure, and to supplement search services. It also describes how collaborative filtering will customize the set of links returned for each user. The users will interact with the DLSI in the same way as they were previously interacting with their digital library before. It's just they need to launch their digital library resource through DLSI system which as of now is this interface (http://hynic.njit.edu/). Through this system, they'll see additional link anchors in their page. When they click on one of these supplemental anchors, users are presented with a list of relevant links.

Figure 2.1 shows a mockup of a digital library document, superimposing two possible sets of links for different elements: the concept "Plant Pathology" and the document as a whole. If the user clicks on the anchor DLSI added to "Plant Pathology," DLSI would generate corresponding list of links with direct access to nine functions in different collections and services relevant for this concept. If the user clicks on the document information icon (in the top right-hand corner), DLSI would generate the corresponding list of links to seven functions relevant for this document. The DLSI also would filter and rank order the set of links generated to a specific user's preferences and current task.

As Figure 2.1 illustrates, DLSI automatically generates link anchors over elements of the display screen, for which relevant services apply. For example, if a user had created a comment about a document element or the element is the subject of a guided tour, DLSI would place a link anchor over that element. Selecting the anchor would display a list of relevant links DLSI has found for that element, including the link to the comment or tour. The system marks entire screens and documents as a link anchor if services apply to these as a whole. (Figure 2.1 illustrates this with an icon in the top right-hand side of the screen.) For example, if the screen is part of a guided tour, or a comment or discussion thread refers to it, then selecting the icon would lead to a list of links leading to these.



Deeply Understanding Complexity

Here are some examples. The agricultural system is very
complex. It consists of farmers in interaction with the
environment (weather, soil, pests), th
problems. One problem currently rec
is Integrated Pest Management.

The problem unfortunately is being ig
experts in Plant Pathology, as well as
the studies of experimental Soil Mana

being ig
s well as
oil mani
s Manag
as peopl
echnique
rheless t
ntific aco
share r
ce. The
t may be
e giving
articular collection or
iew. Other aspects include

very useful for managing the
progress would make important
lerstand the science of insects

Plant Pathology {concept}
Ask an expert about this *{in the Virtual Reference Desk}*
Relevant NASA Experiments { National Space Science Data}
Related journal articles *{in JESSE}*
Search for this concept *{Core Search service}*
View Comments on this concept *{Core Annotation service}*
Create a new comment on this *{Core Annotation service}*
Guided Tours concerning this *{DLSI Guided Tour service}*
Start your own link from this *{DLSI Link service}*
Start a discussion on this *{DLSI discussion environment}*

Deeply Understanding {document}
View Peer Review Comments *{JESSE Peer Review service}*
Enter your own Peer Review *{JESSE Peer Review service}*
Search for similar/related documents *{Core Search service}*
Other collections with this document *{DLSI Collection Registry}*
Create a new comment on document *{Core Annotation service}*
Add to current Guided Tour *{DLSI Guided Tour service}*
Start your own link from document *{ DLSI Link service}*

Figure 2.1 A mockup of a digital library document with DLSI support.

Description: DLSI would automatically add link anchors, including an icon in the top right-hand corner for the document as a whole. Choosing one prompts DLSI to generate a list of links. The figure superimposes two possible sets of links for different elements: the concept "Plant Pathology" and the document as a whole. Each link shows a display label describing the link, and the service or collection it leads to. Most services and collections now are part of the NSF's National Science Digital Library system, and will eventually be integrated through DLSI.

At this stage, it is also important to know that DLSI generates link anchors in two ways. Initially, HTML, Word, or PDF are parsed through the wrapper. Wrapper on the understanding of the structure of the document and its contents uses relationship analysis to mark the tags of the user's interest and puts a link on it. Most of the anchors should be identified in this manner, as this is the fastest way to parse a screen. This is the primary method used by the DLSI. More details on this are provided in the 'Design and Implementation' chapter.

2.1.1 Lexical Analysis

Another way to generate the link anchors is the use of lexical analysis. Lexical Analysis determines "Element of Interest" using Noun Phrase Extractor. It then supplements this element with the anchor tag. The DLSI wrappers perform another minor lexical analysis when they parse documents and display screens to determine additional "elements of interest", which the Integration Manager will supplement with DLSI link anchors. Noun Phrase Extractor works this way: Tokenization is first performed on the document or display screen. It then uses the WordNet lexical database [http://www.cogsci.princeton.edu/~wn/] to assign part-of-speech tags to tokens. Finally, a morphological and syntactic rule base is used to parse sentences and extract noun phrases. The Noun Phrase Extractor extracts noun phrases in their root forms (this takes care of morphological changes) from returned documents. These root form noun phrases are then separated into two lists of phrases: those that are in the master thesaurus file and those that are not. Any found in the master thesaurus will be made into supplemental link anchors. All noun phrases found will be used for automatic concept hierarchy generation.

Glossaries and thesauri developed by domain experts are highly useful in identifying meaningful keywords in documents. So far, Lexical Analysis group has collected thesaurus entries from Medical Subject Headings (MeSH, available at National Library of Medicine [http://www.nlm.nih.gov/mesh/download_mesh.html]), and ASIS Thesaurus of Information Science [http://www.asis.org/Publications/Thesaurus /isframe.htm]. The project team plans to collect many more in the future. These entries are combined to form an integrated master file supplied to the Noun Phrase Extractor when searching for noun phrases. Keywords and key phrases from participating collections and services also will be added to this integrated master file.

Once, the “Elements of Interest” are fetched by parsing the page through wrapper and by doing Lexical Analysis, it needs to sort the links out in preference with the user. This is important, as parsing the page and Lexical Analysis could return back hundreds of results. Therefore, the need for Filtering and Rank Ordering the links arise.

2.1.2 Filtering and Rank Ordering

The number of potential links that DLSI could generate for a particular element on a screen could vary from several to well over a hundred, resulting in the well-known hypermedia problem of cognitive overload (Conklin 1987; Halasz 1988; Thüring et al. 1995). With a large number of links, filtering and ordering them is critical for effective use. Filtering and rank ordering in DLSI poses several challenges. First, it should be customized to each user’s needs. Second, it should dynamically re-organize as the users advance through the system. Third, for the same user, support for multiple needs must be possible. A user may have several different tasks or needs and the links should be re-organized depending on the user's current task.

The DLSI incorporates collaborative filtering to filter information based on people's evaluations or behaviors. It generates recommendations using the following algorithm (Kostan et al., 1997; Herlocker et al., 1999; Im & Hars, 2001):

Step 1: Calculate degree of similarity ("similarity index") between the current user and other users.

Step 2: Identify a group of people ("reference group") who appear to share common interests with the current user. Their evaluations (or clickstreams) will be used for generating recommendations for the current user.

Step 3: Calculate estimated evaluations for items that the current user has not seen (or evaluated). An estimated evaluation predicts the current user's evaluation on an item.

Step 4: Rank order the items according to the estimated evaluations and select the top *n* items to recommend to the current user.

Collaborative filtering can customize the link set for each user. Two different data in DLSI can be used for Collaborative Filtering – users' direct evaluations (user-entered ratings) and indirect evaluations (click vs. non-click) on links generated by DLSI. Users' direct evaluation is continuous data and indirect evaluation is binary data – visit (click) vs. non-visit (non-click). Special Collaborative Filtering algorithms for users' click stream data can predict the probability of a user's next click (Breese et al., 1998; Weiss & Indurkhya, 2001). This algorithm can be applied to DLSI to rank order the links based on the "probability of click" (predicted relevance) of each link in a link set. The probabilities are calculated based on the past behaviors of other users. Users' click data can be collected automatically while users' direct evaluations need to be entered by users. In this research, users' click data will be used as the primary data to filter the links generated by DLSI. Users' direct evaluations will supplement this (Freund et al., 1998). An interface is currently under development to collect direct evaluations.

In DLSI, each element class and instance of a class is assigned unique ID number, which makes Collaborative Filtering integration possible. (Wrappers maintain unique IDs on the documents they parse and their elements.) The Collaborative Filtering engine has a separate database that stores users' click streams. Mutiple needs or multiple contexts of a user can be supported with Collaborative Filtering by combining the Collaborative Filtering algorithm with other context information, such as current task (Good et al., 1999; Im & Hars, 2001). It has been shown that the accuracy and optimal configurations of Collaborative Filtering system vary depending on user's needs (Im & Hars, 2001). Collaborative Filtering treats different tasks separately and provides different recommendations depending on the user and his/her current task.

In order to support mutiple contexts, the interface intends to ask users to optionally enter their current task. They can choose one from a predefined list or add a new one if it is not in the list. (DLSI's user preference module—part of the Integration Manager—will manage this.) Thus, the link order very likely will vary for the same user across task contexts. It is expected that rank ordering will improve user performance because it reduces the time in searching for appropriate links. Collaborative Filtering and Rank Ordering should also lead the user to his/her choice of item faster the before.

2.2 DLSI at NSDL

The NSDL consists of 119 projects of which most of them are collections. The DLSI is not counted towards collections. It is basically a service that will operate on the collections that reside in the NSDL repository. The positive aspect of DLSI is that, it will not be operating in isolation. The DLSI, in co-ordination with other services of NSDL can bring in lot of advantages to the collection and service provider. The benefits that collections will derive by integrating with DLSI are:

- Users of a collection or service will have direct access to related collections and services through DLSI linking. In effect, this enlarges the feature set of a given collection or service. Users will now have anchor tags leading to the information he/she wants from a wide gamut of collections.
- Collections and services will gain much wider use, because DLSI linking will lead users of other services and collections to them. This will be quite advantageous for the collection providers.
- It is known fact that people don't use the web search effectively. Also, if presented with thousands of search results, the user tends to get lost. The DLSI helps the users in a way by providing the anchor tags in accordance with their preferences. In this way the user does not feel lost. The DLSI uses the services of search and discovery module of NSDL to fetch the objects of desire and converts it into the anchor tags.
- Similarly, users will become aware of a service or collection from seeing its links included in DLSI's list of links when using other collections and services.

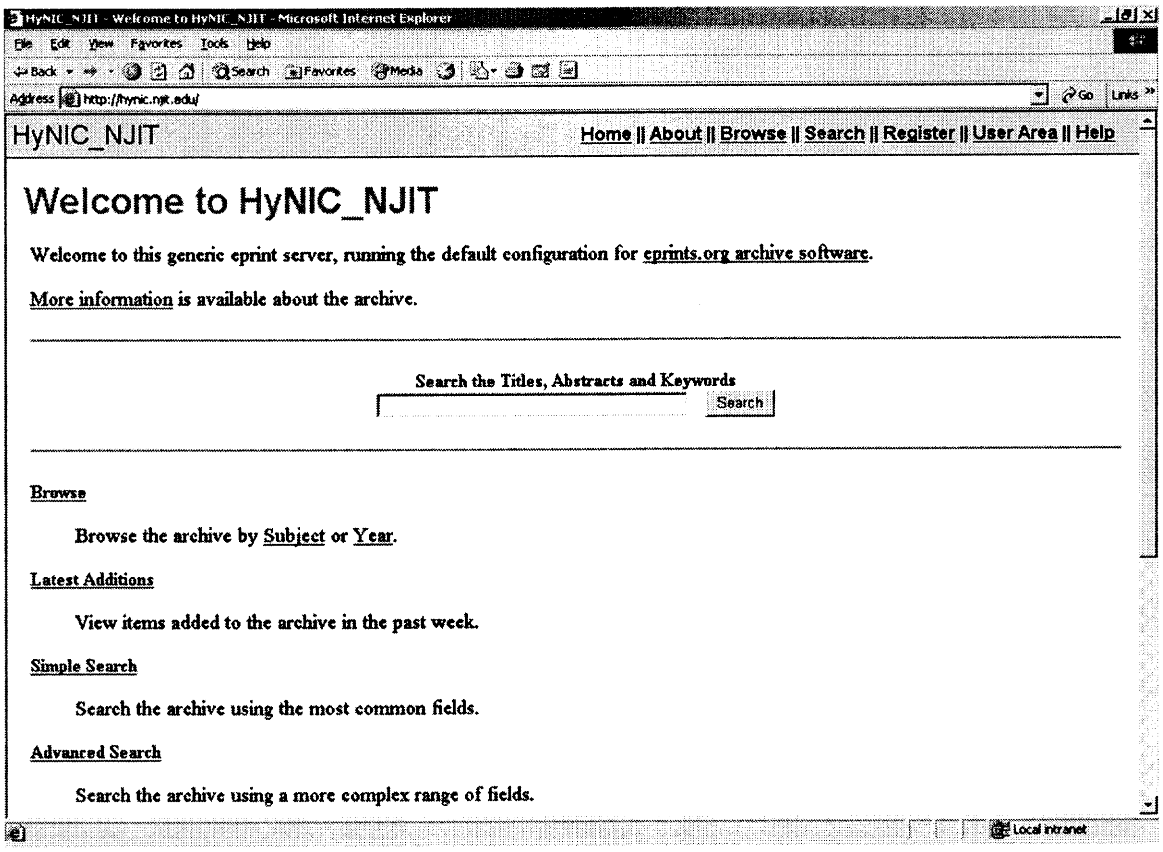
- Within a particular service or collection, DLSI gives the user direct, context-sensitive access to relevant features that the service or collection provides. This streamlined access will enable users possibly to bypass a series of menus or steps to reach a desired feature.
- Collection's metadata will get a much wider scope. This metadata in conjunction with <Meta> tag will enable the collection to be accessed from wide array of other collections. In this way it will optimize the use of metadata.

