# Digital Library Software Architectures: TeachEngineering Case Study

**Venkata Satya Gokul Suryadevara**
**suryadgo@eecs.orst.edu**

Major Professor: Dr. Bella Bose
Committee Member: Dr. Rene Reitsma
Committee Member: Dr. Rajeev Pandey

Department of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331

05/31/2005

# Acknowledgements

I would like to express my sincere appreciation to my major professor, Dr. Bella Bose for his constant support and encouragement. I would also like to thank Dr. Rene Reitsma for his valuable ideas and constructive criticism. His encouragement, support and guidance molded me into a worthy graduate student. I would like to express my sincere gratitude to Dr. Rajeev Pandey, School of Electrical Engineering and Computer Science for his continuous encouragement and concern. I would like to thank the TeachEngineering Digital Library Development Team for their constant support and feedback. On a personal note I would like to acknowledge the support and love of my family and friends. Last but not the least I would like to thank almighty God for showering his blessings on me throughout my life.

# Abstract

Digital libraries are digitally accessible, organized collections of knowledge. Although under this broad definition any digitally accessible data set might be considered a digital library, the term is generally reserved for collections whose structures are carefully documented and made available in the form of so-called *metadata*. There is no specific approach to build a digital library as builders typically use their own architectures which serves only their purpose and satisfies their target audience. These approaches are sometimes hard to understand. In this project, I'm presenting a Best Practices for Digital Library Software Architectures after a careful study of existing architectures and comparing these architectures with the *TeachEngineering* Digital Library Architecture.

# Table of Contents


**ABSTRACT** ..... **3**

**INTRODUCTION** ..... **5**

***TEACHENGINEERING* DIGITAL LIBRARY** ..... **6**

OBJECTIVES ..... 6
PARTNERS ..... 7
DIGITAL LIBRARIES AND NATIONAL SCIENCE DIGITAL LIBRARY ..... 7

**ARCHITECTURES** ..... **9**

CHALLENGE ..... 9
DOCUMENTED SOLUTIONS ..... 9
*DSpace Institutional Digital Repository System* ..... *9*
*The Alexandria Digital Library* ..... *11*
*UpLib: A Universal Personal Digital Library System* ..... *12*
*Tufts Digital Library* ..... *13*
*Columbia University Digital Library* ..... *14*
*National Digital Library, Portugal* ..... *15*
*Common Components* ..... *16*
User Interface ..... 16
Document Repository ..... 17
Search Engine (Indexing and Intelligent Data Retrieval) ..... 17
Metadata and Automatic Metadata Generation ..... 18
Protocols for Client/Server Information Retrieval (Z39.50 and OAI-PMH) ..... 18

***TEACHENGINEERING* ARCHITECTURE** ..... **19**

DOCUMENT TEMPLATES AND AUTHORING ..... 19
DOCUMENT REPOSITORY ..... 19
DOCUMENT SPIDER ..... 19
METADATA GENERATOR ..... 19
RENDERING ENGINE ..... 20
WEB SERVICE ..... 20
USER INTERFACE ..... 21
STEP-BY-STEP DEVELOPMENT OF TEACHENGINEERING DIGITAL COLLECTION DOCUMENT ..... 21
METADATA ..... 22
*Protocols for Client/Server Information Retrieval (Z39.50 and OAI-PMH)* ..... *23*
Z39.50 ..... 23
Open Archive Initiative Protocol for Metadata Harvesting ..... 25

**STATE OF TE** ..... **28**

**TE ARCHITECTURE EVALUATION** ..... **29**

**CONCLUSION** ..... **31**

**REFERENCES** ..... **32**

# Introduction

A digital library is an integrated set of services for capturing, cataloguing, storing, searching, protecting and retrieving digital information. Critical data sometimes become incoherent, unavailable, corrupted and lost. Therefore, it is necessary for the data to be stored and retrieved digitally. The National Science Foundation identified a strong need for a digital library that can provide high quality resources and tools that support innovations in teaching and learning in science, technology, engineering and mathematics education. The National Science Digital Library was born to serve this purpose. *TeachEngineering* is one of several collections of The National Science Digital Library.

Future communities should be able, through the medium of properly designed digital libraries to gain access to various forms of knowledge from anywhere and at anytime in an efficient and user-friendly manner. This can be accomplished if all the digital libraries use a common infrastructure which is highly scalable, customizable and adaptable. It is very difficult to make this happen as the digital libraries have different digital objects. Therefore, digital libraries started developing their content using architectures that are not very easily customizable. In this project, I studied several digital library architectures available and identified the common components which every digital library possesses, compared them with *TeachEngineering* and prepared a Best Practices Guide for digital library software architectures.

This paper is organized into five main sections. Section 1 gives a detailed introduction to the *TeachEngineering* Digital Library, its objectives and the involved partners, and the initiative for building the National Science Digital Library. In section 2, the architectures of different digital libraries are described and the common components are identified. Section 3 gives a picture of the *TeachEngineering* Digital Library and its architecture, and its components. In section 4, the current state of *TeachEngineering* is discussed and section 5 describes the *TeachEngineering* architecture evaluation.

## *TeachEngineering* Digital Library

### *Objectives*

*TeachEngineering* is a digital collection project by several engineering colleges collaborating to create an on-line digital library of engineering resources (the *TeachEngineering* Collection) for use by K-12 teachers. The collection is built from extensive K-12 engineering curriculum developments funded by the NSF GK-12 program. Each institution partnered with numerous local school districts to promote engineering as a vehicle for math and science integration. Lessons and activities that introduce engineering to K-12 students while serving as integrators of science and mathematics concepts populate the collection.

The *TeachEngineering* curriculum collection is broadly classified into 13 subject areas. "The lessons and activities in the TeachEngineering Collection relate to everyday encounters in the lives of youngsters, thus providing a context for student learning."[15]. Curriculum content developed by the curriculum developers is aligned with national science, mathematics and technology educational standards. Lessons are associated with activities that can be constructed at low cost with readily available materials. These activities make learning interesting and exciting. The collection also provides a web portal to several "living laboratories" -- structures, facilities and processes instrumented with sensors, providing data on-line in real time. These living labs provide students and teachers access to real systems and real data. Students can use the living lab data to design engineering solutions to water, energy and transportation problems. The project team also is reaching out to end-users by promoting workshops that train teachers and faculty to use the Collection. "The American Society for Engineering Education involvement guarantees long-term sustainability, with responsibility for certification and testing of new curricular components, and nationwide dissemination and promotion of the Collection."[15]. The Collection content is being standardized, converting a variety of K-12 engineering curricula into searchable, standards-based documents with a common look and feel.

Moving K-12 engineering outreach curricula from individual sites to a unified and useful library provides accessible resources for the K-12 community and stimulates the involvement of engineering faculty and professionals in K-12 education. This work is a trendsetter as it is engaging more engineering programs in K-12, which can support STEM (Science, Technology, Engineering, and Mathematics) education. The team will reach end-users by conducting workshops to train teachers develop and use *TeachEngineering* collection. In these workshops, they will be taught to fit their curriculum into *TeachEngineering* form so that it can be placed immediately into the *TeachEngineering* repository.

### *Partners*

The *TeachEngineering* team includes university professors, staff and students from five universities.

- University of Colorado at Boulder
- Worcester Polytechnic Institute
- Colorado School of Mines
- Duke University and
- Oregon State University

## *Digital Libraries and the National Science Digital Library*

A digital library comprises digital collections, services and infrastructure to support lifelong learning, research, communication and preservation. A digital library is like a traditional library which has a collection of books and references. Unlike a traditional library however, materials are in digital form and usually served over the World Wide Web. There is a lot of potential for digital libraries to store large amounts of data without any storage space constraints.

The National Science Foundation developed the idea of stimulating and sustaining improvements in the quality of science, technology, engineering, and mathematics (STEM) education at all levels. They thought that this could be achieved by building a Digital Library that targets a broad audience – pre-K to 12, undergraduate, graduate, and life-long learners – in formal and informal settings, and individual and collaborative modes. This is when the National Science Digital Library (NSDL) was born and started to be a virtual facility. As a "national treasure" it will enable seamless access to a rich array of interactive learning resources and learning environments, and services not bound by place or time.

The fundamental operational mode of the library consists of an interaction of users, content, and tools via the network that connects these three elements. Users comprise students, educators, and life-long learners. "Content describes a rich and diverse set of materials in multiple media, including structured learning materials; large real-time or archived data sets; audio, video, images, and animations; "born" digital learning objects (e.g. simulations and applets); interactive (virtual or remote) laboratories; and primary source material" [16]. Tools are made highly functional so that users can gain access to the content by searching, referring, validating, integrating, annotating, creating, customizing, sharing, publishing, notification, and collaboration.

Hence, the NSF has studied the development of a national digital library for science, technology, engineering and mathematics (STEM) education and started funding the National STEM Education Digital Library (NSDL) Initiative, "a collective effort to build a national digital library of high-quality educational materials for students and teachers at all levels, in both formal and informal settings" [16]. *TeachEngineering* comes under the umbrella of the National Science Digital Library (NSDL), which to date has monitored some 70 digital collection projects in the past few years. There is no common framework to build the collections of National Science Digital Library but the library specified rules so that linking all the collections would be effective and easy. At present, NSDL is a collection of other digital library collections. All the content in the library comes from several collections it is hosting.

*TeachEngineering* is a searchable, web-based digital library populated with standards-based K-12 math and science curricula for use by engineering faculty and K-12 instructors to teach engineering in K-12 settings. The *TeachEngineering* web site provides K-12 educators with access to a growing curricular collection, as the contents evolve under the stewardship of the American Society for Engineering Education. Development of *TeachEngineering* is funded by NSF-NSDL award# 0226322.

The architecture, document collection and metadata formulation are based on established NSDL digital library protocols. Of the 447 collections currently registered to NSDL, only about 50 or so have been directly funded by the NSDL program.

# Architectures

## *Challenge*

There is no standard software architecture for developing a digital library. Architectures are made *ad hoc* and it is really getting difficult to understand the work flow of any digital library. In this section, I'm presenting my study on existing architectures and identified some components that are found common in all the existing architectures. Most of the digital libraries didn't publish any information on their architecture and hence this study is limited to the information I found on the web, journals and various conference proceedings.

## *Documented Solutions*

### *DSpace* Institutional Digital Repository System

"DSpace is an open source system that acts as a repository for digital research and educational material produced by an organization or institution" [6]. DSpace is designed to operate as a centralized, institutional service. Different communities within the institution such as labs, centers, schools, or departments can have their own separate areas within the system. 'Gatekeepers' are appointed by the communities to monitor and moderate the content before their inclusion in the main repository. The functional aspects of DSpace are listed below and described in Figure 1.

- *Data Model.* DSpace defines a data model for basic organization of data. Each DSpace site is divided into communities; these typically correspond to a laboratory, research center or department. Communities contain collections which are groups of related content. Collections are composed of items and items are further subdivided into bundles of bit streams, which are ordinary computer files.

- *Metadata.* DSpace stores various types of metadata in the system. All the metadata are replicated in Dublin Core records so that it is easily accessible outside DSpace, for example via the OAI (Open Archive Initiative) protocol.

- *User Personal Area.* DSpace's personal user area is called '*E-Personal*'. It holds information about users, authentication and access permissions.

- *Data Submission.* The DSpace system accepts incoming material in a process called '*ingestion.*' Metadata verification is done before its final submission which is called '*workflow.*' One or more human reviewers or gatekeepers oversee the submission and ensure it is suitable for inclusion into the collection. In order for archived material to be cited and accessed using information in a citation, a

'*handle system*' is used to assign globally unique and persistent identifiers to archived items. DSpace also allows end-users to discover content via external reference such as a handle or searching for one or more keywords or browsing through the title, date and author indices.

- *Searching* and *Browsing* is an essential component of discovery in DSpace. It uses a simple API which facilitates with indexing the content, regenerating indexes, and performing searches on entire collections. Search Engine is built in Java and called '*Lucene*'. *Lucene* is an open source search engine library. It is used in full-text searches on cross-platforms. DSpace uses another API for browsing the content. This API allows the user to specify an index and a subsection of that index. DSpace exposes the Dublin Core metadata for items that are publicly accessible. Additionally, the collection structure is also exposed via Open Archive Initiative Protocol for Metadata Harvesting's 'set' mechanism. The web interface of DSpace is built on Java Servlet and JSP technology (Model 1/MVC pattern). The UI supports on-line help, collection home pages configurable by individual communities, searching and browsing facility, and 'My DSpace' area. There is an administrative sub-section and UI supports multiple browsers.