2.3 DLSI with Other Repositories

Apart from providing services to NSDL's collections, the DLSI can also integrate with other services and collections. If the collection or service has an appropriate application-programming interface (API), then it usually is quite easy to parse the output displays and detect the elements within it. If the collection or service provides adequate metadata in tags or metadata as collection in Dublin Core Format (which is becoming increasingly prevalent with XML), parsing can take advantage of these and be straightforward. OAI Parsing can be performed on the existing metadata repository to fetch the metadata. If displays follow a well-defined template or format, which is the case with many services, then parsing also should be relatively easy. Otherwise, if a document's content is unstructured and without embedded metadata, then DLSI may have to rely solely on lexical analysis to identify elements of interest within it. Also, if the document has proper HTML or XML tags (for e.g. all the open tags are properly closed), then it becomes quite easy to parse those pages to fetch the information.

Currently, the project team is working on a project, which will integrate the NJIT Highlander system with the "Shibboleth"- ([Marlena Erdos and Scott Cantor]) The Access Management System of NSDL. The integration will be discussed in detail in the Design and Implementation Chapter.

One more system under development in the Collaborative Hypermedia Laboratory in NJIT that will finally integrate with NSDL is the HyNIC repository. This repository will also be using the services of DLSI. This repository houses the conference proceedings, conference papers, conference posters, books, book chapter, journals, thesis, technical report, news article, pre-prints, and other such related collections. The metadata for all the records in the repository is the Dublin Core Metadata Element Set ([http://dublincore.org/documents/dces/]). Dublin Core will be further discussed under the topic of Metadata Management Information. This repository has the required Application Programming Interface (API), which makes it really easy for the DLSI to parse the output screen before it being displayed to the user. The users of the repository will be using the system just as before. They will be provided with an additional anchor tags generated with the DLSI. The connection of the repository with the NSDL server will be established with the help of Open Archives Initiatives (OAI) Server. Metadata Management service will harvest the metadata collection using the Open Archives Initiative Protocol for Metadata Harvesting (OAIPMH). This protocol will be explained in detail in the Metadata Management section.


HyNIC_NJIT - Welcome to HyNIC_NJIT - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://hynic.njit.edu/
Go Links
HyNIC_NJIT
Home || About || Browse || Search || Register || User Area || Help
Local intranet

Welcome to HyNIC_NJIT

Welcome to this generic eprint server, running the default configuration for eprints.org archive software.

More information is available about the archive.

Search the Titles, Abstracts and Keywords

Search

Browse

Browse the archive by Subject or Year.

Latest Additions

View items added to the archive in the past week.

Simple Search

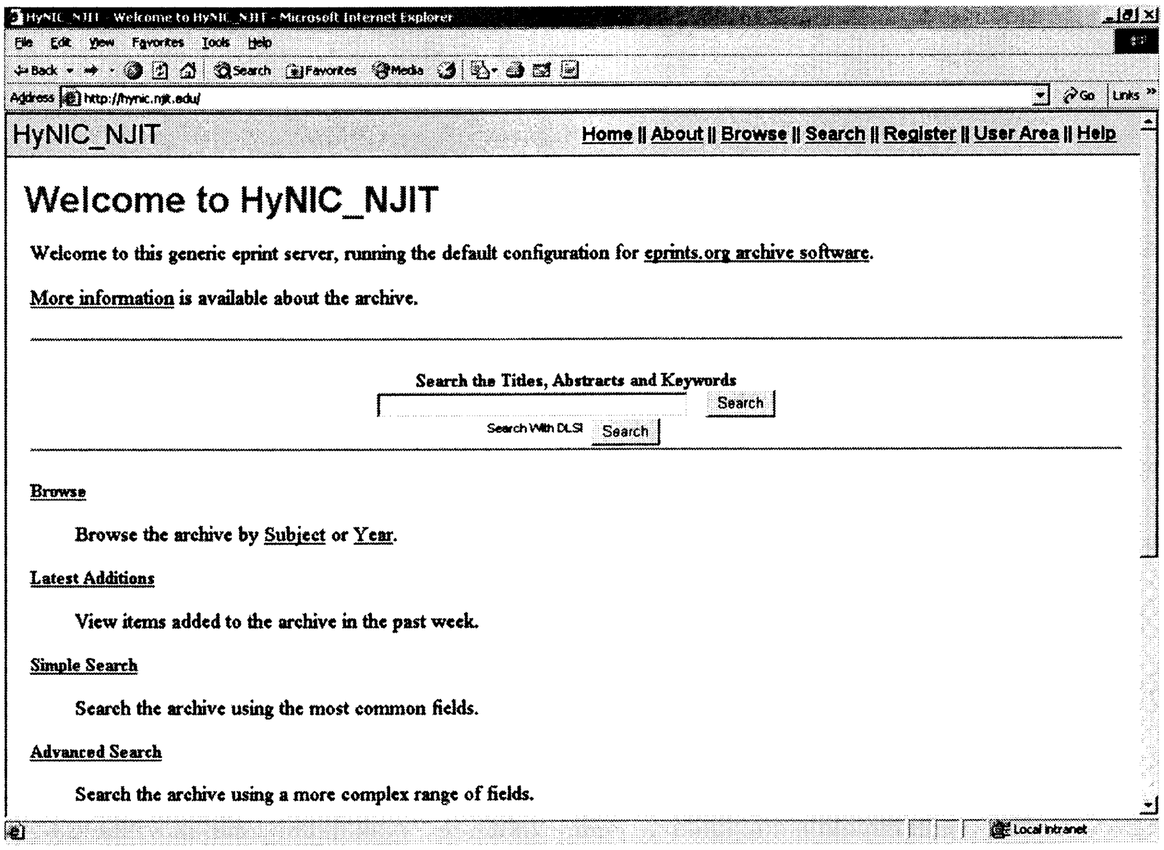
Search the archive using the most common fields.

Advanced Search

Search the archive using a more complex range of fields.

Figure 2.2 HyNIC Repository without DLSI Integration.

Description: This is the figure of HyNIC repository that is currently under development in the Collaborative Hypermedia Laboratory, NJIT. It is developed for the SIGWEB arm of Association of Computing Machinery (ACM).



HyNIC_NJIT - Welcome to HyNIC_NJIT - Microsoft Internet Explorer

Address http://hynic.njit.edu/

HyNIC_NJIT Home || About || Browse || Search || Register || User Area || Help

Welcome to HyNIC_NJIT

Welcome to this generic eprint server, running the default configuration for eprints.org archive software.

More information is available about the archive.

Search the Titles, Abstracts and Keywords [Search]

Search With DLSI [Search]

Browse

Browse the archive by Subject or Year.

Latest Additions

View items added to the archive in the past week.

Simple Search

Search the archive using the most common fields.

Advanced Search

Search the archive using a more complex range of fields.

Figure 2.3 HyNIC repository integrated with DLSI.

Description: In this screen, it can be seen that the user has the option to parse the search result page. A new search button has been included in this page. The text before it says, "Search with DLSI". If the user clicks on this button, then system will consider parsing the result page.

CHAPTER 3

DESIGN AND IMPLEMENTATION

3.1 Wrappers

Wrappers form the integral part of the DLSI project as a whole. Wrappers are potentially the hardest part of the project to implement. This is due to the intricacies involved in its interaction with the services of NSDL. The wrapper will implement three different types of protocols, namely, Security Assertion Mark Up Language (SAML), Open Archives Initiatives Protocol for Metadata Harvesting (OAIPMH), Simple Digital Library Interoperability Protocol (SDLIP), and Hyper Text Transfer Protocol (HTTP).

Wrappers serve three purposes:

- A wrapper connects collections and services with the Integration Manager, and thus indirectly with other collections and services.
- A wrapper receives output produced by its collection or service. This output can include documents and screens for display on the user's Web browser. The wrapper parses the output to detect all elements of interest within it. The wrapper then creates a message in DLSI's protocol containing the display output and the list of elements and their classes/kinds, and passes this message to the DLSI Integration Manager.
- A wrapper translates incoming requests from other systems from DLSI's protocol format into the format its collection or service expects.

The importance and location of wrappers will become clearer if displayed diagrammatically. The location of wrapper within the DLSI Infrastructure is shown as below:

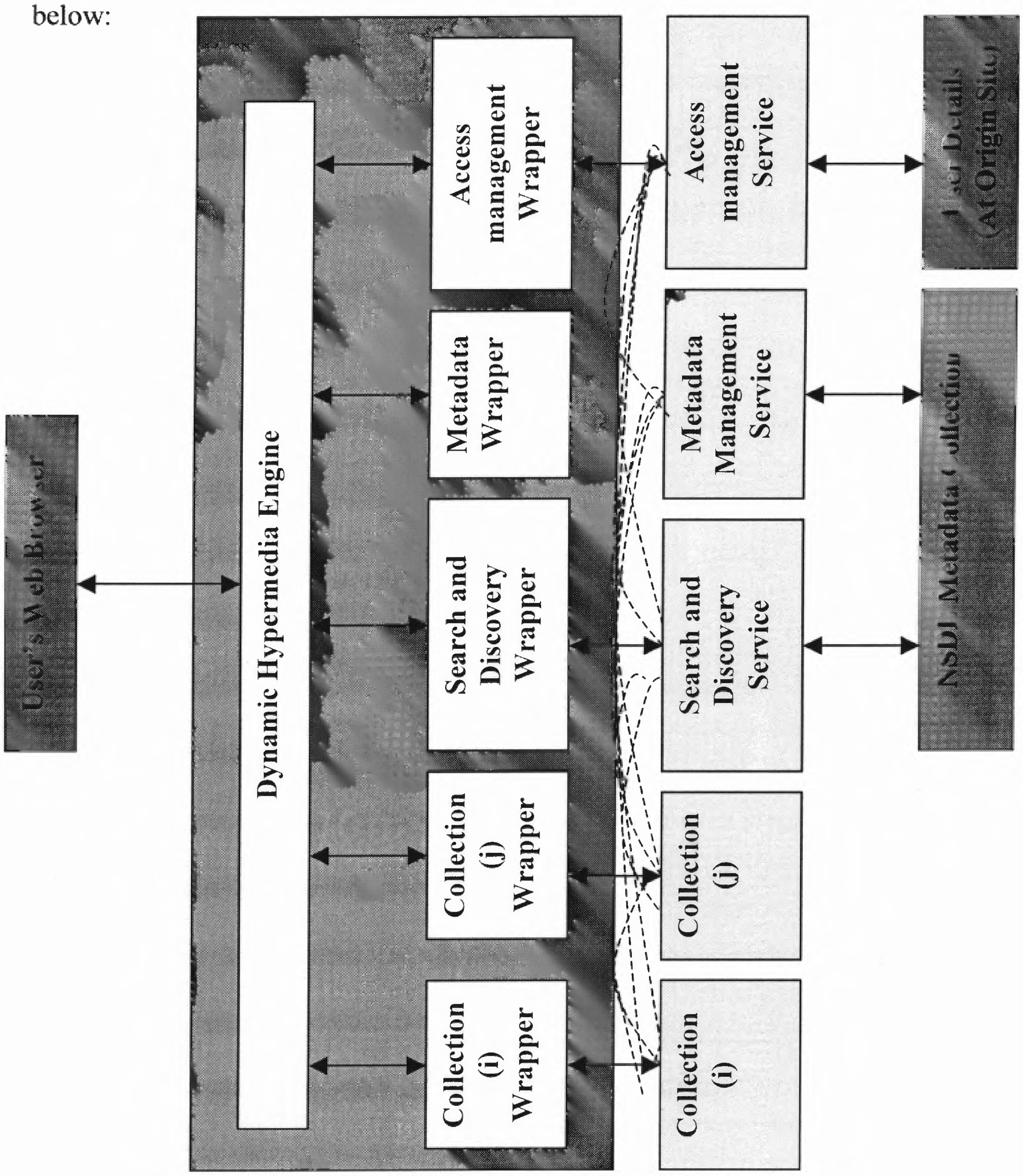

User's Web Browser
Dynamic Hypermedia Engine
Access management Wrapper
Metadata Wrapper
Search and Discovery Wrapper
Collection (j) Wrapper
Collection (i) Wrapper
Access management Service
Metadata Management Service
Search and Discovery Service
Collection (j)
Collection (i)
(At Origin Site)
NSDL Metadata