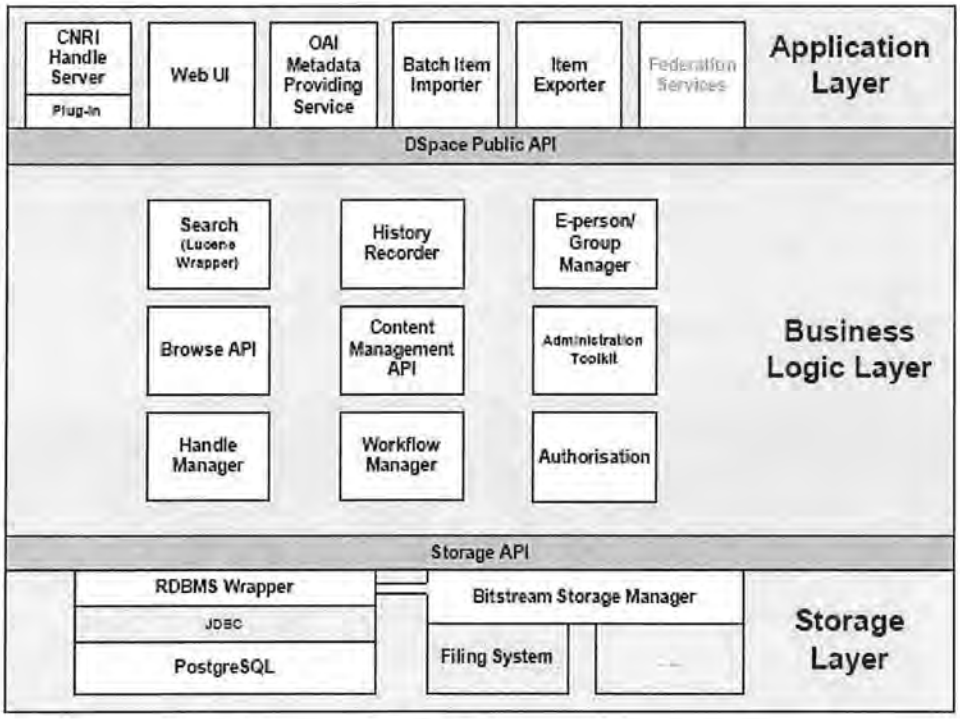


**Figure 1: DSpace Architecture. Source: [6].**

## The Alexandria Digital Library

"The Alexandria Project is a consortium of researchers, developers, and educators, spanning the academic, public, and private sectors, exploring a variety of problems related to distributed digital libraries for georeferenced information." [5]. Figure 2 describes architecture of the Alexandria Digital Library.

The elegant piece of the architecture is the middleware layer, which maps various information servers into fewer standard client interfaces for metadata queries, metadata retrieval and digital library repository. The client is a graphical interface designed to support interactive queries posed by the users. The client version is Java based and distributed as a self installing standalone application for any platform.

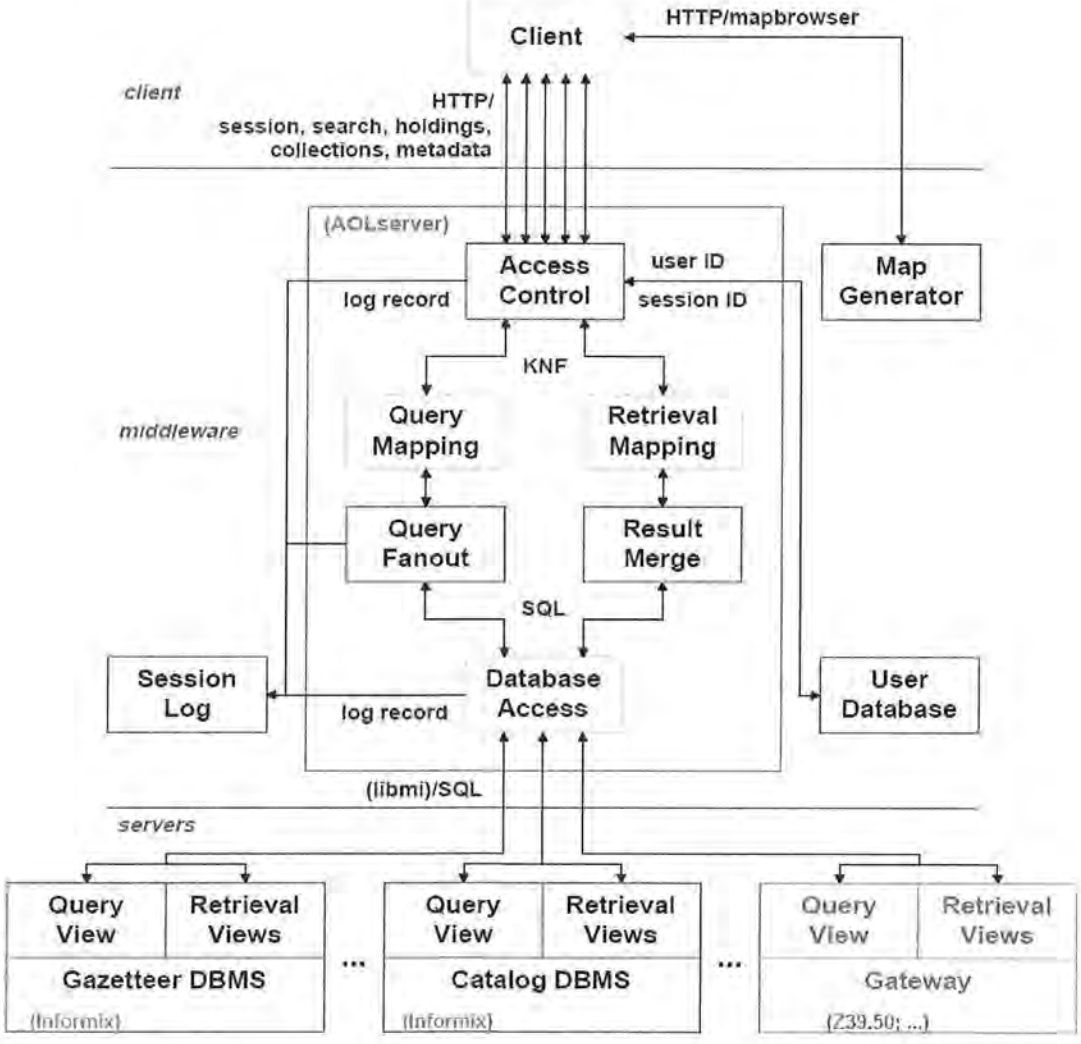


**Figure 2: The Alexandria Digital Library Architecture. Source: [5]**

### *UpLib*: A Universal Personal Digital Library System

"The Universal Personal Digital library project addresses the capture, secure storage, organization, access and use of documents involved in a person's day-to-day activities"[3]. This digital library is designed to be scalable as thousands of documents can be accommodated without reducing the system's responsiveness. Security is also maintained as the documents may be personal medical and financial records.