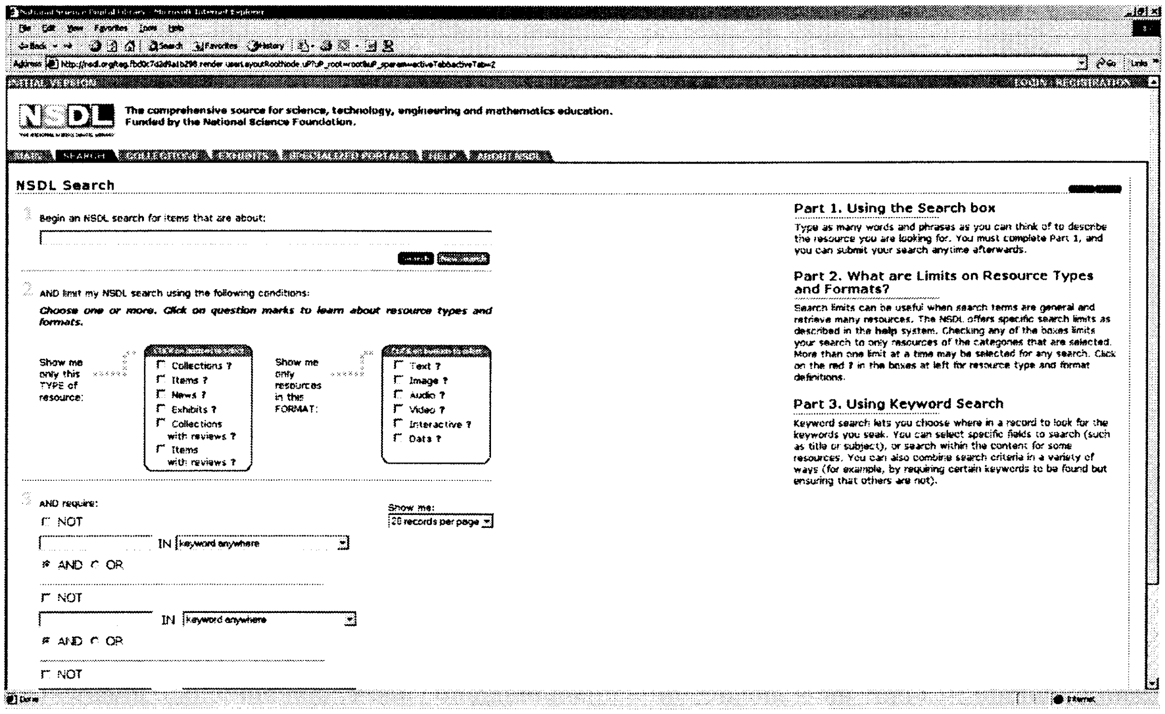
Figure 3.1 DLSI Architecture.

Description: The above figure shows the design view of DLSI. White boxes within the shaded portion represent Dynamic Hypermedia Engine and the wrappers. They both together constitute the DLSI. Below the wrappers are the services and collections of NSDL or any other library in general. Services in turn use other collections.

In the following subsections, let's explain each of the wrappers in detail.

3.1.1 Search and Discovery

The NSDL already has a search facility within its system. Its functionality, in brief, can be explained in two lines. It takes the Search Query provided by the user as input. It searches the metadata for this query and returns the result back to the user. The following diagram shows the search page of NSDL:


File Edit View Favorites Tools Help
Back Search Favorites History
Address http://nsdl.org/tag.fbd0c7d2d9a1b298.render.userLayoutRootNode.uP?uP_root=root&uP_sparam=activeTab&activeTab=2
SEARCH

The comprehensive source for science, technology, engineering and mathematics education.
Funded by the National Science Foundation.

NSDL Search

1 Begin an NSDL search for items that are about:

2 AND limit my NSDL search using the following conditions:

Choose one or more. Click on question marks to learn about resource types and formats.

Show me only this TYPE of resource:
- Collections ?
- Items ?
- News ?
- Exhibits ?
- Collections with reviews ?
- Items with reviews ?

Show me only resources in this FORMAT:
- Text ?
- Image ?
- Audio ?
- Video ?
- Interactive ?
- Data ?

3 AND require:

NOT
IN keyword anywhere
[✓ AND] OR

NOT
IN keyword anywhere
[✓ AND] OR

NOT

Show me: 20 records per page

Part 1. Using the Search box

Type as many words and phrases as you can think of to describe the resource you are looking for. You must complete Part 1, and you can submit your search anytime afterwards.

Part 2. What are Limits on Resource Types and Formats?

Search limits can be useful when search terms are general and retrieve many resources. The NSDL offers specific search limits as described in the **help** system. Checking any of the boxes limits your search to only resources of the categories that are selected. More than one limit at a time may be selected for any search. Click on the red ? in the boxes at left for resource type and format definitions.

Part 3. Using Keyword Search

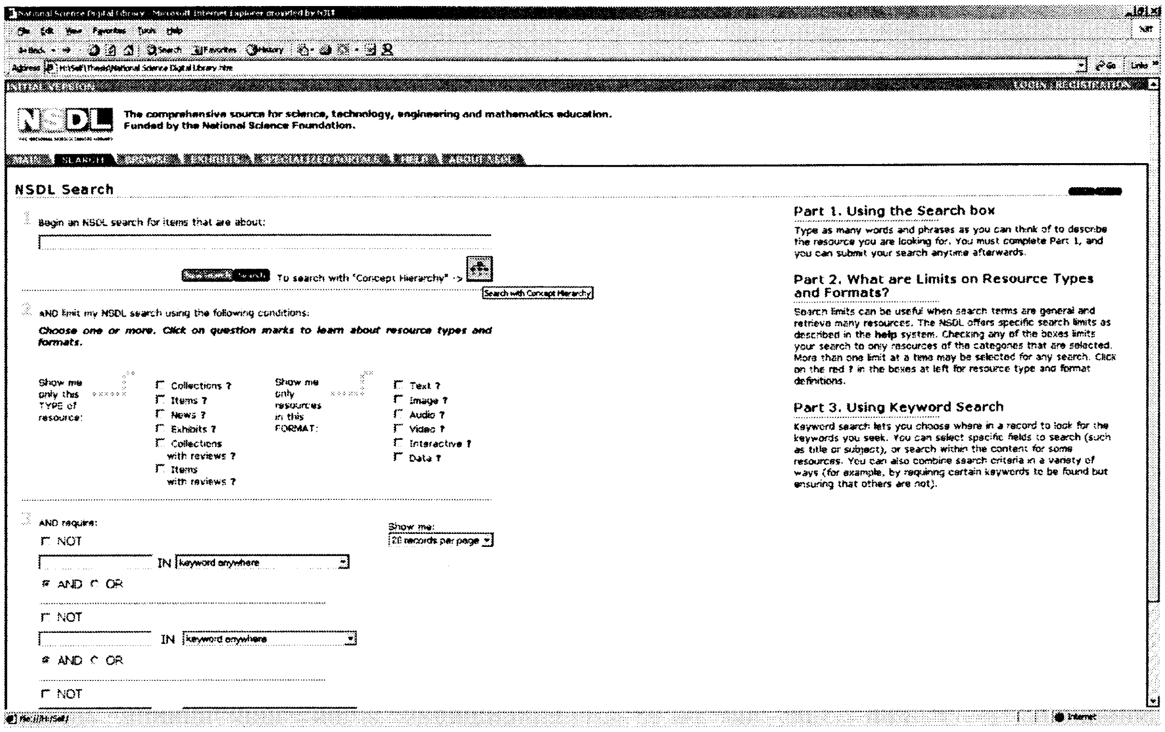
Keyword search lets you choose where in a record to look for the keywords you seek. You can select specific fields to search (such as title or subject), or search within the content for some resources. You can also combine search criteria in a variety of ways (for example, by requiring certain keywords to be found but ensuring that others are not).

Figure 3.2 NSDL Search page.

([http://nsdl.org/tag.a1437ade41ac8ef2.render.userLayoutRootNode.uP?uP_root=root&uP_sparam=activeTab&activeTab=2])

Description: This page doesn't give user an option for Filtering and Rank Ordering their links.

The main disadvantage in this search is that it does not give the weighted relevance on each of the page. To overcome this obstacle, DLSI will be using the services of Filtering and Rank Ordering. The diagram below shows, how user will get the option for Filtering and Rank ordering the links.


National Science Digital Library - Microsoft Internet Explorer provided by NJIT
File Edit View Favorites Tools Help
Back Search Favorites History
Address H:\Self\Thesis\National Science Digital Library.htm
INITIAL VERSION
The comprehensive source for science, technology, engineering and mathematics education.
Funded by the National Science Foundation.
SEARCH

NSDL Search

1. Begin an NSDL search for items that are about:

To search with "Concept Hierarchy" ->

Search with Concept Hierarchy

2. AND limit my NSDL search using the following conditions:

Choose one or more. Click on question marks to learn about resource types and formats.

Show me only this TYPE of resource:
- Collections ?
- Items ?
- News ?
- Exhibits ?
- Collections with reviews ?
- Items with reviews ?

Show me only resources in this FORMAT:
- Text ?
- Image ?
- Audio ?
- Video ?
- Interactive ?
- Data ?

3. AND require:

NOT

IN keyword anywhere

[✓ AND] OR

NOT

IN keyword anywhere

[✓ AND] OR

NOT

Show me: 20 records per page

Part 1. Using the Search box

Type as many words and phrases as you can think of to describe the resource you are looking for. You must complete Part 1, and you can submit your search anytime afterwards.

Part 2. What are Limits on Resource Types and Formats?

Search limits can be useful when search terms are general and retrieve many resources. The NSDL offers specific search limits as described in the **help** system. Checking any of the boxes limits your search to only resources of the categories that are selected. More than one limit at a time may be selected for any search. Click on the red ? in the boxes at left for resource type and format definitions.

Part 3. Using Keyword Search

Keyword search lets you choose where in a record to look for the keywords you seek. You can select specific fields to search (such as title or subject), or search within the content for some resources. You can also combine search criteria in a variety of ways (for example, by requiring certain keywords to be found but ensuring that others are not).

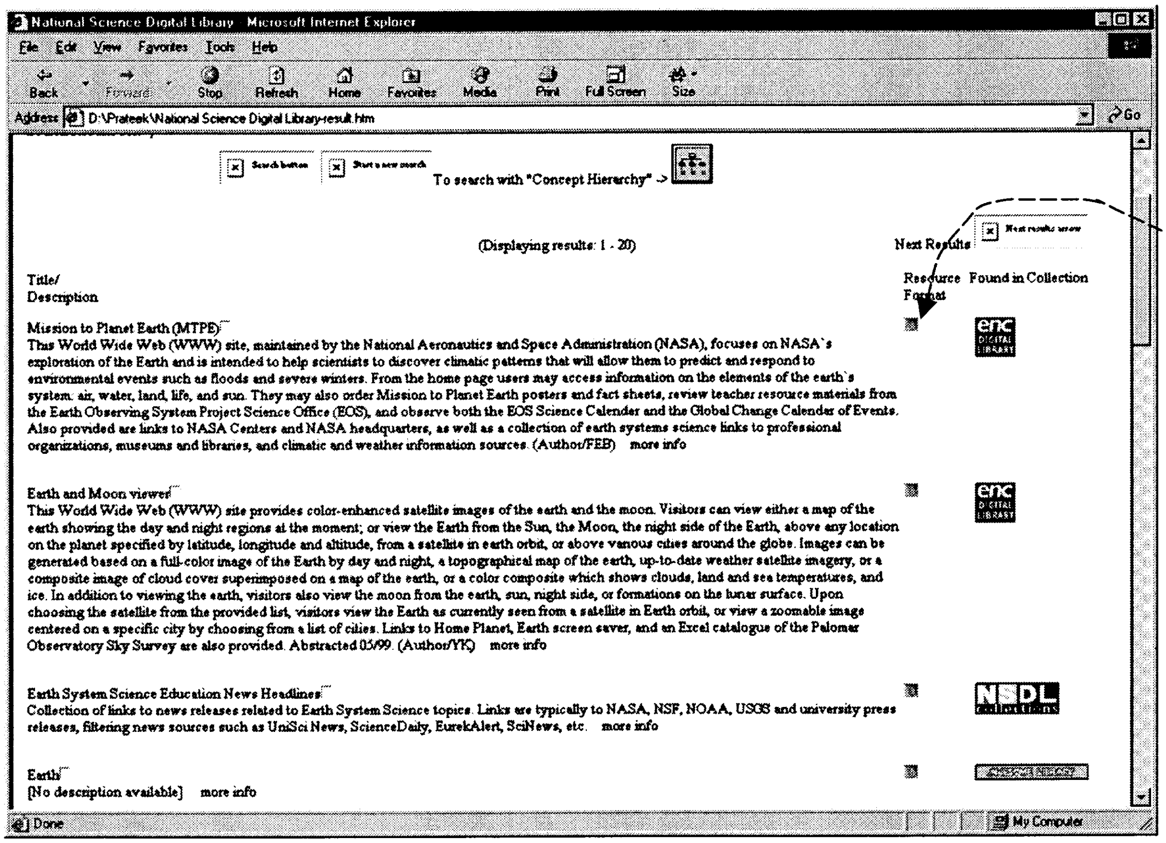
file:///H:/Self/

Figure 3.3 NSDL Search page with DLSI link.

Description: This page has an additional button. If the user wants the links to be filtered and rank ordered, he/she clicks on that.

Another disadvantage in search and discovery service of NSDL is that relationship analysis of all the links and results is missing. To overcome this, DLSI wrappers will harvest all the links from the search result page of the NSDL, do a relationship analysis on that and will pass back the result to the user. Now the difference is that the user will see additional anchor tags on the page. This is entirely user's choice.

If the user prefers to do a relationship analysis on the links, he/she will click on the button. Wrappers will also pass the result to the Filtering and Rank Ordering module to rank order the pages. This exchange of data will take place with the help of XML. The object to be used here is MSXML2.ServerXMLHTTP.


National Science Digital Library - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Favorites Media Print Full Screen Size
Address D:\Prateek\National Science Digital Library-result.htm
Go
Search button
Start a new search
Next results arrow
Done
My Computer

To search with "Concept Hierarchy" ->

(Displaying results: 1 - 20)

Next Results

Title/
Description

Resource Format

Found in Collection

Mission to Planet Earth (MTPE)
This World Wide Web (WWW) site, maintained by the National Aeronautics and Space Administration (NASA), focuses on NASA`s exploration of the Earth and is intended to help scientists to discover climatic patterns that will allow them to predict and respond to environmental events such as floods and severe winters. From the home page users may access information on the elements of the earth`s system: air, water, land, life, and sun. They may also order Mission to Planet Earth posters and fact sheets, review teacher resource materials from the Earth Observing System Project Science Office (EOS), and observe both the EOS Science Calender and the Global Change Calender of Events. Also provided are links to NASA Centers and NASA headquarters, as well as a collection of earth systems science links to professional organizations, museums and libraries, and climatic and weather information sources. (Author/FEB) more info

Earth and Moon viewer
This World Wide Web (WWW) site provides color-enhanced satellite images of the earth and the moon. Visitors can view either a map of the earth showing the day and night regions at the moment; or view the Earth from the Sun, the Moon, the night side of the Earth, above any location on the planet specified by latitude, longitude and altitude, from a satellite in earth orbit, or above various cities around the globe. Images can be generated based on a full-color image of the Earth by day and night, a topographical map of the earth, up-to-date weather satellite imagery, or a composite image of cloud cover superimposed on a map of the earth, or a color composite which shows clouds, land and sea temperatures, and ice. In addition to viewing the earth, visitors also view the moon from the earth, sun, night side, or formations on the lunar surface. Upon choosing the satellite from the provided list, visitors view the Earth as currently seen from a satellite in Earth orbit, or view a zoomable image centered on a specific city by choosing from a list of cities. Links to Home Planet, Earth screen saver, and an Excel catalogue of the Palomar Observatory Sky Survey are also provided. Abstracted 05/99. (Author/YK) more info

Earth System Science Education News Headlines
Collection of links to news releases related to Earth System Science topics. Links are typically to NASA, NSF, NOAA, USGS and university press releases, filtering news sources such as UniSci News, ScienceDaily, EurekAlert, SciNews, etc. more info

Earth
[No description available] more info

Figure 3.4 Relationship Analysis tool within the web page.

Description: This picture shows the small button-type links (marked by arrow) on the right side of the page. These links appear after relationship analysis has been done.

MSXML2.ServerXMLHTTP enables user to establish HTTP connection between files or objects on different Web servers. The ServerXMLHTTP object offers functionality similar to that of the XMLHTTP object. Unlike XMLHTTP, however, the ServerXMLHTTP object does not rely on the WinInet control for HTTP access to remote

XML documents. ServerXMLHTTP uses a new HTTP client stack. Designed for server applications, this server-safe subset of WinInet offers the following advantages:

- Reliability—The HTTP client stack offers longer uptimes. WinInet features that are not critical for server applications, such as URL caching, auto-discovery of proxy servers, HTTP/1.1 chunking, offline support, and support for Gopher and FTP protocols are not included in the new HTTP subset.
- Security—The HTTP client stack does not allow a user-specific state to be shared with another user's session. ServerXMLHTTP does not provide support for certificates.

The ServerXMLHTTP object is commonly used to:

- Receive XML documents from an Active Server Pages (ASP) page on a local or remote Web server (HTTP GET).
- Post XML documents to an ASP page on a local or remote Web server (HTTP POST).
- Post and receive response XML documents from an ASP page (HTTP POST).

The ServerXMLHTTP open method makes the connection between servers and the send method sends the request. With ServerXMLHTTP, the usual sequence is to call **open**, set any custom header information through **setRequestHeader**, call **send**, and then check one of the four response properties.

ServerXMLHTTP offers additional benefits for transporting XML data:

- ServerXMLHTTP maintains state, so transactional XML data can be used in applications that require real-time responses.

- ServerXMLHTTP allows you to serve the response object as a stream or Document Object Model (DOM) object. As a stream, this offers considerable performance benefits when moving data through the HTTP protocol.

ServerXMLHTTP offers some backward compatibility with XMLHTTP. Source code (JScript, Visual Basic Scripting Edition, Visual Basic, or C++) that uses the XMLHTTP component can be easily modified to use the new ServerXMLHTTP component. The maximum number of instances that can exist simultaneously within a single process is 5,460. A similar limitation applies to the XMLHTTP component. However, other factors, such as available memory, CPU processing capacity, or available socket connections can further limit the number of instances that can be active simultaneously. Partition the server application into multiple processes if this limit becomes a bottleneck. The IServerXMLHTTPRequest interface inherits from IXMLHTTPRequest and extends it with the following four new methods: getOption, setOption, waitForResponse, setTimeouts.

All the details above were described to convert the returned search page into XML format and send it to Filtering and Rank Ordering Module. The following describes a way to send a search query to the Search and Discovery module of NSDL.

Search and Discovery module uses Simple Digital Library Interoperability Protocol (SDLIP) ([A. Paepcke, R. Brandriff et. al, http://www-diglib.stanford.edu /~testbed/doc2/SDLIP]) to request searches to be performed over information sources. The DLSI Search and Discovery service, and other clients use SDLIP to request searches to be performed over information sources. The result documents are returned synchronously, or they are streamed from service to client as they become available.

Implementations can be constructed over HTTP or CORBA based transports. In fact, any search service can be accessible through both kinds of transports at the same time.

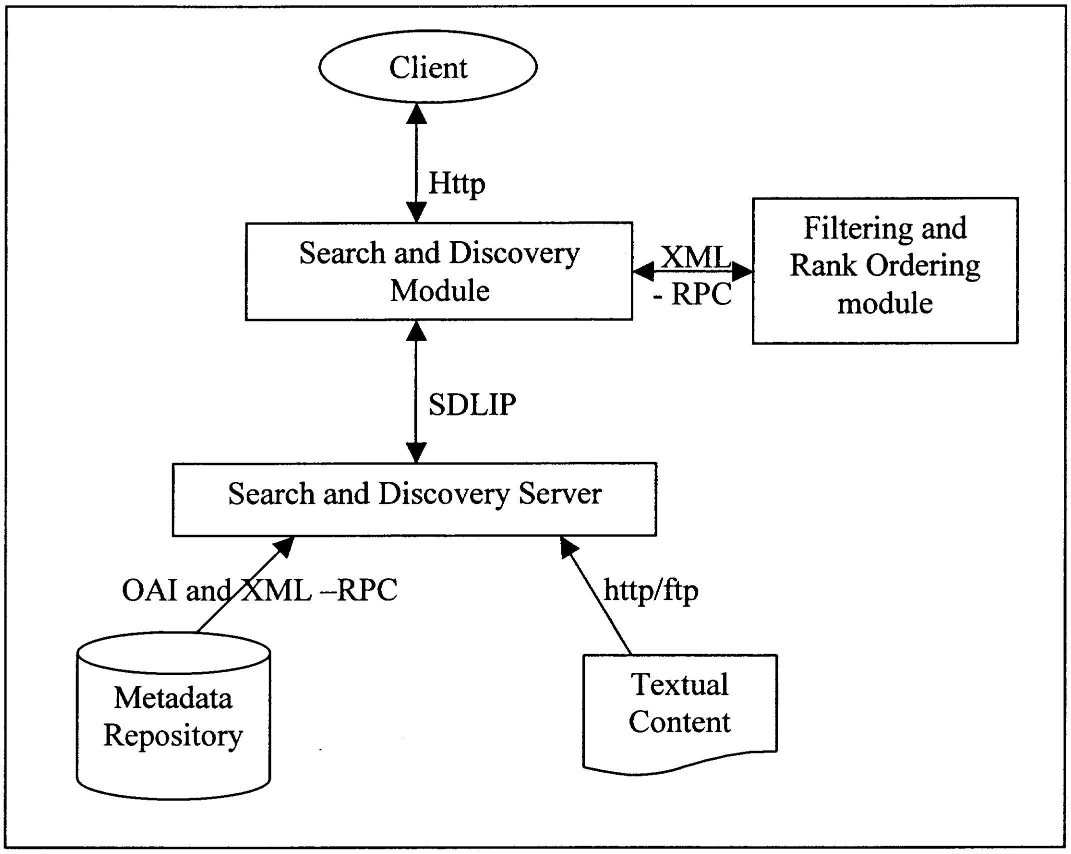

Client
Http
Search and Discovery Module
XML - RPC
Filtering and Rank Ordering module
SDLIP
Search and Discovery Server
OAI and XML –RPC
http/ftp
Metadata Repository
Textual Content

Figure 3.5 Role of SDLIP in integrating DLSI search and discovery wrapper with NSDL Search and discovery server.

Description: The diagram shows the protocols between service-to-service and service-to-collection. It also depicts the role of Filtering and Rank Ordering module.

It must be noted that the information to be served to the user need not be transferred using SDLIP protocol. A form is made especially for that. The user will populate the field by entering the search query and on submission; a procedure will be called which will link the DLSI to the Search and Discovery server at NSDL's end. It will transfer the query using SDLIP and get back the result.

SDLIP has the following goals:

- Simplicity for both client and server side implementations
- Implementations possible via both distributed object technology, such as CORBA, and via HTTP
- Support for stateful and stateless operation at the server side
- Support for dynamic load balancing in server implementations
- Support for thin clients, such as handheld devices

3.1.2 Access Management

The DLSI will be maintaining user's preferences and privileges. The DLSI is using the services of Access Management wrapper for this. Access management service of NSDL is connected to the User Profile server of NSDL. The NSDL maintains a unique profile for each of the user. If the user logs in through the DLSI interface, then as soon as he/she logs-in to www.nsdl.org, DLSI access management wrapper will fetch the user details and pass it on to the Collaborative Filtering Engine which maintain all the user's profile. Collaborative Filtering Engine expects the following parameters to be passed from Access Management Wrapper:

- User-Id.
- Task-Id.
- Source (IP or URL)
- Access Time
- User Agent (User Browser)
- Referrer

- Link-Id.

Only User-Id will be provided by the Access Management Service of NSDL. Other parameters will be created by the DLSI Access Management wrapper. Output from the Collaborative Filtering Engine to Access Management Module is:

- Link-Id. (Sorted)

The method which Access Management Service of DLSI follows to fetch or generate these parameters depends upon the way the user logs into the NSDL's website (www.nsdl.org). Access Management in NSDL is maintained by Shibboleth ([Erdos, Marlena, and Cantor, Scott]).