- *Documents* are captured using digital cameras, scanners for paper documents and document converters for electronic documents.

- *Document Repository.* The users specify a directory that will be used as a root of a particular *document repository*. This allows the repository storage to be integrated with the user's normal backup procedures, and allows direct access to the document storage. The new document folder is passed to the UpLib system, it is assigned a unique name based on current time and placed in an area where normalization takes place. This involves producing other versions of the document and generating more metadata about the document.

- *User Interfaces.* There are two major interfaces to the repository: a visual presentation of all the documents and a full-text search over textual and metadata projection spaces.

- *Metadata* about the whole repository are stored in a top level directory of the repository in standard text format or XML format.

- *Search Engine. Lucene* is used for full-text indexing system which serves the searching facility on UpLib.

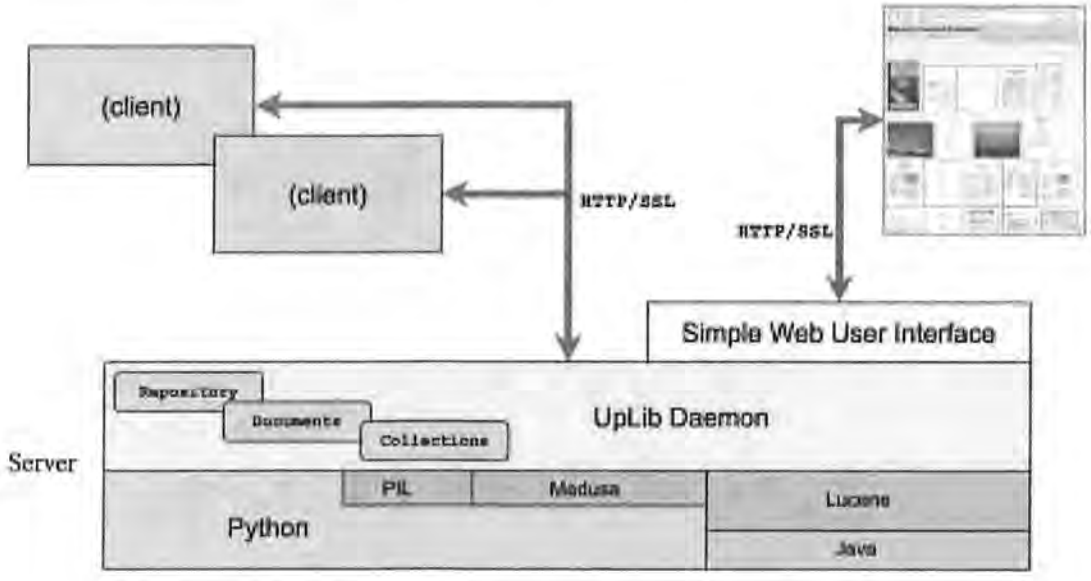


**Figure 3: UpLib Architecture. Source: [3]**

## Tufts Digital Library

Tufts Digital Library is an extensible, modular, flexible and scalable architecture that uses Fedora as its core base. "TDL is designed to allow assimilation and interoperability of existing Tufts Digital Libraries while allowing new creators of digital material to add their content and write new applications for using and managing material" [2]. The architecture is comprised of several services: Drop box and ingestion service, naming service, fedora repository service, indexing and search service and application creation service. There is a drop box facility where the content developers drop their content. The drop box contains a template file provided by the archivists that has basic metadata associated with all the objects. Figure 4 describes the architecture.

- *Document Submission.* The ingestion service automatically collects the objects from the drop box. It validates against the Fedora object schema and waits for the collection developers or 'Gatekeepers' who will moderate the content, look for quality before approving or rejecting the objects based on the digital collection standards.

- *Document Repository.* The Fedora repository service forms the core of the document repository. This service support a variety of data types and accommodates new types as they emerge. It also performs aggregation of mixed, distributed data into complex objects. Images, XML documents, HTML documents, binary files and maps are some of the object types. Each object in the repository is identified with a particular content-type.

- *Metadata.* This library uses the Dublin Core metadata set for storing fields such as author, title, subject etc. Metadata are also acquired and managed as 'data streams.'

- *Search and Indexing.* The Tufts Digital Library has an indexing and search service for accessing its content. *Lucene* is an open source search engine that provides full-text based search, metadata search and advanced search features

- *User Interface.* There is a user interface through which users can access the content.

There is an application creation service which allows current digital library applications to easily interface with the Tufts Digital Library and provide access to the content in the digital library within their own environments.

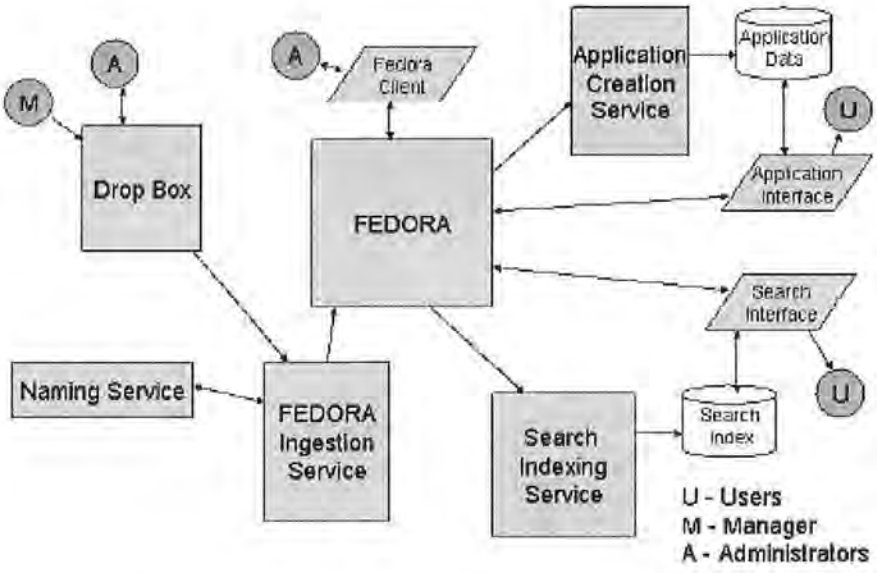


**Figure 4: Tufts Digital Library Architecture. Source: [2]**

## Columbia University Digital Library

"Columbia University's digital library has discrete layers of technology services and each successive layer depends upon the services of the previous layers." [8]. Figure 5 describes the architecture of the Columbia University Digital Library.

- *Document Repository.* All the data are stored in a place called the Repository. This repository lives on a network file system and has tape backup. This library is also coming up with a plan to sign the submission of documents into the repository cryptographically. They introduced a new Identifier to identify the digital objects in the repository.