The following explains in detail the entire flow, from the user's initial point of contact through the "Shibboleth Attribute Requester" better known as SHAR to the parsing of attributes about the user to the DLSI.

1) The user makes an initial request for a resource protected by a Shibboleth Indexical Reference Establisher (SHIRE).
2) The SHIRE obtains the URL of the user's Handle Service (HS), or redirects the user to a "Where Are You From? (WAYF)" service for this purpose.
3) The SHIRE or WAYF asks the HS to create a handle for this user, redirecting the request through the user's browser.
4) The HS returns an opaque handle for the user that can be used by the SHAR to get attributes from the appropriate Attribute Authority at the origin site. The SHIRE, after performing impersonation checks, passes on the handle (and Attribute Authority information, and organization name) to the SHAR.

5) The SHAR asks the Attribute Authority for attributes via an "Attribute Query Message" AQM message.
6) It receives attributes back from the Attribute Authority via an "Attribute Response Message" ARM message.
7) It passes the attribute of the user to DLSI using Security Assertion Markup Language "SAML" protocol.

Shibboleth uses SAML formats and binding protocols whenever possible and appropriate. Of particular note: Shibboleth uses the SAML query and response protocol and formats for the AQM and ARM messages, interacting with DLSI, and Shibboleth uses SAML's attribute statement and assertion format.

Currently, the DLSI team is discussing with team at Columbia University to test-implement Shibboleth structure with NJIT Highlander account. In this collaboration, DLSI will act as "Origin Site" for NJIT. The deployment will be with Shibboleth 8.0. ([http://marsalis.internet2.edu/cgi-bin/viewcvs.cgi/*checkout*/shibboleth/DEPLOY-GUIDE-ORIGIN.html?rev=HEAD&content-type=text/html]).

There are four primary components to DLSI in Shibboleth: the Attribute Authority (AA), the Handle Service (HS), the directory service, and the local sign-on system (SSO). The AA and HS have already been explained above. The AA and HS come along with Shibboleth and an open-source WebISO solution Pubcookie can be obtained from www.pubcookie.org. It is the responsibility of origin site to provide the directory. Shibboleth is able to interface with a directory exporting an LDAP interface containing user attributes, and is designed such that programming interfaces to other repositories are readily implemented. Shibboleth relies on standard web server mechanisms to trigger local authentication. A.htaccess file can be easily used to trigger the local WebISO system or the web server's own Basic Authorization mechanism, which will likely utilize an enterprise authentication system, such as Kerberos [http:// web.mit.edu/kerberos/www/].

From the DLSI's point of view, the first contact will be the redirection of a user to the handle service, which will then consult the SSO system to determine whether the user has already been authenticated. If not, then the browser user will be asked to authenticate, and then sent back to the target URL with a handle bundled in an attribute assertion. Next, a request from the Shibboleth Attribute Requester (SHAR) will arrive at the AA which will include the previously mentioned handle. The AA then consults the Attribute Release Policies (ARP's) for the directory entry corresponding to the handle, queries the directory for these attributes, and releases to the SHAR all attributes the SHAR is entitled to know about that user. To understand this procedure more clearly, let us take an example. Let NJIT be the origin site. WAYF ("Where Are You From?") service at NJIT can be the Highlander service. It redirects the user to the known address for the handle service of NJIT, which will be part of DLSI. Two important requirements for Shibboleth to work from the Origin Site are:

- Shibboleth is known to work on Linux and Solaris, but should function on any platform that has a Tomcat implementation.
- It is recommended that a web server must be deployed that can host Java servlets and Tomcat, although not explicitly necessary, as Tomcat can still host without it.

When the set up is done, DLSI users become part of an organization which agrees to exchange attributes of the user using SAML protocol and abide by common set of policies and protocols. If origin site also becomes part of the club, it dramatically expands the number of targets and origins that can interact without defining bilateral agreements. To install Shibboleth, the following softwares are needed:

- Apache 1.3.26+
- Tomcat 4.1.18+ Java server
- Sun J2SE v 1.4.1_01 SDK
- mod_jk: It can be built against Apache which generally requires gcc or g++compiler
- An enterprise authentication mechanism: Ideally, this will be a WebISO or SSO system such as Pubcookie. The minimal requirement is for the web server to be able to authenticate browser users and supply their identity to the Handle Server.
- An enterprise directory service: Shibboleth currently supports retrieving user attribute information from an LDAP directory.

A complete installation guide can be found at:

http://marsalis.internet2.edu/cgi-bin/viewcvs.cgi/*checkout*/shibboleth/DEPLOY-GUIDE-ORIGIN.html?rev=HEAD&content-type=text/html

3.1.3 Metadata Management

Metadata is supposed to be the most important of any repository. The NSDL maintains exhaustive metadata collection. For the DLSI to mark anchor tags all over the document by identifying elements of interest, it need to first check the metadata related just to that document. At this stage of the project, NSDL does not support that. Alternately, this can be implemented is that, DLSI metadata wrapper harvest the entire metadata repository and then check on that to fetch the metadata of interest. Metadata for a particular document is of real interest for the DLSI, as it can get so much information out of it. This is the best way to know to know about the document's field of study. The entire procedure of metadata harvesting is explained below.

Let us start with the definition of metadata. Metadata is usually defined as "data about data" ([http://metamanagement.comm.nsdlib.org/overview.html#what]) but it may be far more useful to say that it consists of standardized descriptions of resources, whether digital or physical, that aid in the discovery and retrieval of those resources. Libraries have been creating metadata for centuries, in the form of book catalogs, card catalogs and, more recently, online catalogs. In a similar fashion, for over a century, indexing and abstracting agencies have been creating metadata describing journal articles to aid in their discovery and retrieval. A prescribed set of possible descriptive statements is known as a metadata "format" or schema.

There are many "flavors" of metadata differing in basic purpose (administrative, structural or descriptive), depth and richness, or specificity for a particular domain. There is no "one-size-fits-all" metadata solution, although there are ways to "translate" one schema into another, similar to the ways one language can be translated into another. In addition (as will be discussed in greater detail later), there are ways to mix and match metadata statements from different recognized metadata schemas in order meet the unique blend of needs of a particular project

Metadata allows the precise description of resources (and the sharing of such descriptions) in relatively small and discrete packages of information called metadata records, without the necessity of involving the resources themselves in the transaction.

For example, in the networked environment of the Web, metadata records describing resources useful in education can be gathered up (harvested) from geographically distributed resource repositories without affecting the location of the resources they describe. These harvested metadata records can be assembled in metadata repositories where they can function as an online catalog of distributed teaching and learning resources. The primary reason for this in the NSDL context is for resource discovery.

The NSDL Standards Working Group has determined that the Dublin Core set of 15 elements, their associated element refinements plus the three IEEE elements recommended by the DC Education Working Group (when available), will be the standard set used by the NSDL metadata repository.

The Working Group has also identified eight different metadata formats that can be accommodated by the NSDL metadata repository, based on the availability of crosswalks from those formats to Dublin Core. Projects using any of these eight formats in a standard manner can be assured that their metadata will be usable by NSDL.

Though NSDL supports eight different formats of metadata, DLSI at this stage supports only Dublin Core ([www.dublincore.org]) format. The Dublin Core metadata standard is a simple yet effective element set for describing a wide range of networked resources. The Dublin Core standard comprises fifteen elements, the semantics of which have been established through consensus by an international, cross-disciplinary group of professionals from librarianship, computer science, text encoding, the museum community, and other related fields of scholarship.

Another way to look at Dublin Core is as a "small language for making a particular class of statements about resources" ([Baker]). In this language, there are two classes of terms--elements (nouns) and qualifiers (adjectives)--which can be arranged into a simple pattern of statements. The resources themselves are the implied subjects in this language. In the diverse world of the Internet, Dublin Core can be seen as a "metadata pidgin for digital tourists": easily grasped, but not necessarily up to the task of expressing complex relationships or concepts. The Dublin Core metadata element set is a standard for cross-domain information resource description. Elements can be in simple dublincore format or qualified dublincore format. Currently, DLSI supports only simple dublincore format. Dublin Core elements are outlined as follows ([http://dublincore.org /documents/dces/])

Table 3.1 Description of 15 Elements types of Dublin Core.

Element Name: Title		
	Label:	Title
	Definition:	A name given to the resource.
	Comment:	Typically, Title will be a name by which the resource is formally known.
Element Name: Creator		
	Label:	Creator
	Definition:	An entity primarily responsible for making the content of the resource.
	Comment:	Examples of Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity.
Element Name: Subject		
	Label:	Subject and Keywords
	Definition:	A topic of the content of the resource.
	Comment:	Typically, Subject will be expressed as keywords, key phrases or classification codes that describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.
Element Name: Description		
	Label:	Description
	Definition:	An account of the content of the resource.
	Comment:	Examples of Description include, but are not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content.

Element Name: Publisher		
	Label:	Publisher
	Definition:	An entity responsible for making the resource available
	Comment:	Examples of Publisher include a person, an organization, or a service.
Element Name: Contributor		
	Label:	Contributor
	Definition:	An entity responsible for making contributions to the content of the resource.
	Comment:	Examples of Contributor include a person, an organization, or a service. Typically, the name of a Contributor should be used to indicate the entity.
Element Name: Date		
	Label:	Date
	Definition:	A date of an event in the lifecycle of the resource.
	Comment:	Typically, Date will be associated with the creation or availability of the resource. Recommended best practice for encoding the date value is defined in a profile of ISO 8601 [W3CDTF] and includes (among others) dates of the form YYYY-MM-DD.
Element Name: Type		
	Label:	Resource Type
	Definition:	The nature or genre of the content of the resource.
	Comment:	Type includes terms describing general categories, functions, genres, or aggregation levels for content. Recommended best practice is to select a value from a controlled vocabulary (for example, the DCMI Type Vocabulary [DCT1]).
Element Name: Format		
	Label:	Format
	Definition:	The physical or digital manifestation of the resource.
	Comment:	Typically, Format may include the media-type or dimensions of the resource. Format may be used to identify the software, hardware, or other equipment needed to display or operate the resource. Recommended best practice is to select a value from a controlled vocabulary (for example, the list of Internet Media Types [MIME] defining computer media formats).
Element Name: Identifier		
	Label:	Resource Identifier
	Definition:	An unambiguous reference to the resource within a given context.
	Comment:	Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Formal identification systems include but are not limited to the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI) and the International Standard Book Number (ISBN).
Element Name: Source		
	Label:	Source
	Definition:	A Reference to a resource from which the present resource is derived.
	Comment:	The present resource may be derived from the Source resource in whole or in part. Recommended best practice is to identify the referenced resource by means of a string or number conforming to a formal identification system.

Element Name: Language		
	Label:	Language
	Definition:	A language of the intellectual content of the resource.
	Comment:	Recommended best practice is to use RFC 3066 [RFC3066] which, in conjunction with ISO639 [ISO639]), defines two- and three primary language tags with optional sub-tags.
Element Name: Relation		
	Label:	Relation
	Definition:	A reference to a related resource.
	Comment:	Recommended best practice is to identify the referenced resource by means of a string or number conforming to a formal identification system.
Element Name: Coverage		
	Label:	Coverage
	Definition:	The extent or scope of the content of the resource.
	Comment:	Typically, Coverage will include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and to use.
Element Name: Rights		
	Label:	Rights Management
	Definition:	Information about rights held in and over the resource.
	Comment:	Typically, Rights will contain a rights management statement for the resource, or reference a service providing such information. Rights information often encompasses Intellectual Property Rights (IPR), Copyright, and various Property Rights. If the Rights element is absent, no assumptions may be made about any rights held in or over the resource.

It must be remembered that Metadata can be harvested from the repository using Open Archives Protocol for Metadata Harvesting (OAIPMH). Now lets talk about OAIPMH in detail, as this is very important to understand whosoever plans to harvest the metadata repository.

The Open Archives Initiative (OAI) ([http://openarchives.org]) announced the Open Archives Initiative Protocol for Metadata Harvesting (OAIMH) v2.0 ([http://www.openarchives.org/OAI/openarchivesprotocol.html]). OAIPMH 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 DLSI's metadata management wrapper will act as a *Service Provider.* It will be harvesting the repository to fetch elements of interest. So it will act as a harvester (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). At this stage, DLSI does not have capability of selective metadata harvesting. It has to harvest the entire repository and then filter the result to get the relevant metadata for the document. This can be achieved with the use of `<dc:document.id>` type tag for all the records. Another project known as HyNIC project for SIGWEB is to be integrated into the NSDL. It will act as a repository (A repository is a network accessible server that can process the OAI-PMH requests). With all the items in a repository, a Unique Identifier is associated. OAIPMH will request this unique identifier for extracting the metadata. The format of the unique identifier must correspond to that of the URI (Uniform Resource Identifier) syntax. Unique identifiers play two roles in the protocol:

- *Response:* Identifiers are returned by both the `ListIdentifiers` and `List Records` 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.