- *Network protocols.* Web protocols are used to access most of the digital collections, with distinct items referenced directly by web servers using a simple UNIX file system.

- *User Personal Area.* Authentication services will take care of access rights for specified areas. Columbia University is currently working on a method to grant access to external organizations to use their content.

- *Search and Retrieval.* Columbia University is using *Ultraseek* for full-text retrieval. This search and indexing system is fast, precise and popular.

- *Metadata.* They developed a Master Metadata File facility which is relational database application holding bibliographic and structural information. Information

can be exported and imported into several formats and can perform interactive queries on the database.

- *User Interface*. A framework was developed in which content may be used in as many ways, by as many different interfaces, as possible which is a very interesting aspect of this digital library.

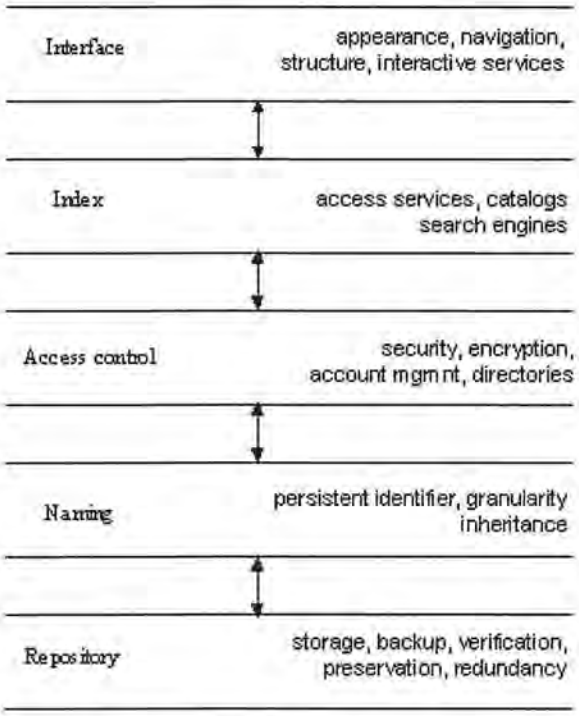


**Figure 5: Columbia University Digital Library Architecture: Source: [8].**

## National Digital Library, Portugal

"The National Digital Library of Portugal is built to address new challenges concerning digitization, digital publishing, digital preservation and, creation, processing, quality control and exchange of metadata, services for resource discovery, access and interoperability" [7]. The architecture of this digital library has deposit, registration, storage, search, retrieval, access and preservation of digital resources as main constituents. There is a separate framework comprising the services and interfaces designed for the integration of the other components.

- *Document Repository*. There are two repositories: Repository-A: repository for access and another Repository-P: repository for persistency. Both the repositories run on the Linux platform. Repository-A is an active component built on the Fedora Framework. There is a unique deposit and registration process which supports the workflow for verification, cataloguing and storage of objects.

- *Metadata.* The submissions are sent to the repository and are structured in the Metadata Encoding and Transmission Standard (The METS schema is a standard for encoding descriptive, administrative, and structural metadata regarding objects within a digital library, expressed using the XML Schema. The standard is being developed as an initiative of the Digital Library Federation).

- *Searching and Browsing.* There is a separate Search and Browse service which includes a specific Content Index that provides full-text indexing and intelligent indexing for the digitized objects structured in METS.

- *User personal area.* The Access service provides access for the users to this library which also includes User Workspace, where registered users can maintain virtual collections. Privileges are granted based on the user type.

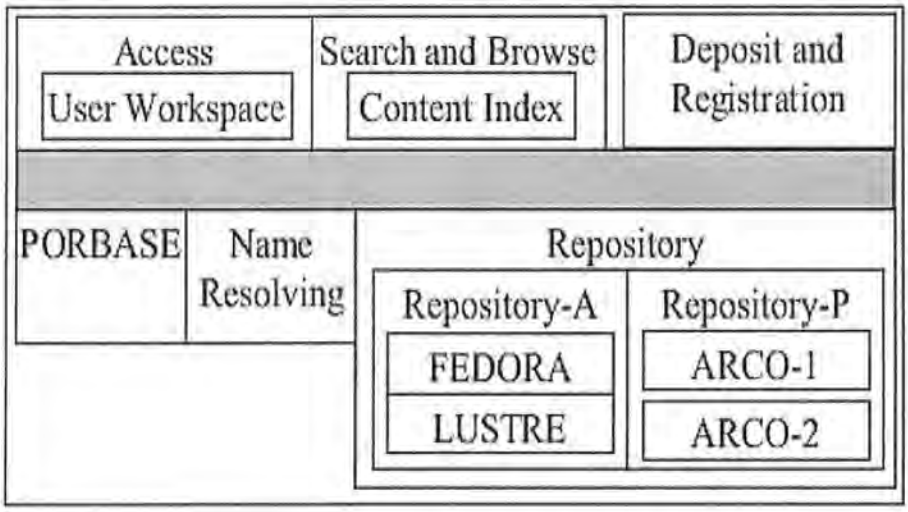


**Figure 6: Architecture of the National Digital Library of Portugal. Source: [7].**

## Common Components

After a careful study of architectures of the above digital libraries, we can conclude that although all of them have their own architectures, they have certain components in common.

### User Interface

The User Interface plays a key role in most libraries. Navigation of content is its primary objective. Most digital libraries support simple navigation to their contents rather than complex graphics or bulky images. The users of the digital libraries are very diverse and include students, instructors, and public at all levels, collection developers, librarians, and community interest groups. The common flavor in all the user interfaces we have identified is that the user formulates a query, gets a list of documents as a result and can

then either print the documents or download and display them. Most of the user interfaces had the following elements in common:

- Online-help
- Personal User Area
- Searching and Browsing
- Admin Section

Basic properties that every user interface should have to satisfy the audience are:

- A simple but catchy homepage that tells everything about the library
- An easy browsing facility of the contents of the digital library
- A search capability with simple keywords and advanced search options
- A document availability in a printable form

## Document Repository

Digital Libraries are comprised of different types of digital documents. Different file formats exist and most of the digital library documents fall into one of these formats. We identified that almost all the libraries had a safe storage place for their digital documents called the Document Repository. Most of the digital libraries have a file structure depending on the type of audience they serve. Quite a few of them have a flat file structure. Not many libraries are practicing version control on their documents. In most cases, documents are submitted via email or ftp to administrators who moderate and store them into the repository.

## Search Engine (Indexing and Intelligent Data Retrieval)

The most important component of a digital library is its search engine. The Search Engine is typically a web-based interface between users and the library's databases. A short description or a keyword is submitted to the search engine which gets back with all the related results. Most of the digital libraries have their search engine pointed to their metadata. As mentioned above in the documented solutions, many digital libraries use Apache's *Lucene,* a high-performance, full-featured test search engine library written in Java. Fundamental concepts of *Lucene* are index, document, field and term. A 'term' is a string, a 'field' is a sequence of terms and a 'document' is a sequence of fields. The 'index' stores statistics about terms in order to make term-based search more efficient.

## Metadata and Automatic Metadata Generation

All the digital libraries we looked into contain and serve metadata. However, only few of them support automatic generation of metadata. Metadata, in most cases, are generated by hand by the moderator of that digital library or by the contributors of the items.

## Protocols for Client/Server Information Retrieval (Z39.50 and OAI-PMH)

Most of the communication for components uses HTTP. Z39.50 is an old standard protocol but still in use for sharing metadata. Another protocol for sharing metadata is OAI-PMH which came into effect recently. Since metadata plays a key role in search and retrieval of the documents, there should always be a technique to serve these metadata to the outside world so that the digital documents can be shared among various libraries. Z39.50 and OAI-PMH are two standard protocols serving this purpose. Digital libraries that are built in the recent past have made sure that their metadata are shared using one of these protocols.

## *TeachEngineering* Architecture

### *Document Templates and Authoring*

The collection contains four types of documents: *Subject Areas*, *Curricular Units*, *Lessons* and (hands-on) *Activities*. All documents are stored in *XML* format. The structure and semantics for each of the document types are defined by their respective *XML* schemas.

To support authoring of these documents, we have designed a set of *XML templates*: bare-bones implementations of the four types of content documents. Document authors use these templates for authoring their documents. The software used for editing the templates is *Authentic*, a free-of-charge *XML* document editing tool produced by *Altova*. The templates themselves are created and maintained using *Altova's XMLSpy* software. Using this approach, document authors do not need to know *XML* in order to author their documents. This software also guarantees that the resulting documents contain well-formed *XML* and allow them to be validated against our *XML* schemas.

### *Document Repository*

The *XML* documents created with *Authentic* are stored in a centralized directory structure under (*CVS*) version control. All the documents combine to form our 'collection.' Document authors at the various universities check-in new documents or check-out existing documents to make changes after which they check them back in to the repository. An automated process checks out the collection daily and places it under the control of the *TeachEngineering* web server.

### *Document Spider*

The document *spider* is an automated process that periodically searches all documents in the collection for information that must be registered to the search engine. It stores its results in a relational database. The spider is built using *Java* and *JDOM* APIs with *XPath* functionality. *XPath* is a protocol for extracting information from an *XML* document. "The spider traverses the *XML* documents recursively following, within a document, all references to other *TeachEngineering* documents as well as external links" [1].

### *Metadata Generator*

In order to expose the collection to data harvesters and other service providers, another automated process generates Dublin Core metadata describing these documents. No metadata are manually edited or generated. An example of the metadata of a document follows:

```
<?xml version="1.0" ?>
- <nsdl_dc:nsdl_dc schemaVersion="1.02.000" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dct="http://purl.org/dc/terms/"
    xmlns:nsdl_dc="http://ns.nsdl.org/nsdl_dc_v1.02/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ns.nsdl.org/nsdl_dc_v1.02/ http://ns.nsdl.org/schemas/nsdl_dc/nsdl_dc_v1.02.xsd">
    <dc:title xml:lang="en-US">Air Pollution</dc:title>
    <dc:creator>Integrated Teaching and Learning Program, College of Engineering, University of Colorado at Boulder</dc:creator>
    <dc:subject>air pollution, air, pollution</dc:subject>
    <dc:description xml:lang="en-US">Students are introduced to the concept of air quality by investigating the composition,
      properties, atmospheric layers and everyday importance of air. They explore the sources and effects of visible and invisible
      air pollution. By learning some fundamental meteorology concepts (air pressure, barometers, prediction, convection
      currents, temperature inversions), students learn the impact of weather on air pollution control and prevention. Looking at
      models and maps, they explore the consequences of pollutant transport via weather and water cycles. Students are
      introduced to acids, bases and pH, and the environmental problem of acid rain, including how engineers address this type of
      pollution. Using simple models, they study the greenhouse effect, the impact of increased greenhouse gases on the planet's
      protective ozone layer and the global warming theory. Students explore the causes and effects of the Earth's ozone holes
      through an interactive simulation. Students identify the types and sources of indoor air pollutants in their school and home,
      evaluating actions that can be taken to reduce and prevent poor indoor air quality. By building and observing a few simple
      models of pollutant recovery methods, students explore the modern industrial technologies designed by engineers to clean
      up and prevent air pollution.</dc:description>
    <dc:publisher>TeachEngineering.com</dc:publisher>
    <dc:type xsi:type="dct:DCMIType">Text</dc:type>
    <dc:type>Curricular Unit</dc:type>
    <dc:format xsi:type="dct:IMT">text/xml</dc:format>
    <dc:identifier xsi:type="dct:URI">http://www.teachengineering.com/view_curriculumunit.php?
      url=http://www.teachengineering.com/collection/cub_/curricular_units/cub_air/cub_air_curriculumunit.xml</dc:identifier>
    <dc:language xsi:type="dct:RFC3066">en-US</dc:language>
    <dc:coverage xsi:type="dct:TGN">United States</dc:coverage>
    <dc:rights>Copyright 2005 - Integrated Teaching and Learning Program, College of Engineering, University of Colorado at
      Boulder</dc:rights>
    <dc:rights>http://www.teachengineering.com/policy_ipp.php</dc:rights>
    <dct:created xsi:type="dct:W3CDTF">20050309</dct:created>
    <dct:educationLevel xsi:type="nsdl_dc:NSDLEdLevel">Elementary School</dct:educationLevel>
    <dct:mediator>Teacher</dct:mediator>
  </nsdl_dc:nsdl_dc>
```

**Figure 7: A sample TeachEngineering metadata record**

## *Rendering Engine*

The rendering engine converts the *XML* documents into a visual form (HTML) to be browsed from an *HTTP* client such as a Web browser. The rendering engine is programmed using *PHP*'s document object model (*DOM*) and *XPath*. Once a document is requested for viewing by a user, a *DOM* object is built from it and *XPath* expressions are used to extract the document's data. Two types of data appear on the rendered document: data internal to the document and data that relate the document to other documents in the collection. The latter data are retrieved from the database generated by the document spider.