The DLSI will be fetching a record from the repository. 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 datestamp of the record.

The DLSI metadata management wrapper can also do selective harvesting. Harvesters may specify set membership as criteria for selective harvesting. To specify set-based selective harvesting, a setSpec is included as the value of the optional set argument to the ListRecords and ListIdentifiers requests, thereby specifying selective harvesting of records from items within the respective set. When a setSpec is used as an argument, the response must include:

- The records corresponding to the metadataPrefix argument, or headers thereof in the case of deleted records, available from those items in the set specified by the setSpec;
- The records corresponding to the metadataPrefix argument, or headers thereof in the case of deleted records, available from those items in sets that are descendant from the specified set.

Harvesters may use *datestamps* to harvest only those records that were created, deleted, or modified within a specified date range.

HTTP Request Format: It must be known that OAI-PMH requests are expressed as HTTP requests. OAI-PMH requests must be submitted using either the HTTP `GET` or `POST` method. URLs for `GET` requests have keyword arguments appended to the base URL, separated from it by a question mark `(?)`. For example, the URL of a `GetRecord` request to a repository with base URL that is `http://www.moorlandschool.co.uk/earth/` might be:

http://www.moorlandschool.co.uk/earth?verb=GetRecord&identifier=oai%3AarXiv.org %3Ahep-th%2F9901001&metadataPrefix=oai_dc

Submitting the same request using HTTP `POST` method would use just the base URL as URL. The format of `POST` can be:

```
POST http://www.moorlandschool.co.uk/earth HTTP/1.0

Content-Length: 100

Content-Type: application/x-www-form-urlencoded

Verb=GetRecord&identifier=oai%3AarXiv.org%3Ahepth%2F9901001&metad
ataPrefix=oai_dc
```

HTTP Response Format: Although DLSI wrappers will not use HTTP response format, it is mentioned because it will be implemented by HyNIC repository. All responses to OAI-PMH requests must be well-formed XML instance documents. Encoding of the XML must use the UTF-8 representation of Unicode. Character references, rather than entity references, must be used. Character references allow XML responses to be treated as stand-alone documents that can be manipulated without dependency on entity declarations external to the document. The XML data for all responses to OAI-PMH requests must validate against the XML Schema. HyNIC repository's successful reply to the GetRecord request is shown below:

```
<?xml version="1.0" encoding="UTF-8" ?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
         http://hynic.njit.edu/reply/OAI-PMH.xsd">
 <responseDate>2003-04-22T19:20:30Z</responseDate>
 <request verb="GetRecord" identifier="oai:arXiv.org:hep-th/9901001"
          metadataPrefix="oai_dc">http://hynic.njit.edu</request>
 <GetRecord>
   <record>...</record>
 </GetRecord>
</OAI-PMH>
```

The code for XML schema format for validating Responses to OAI-PMH Requests is provided in the APPENDIX.

Protocol Requests and Responses in Detail: GetRecord verb is used to retrieve an individual metadata record from a repository. It requires the following arguments:

- `identifier` a *required* argument that specifies the unique identifier of the item in the repository from which the record must be disseminated.
- `metadataPrefix` a *required* argument that specifies the `metadataPrefix` of the format that should be included in the metadata part of the returned record . A record should only be returned if the format specified by the `metadataPrefix` could be disseminated from the item identified by the value of the identifier argument. The metadata formats supported by a repository and for a particular record can be retrieved using the `ListMetadataFormats` request.
- `badArgument` - The request includes illegal arguments or is missing required arguments.

- `cannotDisseminateFormat` - The value of the `metadataPrefix` argument is not supported by the item identified by the value of the `identifier` argument
- `idDoesNotExist` - The value of the `identifier` argument is unknown or illegal in this repository.

XML Schema for validating Unqualified Dublin Core metadata associated with the reserved oai_dc metadataPrefix will be shown in the APPENDIX

This design and implementation section has covered the entire implementation of DLSI wrappers which will interact with the Core Services of NSDL and fetch the required result and feed the Dynamic Hypermedia Engine. The output from the engine will finally be diverted to the user.

CHAPTER 4

FUTURE ENHANCEMENTS

This is not the end of the project as There will be several modifications involved. The NSDL itself is an involving entity currently. The NSDL was launched in December 2002. There are still several issues that remain to be addressed and several possibility remains to be explored. Workgroups have been formed for the development of services and collections. Whosoever continues to work on this project, need to become a member of these communities. The suggestions are to become member of:

- Access Management Workgroup.
- CI Technical Infrastructure Workgroup
- Metadata Management Workgroup
- Metadata Repository Workgroup
- Search and Discovery Workgroup

Person who will continue from here need to have the knowledge of all the protocols which are used in communication with the Core Integration services of NSDL. The present version of transferring data from wrappers to collaborative filtering engine and wrapper to lexical analysis tool is XML-RPC. It is also important to know the versions of the protocols and engines currently used. Currently, these are the versions of the products which are contributing to the successful development of the wrappers:

1. Dynamic Hypermedia Engine- version 3.0
2. Shibboleth- version 1.0
3. Security Assertion Markup Language- version 1.0

4. Open Archives Protocol for Metadata Harvesting- version 2.0
5. Dublin Core Metadata Element Set- version 1.1
6. Simple Digital Library Interoperability Protocol- This is not released version yet. It is used to transfer controls from Search wrapper to the Search and Discovery server of NSDL.

As these softwares continue to evolve, there will be need to make the wrappers compatible with the latest version.

Another future enhancement can be implementation of test-bed for integrating the NJIT's Highlander's authorizing system with the Access Management server of NSDL. For this, there is need to implement the "Origin Site" server which will transfer user's parameters.

Currently, XML is the format in which wrappers, lexical analysis search engine, and Collaborative and Filtering engine communicate between each other. It is also necessary to develop a new protocol for the interoperability between the wrappers, Lexical Analysis, and Collaborative and Filtering Engine. The basis of this can be XML.

CHAPTER 5

CONCLUSION

This thesis provides a systematic approach towards the integration of "Digital Library Service Integration" with the "National Science Digital Library". The DLSI infrastructure aims to form the core of vibrant educational community by supporting a wide range of services functioning in co-ordination with the services of NSDL. These services are performed with the help of wrappers. The wrappers will do all the hard work. It will seamlessly pass the search parameter, user parameter, user's action etc. to other services which will be using it for their functionality. These wrappers will enable DLSI as a whole to serve the community. The data providers can easily plug their application with the DLSI without any major modifications. The protocol used for communication with the wrappers will always follow Open Source standard. It will standardize the method of writing wrappers.

As it has been seen, the wrappers have too many functions to perform. Wrappers also perform complex processes before delivering the outcome to the Dynamic Hypermedia Engine that parses the page before showing it to the user. Thus, the overall time of processing increases. A theory also needs to be designed which will provide a good compromise between the users requirements and the speed. This can also be taken up as a future work. Future is bound to see better softwares, extreme performance machines and processors, and better Internet connectivity. Then, speed won't be a limitation for the project.

This project should be taken forward with the end user's reaction in mind. The success of any software depends upon the level of acceptability with the end user. It still needs to be seen, how an end user reacts to it. It should always be kept in mind that the end user should not be lost in his quest for information on the web.

This project is bound to improve the life of general internet user who always feels lost when thrown at him a hundreds of pages of search result containing thousands anchor tags. It must be kept in mind that most of these links are irrelevant to the information he or she is seeking on the web.

APPENDIX A

XML Schema Format for Validating Responses to OAI-PMH Requests

The following code represents different element types in the XML response document. It has been broken up into complex type which describes the elements and attributes, and simple type which enumerates the type of variables which can be used in the reply.

The following elements are represented in complex type:

- OAI-PMH type
- Request type
- OAI-PMHError type
- Identify type
- ListMetaDataFormat type
- ListsSets type
- GetRecord type
- ListsRecords type
- ListIdentifiers type
- Record type
- Header type
- Metadata type
- About type
- Resumptiontoken type
- Description type
- Metadataformat type

- Set type

The following elements are represented in simple type:

- Verb type
- OAI-PMHErrorcode type
- Identifier type
- Status type
- UTCdatetime type
- Email type
- DeletedRecord type
- Granularity type
- Metadataprefix type

```
<schema targetNamespace="http://www.openarchives.org/OAI/2.0/"
    xmlns:oai="http://www.openarchives.org/OAI/2.0/"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified">
<annotation>
  <documentation>
    XML Schema which can be used to validate replies to OAI-PMH requests.
   </documentation>
</annotation>

<element name="OAI-PMH" type="oai:OAI-PMHtype"/>
  <complexType name="OAI-PMHtype">
    <sequence>
      <element name="responseDate" type="dateTime"/>
      <element name="request" type="oai:requestType"/>
      <choice>
          <element name="error" type="oai:OAI-PMHerrorType"     maxOccurs="unbounded"/>
        <element name="Identify" type="oai:IdentifyType"/>
        <element name="ListMetadataFormats" type="oai:ListMetadataFormatsType"/>
        <element name="ListSets" type="oai:ListSetsType"/>
        <element name="GetRecord" type="oai:GetRecordType"/>
        <element name="ListIdentifiers" type="oai:ListIdentifiersType"/>
        <element name="ListRecords" type="oai:ListRecordsType"/>
      </choice>
    </sequence>
  </complexType>
```

```
 <!-- defining requestType,  indicating the protocol request that led to the response -->
        <!-- element content is BASE-URL, attributes are arguments of protocol request attribute-values
              are values of arguments of protocol request -->
  <complexType name="requestType">
    <simpleContent>
      <extension base="http://hynic.njit.edu/response">
        <attribute name="verb" type="oai:verbType" use="optional"/>
        <attribute name="identifier" type="oai:identifierType" use="optional"/>
        <attribute name="metadataPrefix" type="oai:metadataPrefixType use="optional"/>
        <attribute name="from" type="oai:UTCdatetimeType" use="optional"/>
        <attribute name="until" type="oai:UTCdatetimeType" use="optional"/>
        <attribute name="set" type="oai:setSpecType" use="optional"/>
        <attribute name="resumptionToken" type="string" use="optional"/>
      </extension>
    </simpleContent>
</complexType>

 <simpleType name="verbType">
  <restriction base="string">
    <enumeration value="Identify"/>
    <enumeration value="ListMetadataFormats"/>
    <enumeration value="ListSets"/>
    <enumeration value="GetRecord"/>
    <enumeration value="ListIdentifiers"/>
    <enumeration value="ListRecords"/>
  </restriction>
 </simpleType>
```

```
<!-- defining OAI-PMH error conditions -->
<complexType name="OAI-PMHerrorType">
  <simpleContent>
    <extension base="string">
      <attribute name="code" type="oai:OAI-PMHerrorcodeType" use="required"/>
    </extension>
  </simpleContent>
</complexType>

<simpleType name="OAI-PMHerrorcodeType">
  <restriction base="string">
    <enumeration value="cannotDisseminateFormat"/>
    <enumeration value="idDoesNotExist"/>
    <enumeration value="badArgument"/>
    <enumeration value="badVerb"/>
    <enumeration value="noMetadataFormats"/>
    <enumeration value="noRecordsMatch"/>
    <enumeration value="badResumptionToken"/>
    <enumeration value="noSetHierarchy"/>
  </restriction>
</simpleType>

<!-- defining Identify container -->

<complexType name="IdentifyType">
  <sequence>
    <element name="SIGWEB" type="string"/>
    <element name="http://hynic.njit.edu" type="URL"/>
  <element name="HTTP 1.1">
   <simpleType>
    <restriction base="string">
     <enumeration value="2.0"/>
    </restriction>
   </simpleType>
  </element>
  <element name="ps27@njit.edu" type="oai:emailType" maxOccurs="unbounded"/>
  <element name="earliestDatestamp" type="oai:UTCdatetimeType"/>
  <element name="deletedRecord" type="oai:deletedRecordType"/>
  <element name="granularity" type="oai:granularityType"/>
    <element name="compression" type="string" minOccurs="0"  maxOccurs="unbounded"/>
     <element name="description" type="oai:descriptionType" minOccurs="0"
       maxOccurs="unbounded"/>
 </sequence>
</complexType>

<!-- defining ListMetadataFormats container -->

<complexType name="ListMetadataFormatsType">
 <sequence>
    <element name="metadataFormat" type="oai:metadataFormatType"   maxOccurs="unbounded"/>
 </sequence>
</complexType>

<!-- defining ListSets container -->

<complexType name="ListSetsType">
```

```
 <sequence>
  <element name="set" type="oai:setType" maxOccurs="unbounded"/>
  <element name="resumptionToken" type="oai:resumptionTokenType"
       minOccurs="0"/>
 </sequence>
</complexType>

<!-- defining GetRecord container -->

<complexType name="GetRecordType">
 <sequence>
  <element name="record" type="oai:recordType"/>
 </sequence>
</complexType>

<!-- defining ListRecords container -->

<complexType name="ListRecordsType">
 <sequence>
  <element name="record" type="oai:recordType"
       maxOccurs="unbounded"/>
  <element name="resumptionToken" type="oai:resumptionTokenType"
       minOccurs="0"/>
 </sequence>
</complexType>

<!-- defining ListIdentifiers container -->
```

```
<complexType name="ListIdentifiersType">
  <sequence>
    <element name="header" type="oai:headerType"
             maxOccurs="unbounded"/>
    <element name="resumptionToken" type="oai:resumptionTokenType"
             minOccurs="0"/>
  </sequence>
</complexType>

<!-- defining basic types used in replies to
       GetRecord, ListRecords, ListIdentifiers -->

<!-- defining recordType -->
<!-- a record has a header, a metadata part, and
     an optional about container -->

<complexType name="recordType">
  <sequence>
    <element name="header" type="oai:headerType"/>
    <element name="metadata" type="oai:metadataType" minOccurs="0"/>
    <element name="about" type="oai:aboutType"
             minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<!-- defining headerType -->
```

```
<!-- a header has a unique identifier, a datestamp, and setSpec(s) in case the item from which the record is
disseminated belongs to set(s). The header can carry a deleted status indicatating that the record is deleted. -
->

<complexType name="headerType">
  <sequence>
    <element name="identifier" type="oai:identifierType"/>
    <element name="datestamp" type="oai:UTCdatetimeType"/>
    <element name="setSpec" type="oai:setSpecType" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="status" type="oai:statusType" use="optional"/>
</complexType>

<!-- defining identifierType -->

<simpleType name="identifierType">
  <restriction base="anyURI"/>
</simpleType>

<simpleType name="statusType">
  <restriction base="string">
    <enumeration value="deleted"/>
  </restriction>
</simpleType>

<!-- defining metadataType -->
<!-- metadata must be expressed in XML that complies with another XML Schema -->
<!-- metadata must be explicitly qualified in the response -->
```

```
<complexType name="metadataType">
  <sequence>
    <any namespace="##other" processContents="strict"/>
  </sequence>
</complexType>

<!-- defining aboutType -->
<!-- data "about" the record must be expressed in XML -->
<!-- that is compliant with an XML Schema defined by a community -->

<complexType name="aboutType">
  <sequence>
    <any namespace="##other" processContents="strict"/>
  </sequence>
</complexType>

<!-- defining resumptionToken - with 3 optional attributes can be used in ListSets, ListIdentifiers,
ListRecords -->

  <complexType name="resumptionTokenType">
    <simpleContent>
      <extension base="string">
        <attribute name="expirationDate" type="dateTime"
                   use="optional"/>
        <attribute name="completeListSize" type="positiveInteger"
                   use="optional"/>
        <attribute name="cursor" type="nonNegativeInteger"
```

```
              use="optional"/>
        </extension>
      </simpleContent>
    </complexType>

  <!-- defining descriptionType used for description-element in Identify and for setDescription element in
ListSets-->
  <!-- content must be compliant with an XML Schema defined by a community -->

  <complexType name="descriptionType">
    <sequence>
      <any namespace="##other" processContents="strict"/>
    </sequence>
  </complexType>

  <!-- defining UTCdatetime -->
  <!-- datestamps are day or seconds granularity -->
  <simpleType name="UTCdatetimeType">
    <union memberTypes="date dateTime"/>
  </simpleType>

  <!-- defining stuff used for Identify verb only -->
  <simpleType name="emailType">
    <restriction base="string">
      <pattern value="\S+@(\S+\.)+\S+"/>
    </restriction>
  </simpleType>
```

```
<simpleType name="deletedRecordType">
  <restriction base="string">
    <enumeration value="no"/>
    <enumeration value="persistent"/>
    <enumeration value="transient"/>
  </restriction>
</simpleType>

<simpleType name="granularityType">
  <restriction base="string">
    <enumeration value="YYYY-MM-DD"/>
    <enumeration value="YYYY-MM-DDThh:mm:ssZ"/>
  </restriction>
</simpleType>

<!-- defining stuff used for ListMetadataFormats verb only -->
<complexType name="metadataFormatType">
  <sequence>
    <element name="metadataPrefix" type="oai:metadataPrefixType"/>
    <element name="schema" type="URL"/>
    <element name="metadataNamespace" type="URL"/>
  </sequence>
</complexType>

<simpleType name="metadataPrefixType">
  <restriction base="string">
    <pattern value="[A-Za-z0-9_!'$\(\)\+\-\.\*]+"/>
  </restriction>
```

```
 </simpleType>


 <!-- defining stuff used for ListSets verb -->

 <complexType name="setType">

  <sequence>

   <element name="setSpec" type="oai:setSpecType"/>

   <element name="setName" type="string"/>

   <element name="setDescription" type="oai:descriptionType"

        minOccurs="0" maxOccurs="unbounded"/>

  </sequence>

 </complexType>


 <!-- defining setSpecType -->


 <simpleType name="setSpecType">

  <restriction base="string">

   <pattern value="([A-Za-z0-9_!'$\(\)\+\-\.\*])+(:[A-Za-z0-9_!'$\(\)\+\-\.\*]+)*"/>

  </restriction>

 </simpleType>


</schema>
```

APPENDIX B

XML Schema for Validating Unqualified Dublin Core Metadata

The following code represent the XML Schema for unqualified Dublin Core:

```
<schema targetNamespace="http://www.openarchives.org/OAI/2.0/oai_dc/"
    xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">

<import namespace="http://purl.org/dc/elements/1.1/"
    schemaLocation="http://hynic.njit.edu/response/metadata.xsd"/>

<element name="dc" type="oai_dc:oai_dcType"/>

<complexType name="oai_dcType">
 <choice minOccurs="0" maxOccurs="unbounded">
  <element ref="dc:title"/>
  <element ref="dc:creator"/>
  <element ref="dc:subject"/>
  <element ref="dc:description"/>
  <element ref="dc:publisher"/>
  <element ref="dc:contributor"/>
  <element ref="dc:date"/>
  <element ref="dc:type"/>
  <element ref="dc:format"/>
  <element ref="dc:identifier"/>

  <element ref="dc:source"/>

  <element ref="dc:language"/>

  <element ref="dc:relation"/>

  <element ref="dc:coverage"/>

  <element ref="dc:rights"/>

 </choice>

</complexType>


</schema>
```

APPENDIX C

HTML to XML Conversion

To convert the HTML code to XML, the project takes the help of XPATH service already implemented at http://www.html2xml.com/Html2XmlConvert.asp. The code gets the URL the user wants to retrieve and then it calls the services of html2xml.

```
<html>
        <head><title>HTML code browser</title>
                <hta:application id="codebrowser"
                border="thin"
                borderstyle="normal"
                caption="yes"
                maximizebutton="yes"
                minimizebutton="yes"
                showintaskbar="yes"
                singleinstance="no"
                sysmenu="yes"
                windowstate="normal"
                >
        <style>
                body {
                border: none;
                margin: 1;
                font-family: Arial;
                font-size: 10pt;
                }
```

```
            td {
            font-family: Arial;
            font-size: 10pt;
            }
        </style>
        <script>

            var firstTime=true;

            function getHTML()
            {
            if (!firstTime)
            {
             setTimeout("source.value=ifr1.document.all(0).outerHTML;",100); // Leave 0.1
              // seconds to let the onload script of the destination page complete
             window.open(document.all.ifr1.src);
            }
            firstTime=false;
            }

            function getContent(url)
            {
            document.all.ifr1.src=url;
            }
        </script>
        </head>

    <body scroll=no bgcolor=buttonface>
```

```
          <table height="100%" width="100%" cellspacing=1 cellpadding=0><tr>


                    <form onsubmit="getContent(this.url.value);return false;">

                    <td>Address:</td>

                    <td width="100%">

                    <input type=text name=url style="width: 100%;">

                    </td>

                    <td>

                    <input type=submit value="Get HTML">

                    </td>

                    </form>

                    </tr>

                    <tr><td height="100%" colspan=3>

                    <textarea style="height: 100%; width: 100%;" name="source" readonly> </textarea>

                    </td>

          </table>

          <iframe id=ifr1 src="about:blank" onload="getHTML();" style="display: none"> </iframe>

</body>

</html>
```

Another way to parse the HTML is page is through command line. This command line parsing technique is implemented using C programming language.