## *Web Service*

A web service API allows external programs to make requests to the *TeachEngineering* collection. This service is implemented in *PHP-REST*.

### *User Interface*

The user interface manages searches based on a variety of criteria such as keywords, grade level, cost for executing curriculum, required time, etc. In addition, documents can be listed in a variety of groupings and sorted in a variety of ways and can be cross-referenced with variables such as educational standards. The website also offers a personal area where users can store various preferred curricular materials, write curricular reviews, rate curricular materials, etc. Figure 8 describes the *TeachEngineering* Digital Library Architecture.

### *Step-by-step Development of TeachEngineering Digital Collection Document*

The sequence of events leading from the creation of some external curricular material to the K-12 teacher rendering this material as a *TeachEngineering* document on a web browser are listed below.

1. Curriculum developers create curricular material such as a *lesson* in a native format such as *MSWord.*
2. They download the *Document Template*, for a *lesson* and the *Authentic* software for editing the template.
3. They load the template into *Authentic* and cut and paste the various parts of the *MSWord* version of the document into the proper places in the *XML* lesson template.
4. After checking the resulting XML document for well-formedness, they check the document in to the *Document Repository* using *CVS.*
5. Once a day the *Document Repository* is automatically checked-out and put under control of the *TeachEngineering* web server.
6. Following this checkout, the *spider* extracts from the *XML* documents all the information needed by the search engine and some information needed by the rendering engine and stores this in a relational database.
7. Once a day, the *Metadata Generator* generates the metadata for all of the collection documents and makes them available for harvesting.
8. A K-12 teacher (user) accesses the *Search Engine* through the web-based *User Interface* and selects a document for viewing.
9. The *Rendering Engine* renders the document on the user's web browser.

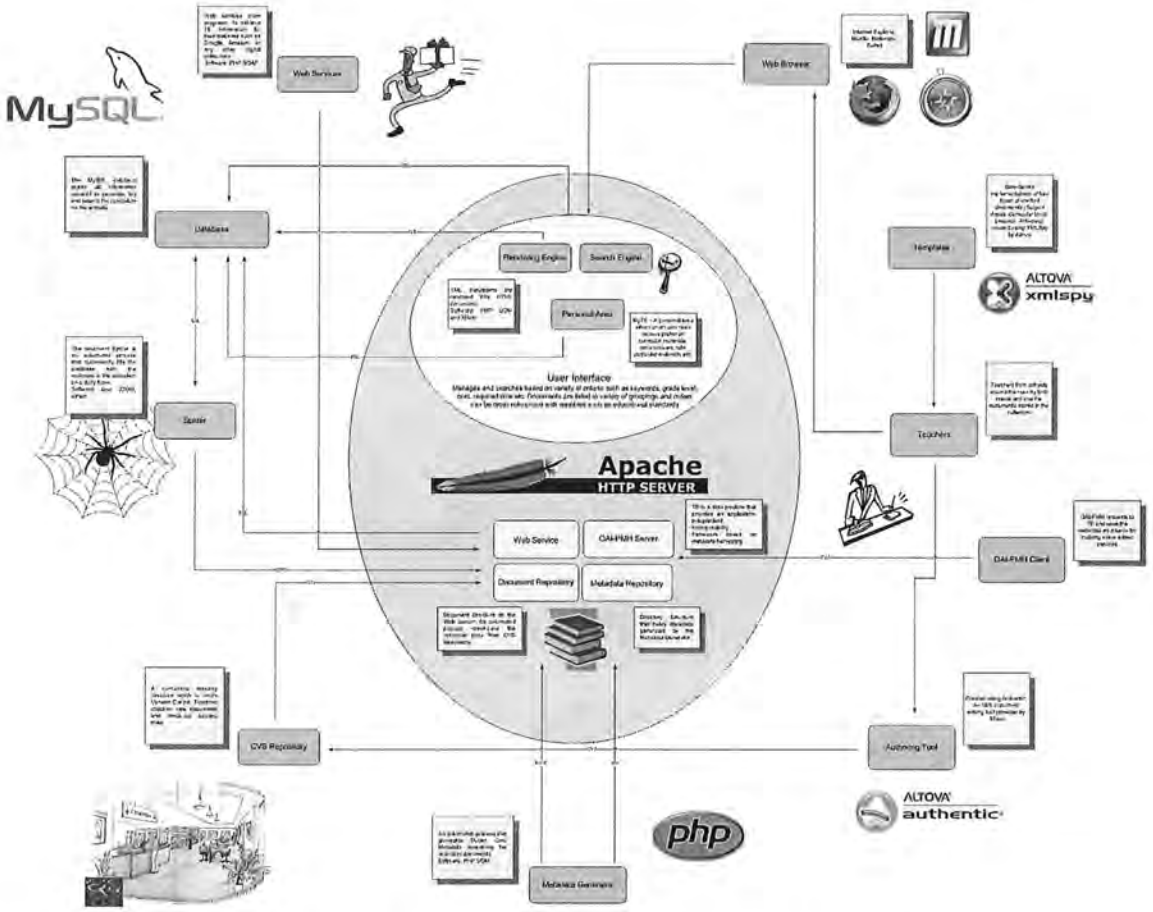


**Figure 8: *TeachEngineering* Digital Library Architecture.**

## *Metadata*

Metadata comprise structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata are often called data about data or information about information. The term *metadata* is used differently in different communities. Some use it to refer to machine understandable information, while others use it only for records that describe other resources. In the library environment, metadata are commonly used for any formal scheme of resource description, applying to any type of object, digital or non-digital.

There are three main types of metadata:

- *Descriptive metadata* describe a resource for purposes such as discovery and identification. They can include elements such as title, abstract, author, and keywords.

- *Structural metadata* indicate how compound objects are composed, for example, how pages are ordered to form chapters.
- *Administrative metadata* provide information to help manage a resource, such as when and how it was created, file type and other technical information, and who can access it.

Describing a resource with metadata allows it to be understood by both humans and machines in ways that promote interoperability. Interoperability is the ability of multiple systems with different hardware and software platforms, data structures, and interfaces to exchange data with minimal loss of content and functionality. Using defined metadata schemes, shared transfer protocols, and crosswalks between schemes, resources across the network can be searched.

Two approaches to interoperability are *cross-system search* –also known as *federated search* and *metadata harvesting*. The Z39.50 protocol is commonly used for cross-system search. Z39.50 implementers do not share metadata but map their own search capabilities to a common set of search attributes. A contrasting approach taken by the Open Archives Initiative is for all data providers to translate their native metadata to a common core set of elements and expose this for harvesting. A search service provider then gathers the metadata into a consistent central index to allow cross-repository searching regardless of the metadata formats used by participating repositories.