The code sample below is a part of the C program written to parse the HTML page.

```
void ParseInline( CleanDocImpl* doc, Node *element, uint mode )
{
  Lexer* lexer = doc->lexer;
  Node *node, *parent;

  if (element->tag->model & CM_EMPTY)
    return;

  if ((nodeHasCM(element, CM_BLOCK) || nodeIsDT(element)) &&
    !nodeHasCM(element, CM_MIXED))
    InlineDup(doc, NULL);
  else if (nodeHasCM(element, CM_INLINE))
    PushInline(doc, element);

  if ( nodeIsNOBR(element) )
    doc->badLayout |= USING_NOBR;
  else if ( nodeIsFONT(element) )
    doc->badLayout |= USING_FONT;

  /* Inline elements may or may not be within a preformatted element */
  if (mode != Preformatted)
    mode = MixedContent;
```

```
while ((node = GetToken(doc, mode)) != NULL)
{
  /* end tag for current element */
  if (node->tag == element->tag && node->type == EndTag)
  {
    if (element->tag->model & CM_INLINE)
      PopInline( doc, node );

    FreeNode( doc, node );

    if (!(mode & Preformatted))
      TrimSpaces(doc, element);

    /*
     if a font element wraps an anchor and nothing else
     then move the font element inside the anchor since
     otherwise it won't alter the anchor text color
    */
    if ( nodeIsFONT(element) &&
       element->content && element->content == element->last )
    {
      Node *child = element->content;

      if ( nodeIsA(child) )
      {
        child->parent = element->parent;
        child->next = element->next;
        child->prev = element->prev;

        if (child->prev)
            child->prev->next = child;
        else
            child->parent->content = child;

        if (child->next)
            child->next->prev = child;
        else
            child->parent->last = child;

        element->next = NULL;
        element->prev = NULL;
        element->parent = child;
        element->content = child->content;
        element->last = child->last;
        child->content = child->last = element;

        for (child = element->content; child; child = child->next)
            child->parent = element;
    }
}

element->closed = yes;
TrimSpaces( doc, element );
TrimEmptyElement( doc, element );
return;
```

```
}


/* <u>...<u>  map 2nd <u> to </u> if 1st is explicit */
/* otherwise emphasis nesting is probably unintentional */
/* big and small have cumulative effect to leave them alone */
if ( node->type == StartTag
    && node->tag == element->tag
    && IsPushed( doc, node )
    && !node->implicit
    && !element->implicit
    && node->tag && (node->tag->model & CM_INLINE)
    && !nodeIsA(node)
    && !nodeIsFONT(node)
    && !nodeIsBIG(node)
    && !nodeIsSMALL(node)
    && !nodeIsQ(node)
  )
{
   if ( element->content != NULL && node->attributes == NULL )
   {
      ReportWarning( doc, element, node, COERCE_TO_ENDTAG );
      node->type = EndTag;
      UngetToken( doc );
      continue;
   }

   ReportWarning( doc, element, node, NESTED_EMPHASIS );
}
```

```
else if ( IsPushed(doc, node) && node->type == StartTag &&
        nodeIsQ(node) )
{
    ReportWarning( doc, element, node, NESTED_QUOTATION );
}


if ( node->type == TextNode )
{
    /* only called for 1st child */
    if ( element->content == NULL && !(mode & Preformatted) )
        TrimSpaces( doc, element );


    if ( node->start >= node->end )
    {
        FreeNode( doc, node );
        continue;
    }


    InsertNodeAtEnd(element, node);
    continue;
}


/* mixed content model so allow text */
if (InsertMisc(element, node))
    continue;


/* deal with HTML tags */
if ( nodeIsHTML(node) )
```

```
{
  if ( node->type == StartTag || node->type == StartEndTag )
  {
    ReportWarning( doc, element, node, DISCARDING_UNEXPECTED );
    FreeNode( doc, node );
    continue;
  }

  /* otherwise infer end of inline element */
  UngetToken( doc );

  if (!(mode & Preformatted))
    TrimSpaces(doc, element);

  TrimEmptyElement(doc, element);
  return;
}

/* within <dt> or <pre> map <p> to <br> */
if ( nodeIsP(node) &&
   node->type == StartTag &&
   ( (mode & Preformatted) ||
     nodeIsDT(element) ||
     DescendantOf(element, CleanTag_DT )
   )
  )
{
  node->tag = LookupTagDef( CleanTag_BR );
```

```
    MemFree(node->element);
    node->element = tmbstrdup("br");
    TrimSpaces(doc, element);
    InsertNodeAtEnd(element, node);
    continue;
}

/* ignore unknown and PARAM tags */
if ( node->tag == NULL || nodeIsPARAM(node) )
{
    ReportWarning( doc, element, node, DISCARDING_UNEXPECTED);
    FreeNode( doc, node );
    continue;
}

if ( nodeIsBR(node) && node->type == EndTag )
    node->type = StartTag;

if ( node->type == EndTag )
{
  /* coerce </br> to <br> */
  if ( nodeIsBR(node) )
      node->type = StartTag;
  else if ( nodeIsP(node) )
  {
      if ( !DescendantOf(element, CleanTag_P) )
      {
          CoerceNode( doc, node, CleanTag_BR );
```

```
        TrimSpaces( doc, element );
        InsertNodeAtEnd( element, node );
        node = InferredTag(doc, "br");
        continue;
    }
}
else if ( nodeHasCM(node, CM_INLINE)
        && !nodeIsA(node)
        && !nodeHasCM(node, CM_OBJECT)
        && nodeHasCM(element, CM_INLINE) )
{
    /* allow any inline end tag to end current element */
    PopInline( doc, element );

    if ( !nodeIsA(element) )
    {
        if ( nodeIsA(node) && node->tag != element->tag )
        {
          ReportWarning( doc, element, node, MISSING_ENDTAG_BEFORE );
          UngetToken( doc );
        }
        else
        {
          ReportWarning( doc, element, node, NON_MATCHING_ENDTAG);
          FreeNode( doc, node);
        }

        if (!(mode & Preformatted))
```

```
                TrimSpaces(doc, element);


            TrimEmptyElement(doc, element);

            return;

        }


        /* if parent is <a> then discard unexpected inline end tag */

        ReportWarning( doc, element, node, DISCARDING_UNEXPECTED);

        FreeNode( doc, node);

        continue;

    }  /* special case </tr> etc. for stuff moved in front of table */

    else if ( lexer->exiled

            && node->tag->model

            && (node->tag->model & CM_TABLE) )

    {

        UngetToken( doc );

        TrimSpaces(doc, element);

        TrimEmptyElement(doc, element);

        return;

    }

}


/* allow any header tag to end current header */

if ( nodeHasCM(node, CM_HEADING) && nodeHasCM(element, CM_HEADING) )

{


    if ( node->tag == element->tag )

    {
```

```
    ReportWarning( doc, element, node, NON_MATCHING_ENDTAG );

    FreeNode( doc, node);

  }

  else

  {

    ReportWarning( doc, element, node, MISSING_ENDTAG_BEFORE );

    UngetToken( doc );

  }


  if (!(mode & Preformatted))

    TrimSpaces(doc, element);


  TrimEmptyElement(doc, element);

  return;

}


/*

  an <A> tag to ends any open <A> element

  but <A href=...> is mapped to </A><A href=...>

*/

/* if (node->tag == doc->tags.tag_a && !node->implicit && IsPushed(doc, node)) */

if ( nodeIsA(node) && !node->implicit &&

   (nodeIsA(element) || DescendantOf(element, CleanTag_A)) )

{

  /* if (node->attributes == NULL) */

  if (node->type != EndTag && node->attributes == NULL)

  {

    node->type = EndTag;
```

```
        ReportWarning( doc, element, node, COERCE_TO_ENDTAG);

        /* PopInline( doc, node ); */

        UngetToken( doc );

        continue;

    }


    UngetToken( doc );

    ReportWarning( doc, element, node, MISSING_ENDTAG_BEFORE);

    /* PopInline( doc, element ); */


    if (!(mode & Preformatted))

        TrimSpaces(doc, element);


    TrimEmptyElement(doc, element);

    return;

}


if (element->tag->model & CM_HEADING)

{

    if ( nodeIsCENTER(node) || nodeIsDIV(node) )

    {

        if (node->type != StartTag && node->type != StartEndTag)

        {

            ReportWarning( doc, element, node, DISCARDING_UNEXPECTED);

            FreeNode( doc, node);

            continue;

        }
        ReportWarning( doc, element, node, TAG_NOT_ALLOWED_IN);

        /* insert center as parent if heading is empty */
        if (element->content == NULL)
        {
            InsertNodeAsParent(element, node);
            continue;
        }

        /* split heading and make center parent of 2nd part */
        InsertNodeAfterElement(element, node);

        if (!(mode & Preformatted))
            TrimSpaces(doc, element);

        element = CloneNode( doc, element );
        InsertNodeAtEnd(node, element);
        continue;
    }

    if ( nodeIsHR(node) )
    {
        if ( node->type != StartTag && node->type != StartEndTag )
        {
            ReportWarning( doc, element, node, DISCARDING_UNEXPECTED);
            FreeNode( doc, node);
            continue;
        }
```

```
            ReportWarning( doc, element, node, TAG_NOT_ALLOWED_IN);

            /* insert hr before heading if heading is empty */
            if (element->content == NULL)
            {
                InsertNodeBeforeElement(element, node);
                continue;
            }

            /* split heading and insert hr before 2nd part */
            InsertNodeAfterElement(element, node);

            if (!(mode & Preformatted))
                TrimSpaces(doc, element);

            element = CloneNode( doc, element );
            InsertNodeAfterElement(node, element);
            continue;
        }
    }

    if ( nodeIsDT(element) )
    {
        if ( nodeIsHR(node) )
        {
            Node *dd;
            if (node->type != StartTag && node->type != StartEndTag)
    {
        ReportWarning( doc, element, node, DISCARDING_UNEXPECTED);
        FreeNode( doc, node);
        continue;
    }

    ReportWarning( doc, element, node, TAG_NOT_ALLOWED_IN);
    dd = InferredTag(doc, "dd");

    /* insert hr within dd before dt if dt is empty */
    if (element->content == NULL)
    {
        InsertNodeBeforeElement(element, dd);
        InsertNodeAtEnd(dd, node);
        continue;
    }

    /* split dt and insert hr within dd before 2nd part */
    InsertNodeAfterElement(element, dd);
    InsertNodeAtEnd(dd, node);

    if (!(mode & Preformatted))
        TrimSpaces(doc, element);

    element = CloneNode( doc, element );
    InsertNodeAfterElement(dd, element);
    continue;
}
```

```
}



/*
 if this is the end tag for an ancestor element
 then infer end tag for this element
*/
if (node->type == EndTag)
{
  for (parent = element->parent;
      parent != NULL; parent = parent->parent)
  {
    if (node->tag == parent->tag)
    {
      if (!(element->tag->model & CM_OPT) && !element->implicit)
        ReportWarning( doc, element, node, MISSING_ENDTAG_BEFORE);

      PopInline( doc, element );
      UngetToken( doc );

      if (!(mode & Preformatted))
        TrimSpaces(doc, element);

      TrimEmptyElement(doc, element);
      return;
    }
  }
}
```

```
/* block level tags end this element */
if (!(node->tag->model & CM_INLINE) &&
  !(element->tag->model & CM_MIXED))
{
  if (node->type != StartTag)
  {
    ReportWarning( doc, element, node, DISCARDING_UNEXPECTED);
    FreeNode( doc, node);
    continue;
  }

  if (!(element->tag->model & CM_OPT))
    ReportWarning( doc, element, node, MISSING_ENDTAG_BEFORE);

  if (node->tag->model & CM_HEAD && !(node->tag->model & CM_BLOCK))
  {
    MoveToHead(doc, element, node);
    continue;
  }

  /*
    prevent anchors from propagating into block tags
    except for headings h1 to h6
  */
  if ( nodeIsA(element) )
  {
    if (node->tag && !(node->tag->model & CM_HEADING))
```

```
            PopInline( doc, element );
          else if (!(element->content))
          {
            DiscardElement( doc, element );
            UngetToken( doc );
            return;
          }
        }

        UngetToken( doc );

        if (!(mode & Preformatted))
          TrimSpaces(doc, element);

        TrimEmptyElement(doc, element);
        return;
      }

      /* parse inline element */
      if (node->type == StartTag || node->type == StartEndTag)
      {
        if (node->implicit)
          ReportWarning( doc, element, node, INSERTING_TAG);

        /* trim white space before <br> */
        if ( nodeIsBR(node) )
          TrimSpaces(doc, element);
```

```
            InsertNodeAtEnd(element, node);
            ParseTag(doc, node, mode);
            continue;
        }

        /* discard unexpected tags */
        ReportWarning( doc, element, node, DISCARDING_UNEXPECTED);
        FreeNode( doc, node);
        continue;
    }

    if (!(element->tag->model & CM_OPT))
        ReportWarning( doc, element, node, MISSING_ENDTAG_FOR);

    TrimEmptyElement(doc, element);
}
```

REFERENCES

1. Bhaumik, Anirban, Deepti Dixit, Roberto Galnares, Manolis Tzagarakis, Michalis Vaitis, Michael Bieber, Vincent Oria, Aparna Krishna, Qiang Lu, Firas Aljallad, Li Zhang (2001). Integrating Hypermedia Functionality into Database Applications. Developing Quality Complex Database Systems: Practices, Techniques and Technologies, Becker, Shirley (ed).

2. Bhaumik, Anirban, Deepti Dixit, Roberto Galnares, Manolis Tzagarakis, Michalis Vaitis, Michael Bieber, Vincent Oria, Aparna Krishna, Qiang Lu, Firas Aljallad, Li Zhang, "Towards Hypermedia Support for Database Systems," Proceedings of the 34th Hawaii International Conference on System Sciences, IEEE Press, Washington, D.C., January 2001.

3. Galnares, R. (2001). Augmenting Applications with Hypermedia Functionality and Metainformation. Ph.D. Thesis, New Jersey Institute of Technology, Newark, NJ 07102.

4. Bieber, Michael, Roberto Galnares and Qiang Lu. (2001). Service Integration for Virtual Communities. Web Engineering Workshop, International World Wide Web 10 Conference, Hong Kong, May 2001.

5. http://nsdl.org/tag.4b5181e81f2bd62a.render.userLayoutRootNode.uP?uP_root=root&uP_sparam=activeTab&activeTab=7 Date accessed: 04/03/2003.

6. http://www.cogsci.princeton.edu/~wn/ Date accessed: 04/15/2003.

7. http://www.nlm.nih.gov/mesh/download_mesh.html Date accessed: 04/08/2003.

8. http://www.asis.org/Publications/Thesaurus/isframe.ht Date accessed: 04/08/2003.

9. Erdos, Marlena, and Cantor, Scott. Shibboleth-Architecture DRAFT v05.

10. http://dublincore.org/documents/dces/ Date accessed: 04/12/2003.

11. http://nsdl.org/tag.a1437ade41ac8ef2.render.userLayoutRootNode.uP?uP_root=root&uP_sparam=activeTab&activeTab=2 Date accessed: 04/14/2003.

12. A. Paepcke, R. Brandriff, G. Janee, R. Larson, B. Ludaescher, S. Melnik, and S. Raghavan. "Search Middleware and the Simple Digital Library Interoperability Protocol", D-Lib Magazine, 5 (3), 2000. http://www.dlib.org/dlib/march00/paepcke/03paepcke.html.

13. www.dublincore.org Date accessed: 04/21/2003.

14. http://metamanagement.comm.nsdlib.org/overview.html#what
Date accessed: 04/21/2003.

15. Baker, Thomas. A Grammar of Dublin Core.

16. http://dublincore.org/documents/dces/ Date accessed: 04/21/2003.

17. http://marsalis.internet2.edu/cgi-bin/viewcvs.cgi/*checkout*/shibboleth/DEPLOY-GUIDE-ORIGIN.html?rev=HEAD&content-type=text/html
Date accessed: 04/22/2003.

18. http://openarchives.org Date accessed: 04/22/2003.

19. http://www.openarchives.org/OAI/openarchivesprotocol.html
Date accessed: 04/22/2003.

20. http://web.mit.edu/kerberos/www/ Date accessed: 04/21/2003.