## Protocols for Client/Server Information Retrieval (Z39.50 and OAI-PMH)

### Z39.50

Z39.50 is a client/server protocol for searching and retrieving information from remote databases. This standard is maintained by the Library of Congress. Z39.50 is widely used in library environments. It is mostly implemented in inter-library catalogue searches. Z39.50 protocol supports a number of actions including search, retrieval, sort, and browse. It is an open standard that enables communication between systems that run on different hardware and use different software.

Cross platform communication is possible using this protocol. To initiate a Z39.50 session, the client initiates a Z-association by a series of messages sent to the server. The client and the server negotiate how many records to pass between them and after this negotiation the client submits its query. The Z39.50 client translates the query into a standard form and passes it to the Z39.50 server. The server executes this query over its databases and the 'result set' is dispatched to the client. If the size of the set is large, it is broken into components and transmitted piecemeal.

### Searching a database

The search query that is passed to the database contains search terms and attributes of those search terms. For example: author, title, date, etc. For example, if a user wants to search for an author's name, a "use" attribute specifies the search term as "author." If the user wants to search for all books published after a certain date, a "use" attribute specifies

the search term is a "date of publication" and a "relation" attribute specifies that the user wants all dates of publication "greater than" a particular date. All these attributes are enumerated in Z39.50 attribute sets. The server creates a result set after executing the query. Clients can either request for return of those records or can issue additional searches.

**Retrieving records from a database**
When the user wants to display records from the result set, Z39.50 provides some choices about which data elements from the database record the user can request. Standardized element set names and record syntax are used to support client/server communication while retrieving information.

Z39.50 is a pre-Web technology, and various working groups are attempting to update it to fit better into the modern environment. Z39.50 International: Next Generation is the current attempt and the most important protocols involved are the twin protocols SRU/SRW which drop the Z39.50 communications protocol (replacing it with HTTP) but attempt to preserve the benefits of the query syntax. SRU (Search/Retrieval using URL) enables queries to be expressed in URL query strings while SRW (Search/Retrieval using web service) uses SOAP.

**Search/Retrieval as Web Service**
SRW is an XML-oriented protocol for performing searches and other information retrieval operations across the Internet. It uses technologies such as SOAP and XPath. The protocol can be carried in two ways: SOAP or parameters in a URL. SRW allows users to search databases of records. Clients send requests with parameters and the server responds with XML records.
For example, a search request might look like the following:

```
<searchRetrieveRequest>
        <version>1.1</version>
        <query>dc.title all "water"</query>
        <maximumRecords>1</maximumRecords>
        <startRecord>1</startrecord>
        <recordSchema>dc</recordSchema>
</searchRetrieveRequest>
```

The response is also in XML:

```
<searchRetrieveResponse>
    <version>1.1</version>
    <numberOfRecords>10</numberOfRecords>
    <records>
      <record>
      <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
      <recordData>
          <dc:record>
          <dc:title>ocean water</dc:title>
          </dc:record>
      </recordData>
```

```
      </record>
    </records>
</searchRetrieveResponse>
```

**Search/Retrieval using URL**
SRU is a REST-ful web service protocol. The parameters are encoded as name/value pairs in the query string of a URL. Operations sent by SRU clients can only be transmitted via HTTP GET requests. The results of SRU requests are also XML streams.

## Open Archive Initiative Protocol for Metadata Harvesting

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) provides an application-independent interoperability framework based on metadata harvesting. This is the most recent protocol used in information retrieval. OAI-PMH is a simple and easily implemented protocol. It facilitates sharing of metadata via a harvesting model.
There are two classes of participants in the OAI framework. D*ata provider* maintains one or more repositories that support OAI-PMH to expose metadata. *TeachEngineering* acts as the data provider serving metadata to *NSDL.* S*ervice provider* issues OAI-PMH requests to the data providers and uses the metadata as a basis for building value-added services. *NSDL* acts as a service provider that uses metadata served by *TeachEngineering.*

OAI-PMH requests are implemented as HTTP requests. Requests must be submitted using HTTP POST or GET methods. Since the POST method has the advantage of imposing no limitations on the length of the arguments, it is used more than the GET method but the repositories must support both the methods. There is a single base URL for all the requests.

All requests consist of *keyword* arguments in the key=value pair form. Arguments appear in any order but are separated by ampersand (&).

**Protocol Requests and Responses**
Requests or verbs declared in OAI-PMH are described below.

- **GetRecord**
  This verb is used to retrieve an individual metadata record from a repository.
  Arguments:
  - Identifier: specifies unique identifier of the item in the repository
  - metadataPrefix: specifies metadata prefix of the format that should be included in the metadata part of the returned record.

- **Identify**

This verb is used to retrieve information about a repository. This request has no arguments.
The response to this request consists of:

- repositoryName: name of the repository
- baseURL: base URL of the repository
- protocolVersion: the version of OAI-PMH supported by the repository
- earliestDatestamp: UTCdatetime which is the lower limit of all datestamps.
- deleteRecord: manner in which the repository supports deleteRecords.
- granularity

- **ListIdentifiers**
  This method is an abbreviated form of ListRecords which retrieves only headers.
  Arguments:
  - from: lower bound of datestamp
  - until: upper bound of datestamp
  - metadataPrefix
  - set: specifies set criteria for selective harvesting
  - resumptionToken

- **ListMetadataFormats**
  This verb is used to retrieve the metadata formats available from a repository.
  Arguments:
  - identifier: unique identifier of the item for which available metadata formats are being requested.

- **ListRecords**
  This verb is used to harvest records from a repository.
  Arguments:
  - from
  - until
  - set
  - resumptionToken
  - metadataPrefix

- **ListSets**
  This verb is used to retrieve set structure of a repository
  Arguments:
  - resumptionToken

There are some exceptions that should be handled. If these exceptions are not handled, the service provider cannot harvest data from the data provider. OAI-PMH defines a set of requests and the format of the responses to each of these requests specified through an XML schema. It must also respond to malformed requests with appropriate errors and exception conditions.

A number of software tools are available at the OAI website (http://www.openarchives.org/tools/tools.html). *TeachEngineering* uses *OAI-PMH2 XMLFile*, a file-based data provider. This is a data provider module that operates over a set of XML files that contain the metadata. It is meant to require a minimum of effort while retaining all the flexibility of the OAI protocol.

We can check our repository against each of the OAI-PMH verbs in turn, setting parameters where required for date ranges, metadataPrefix, identifier, set, and resumption token. Thus, all aspects of the protocol are tested, and the results of queries are checked for conformance with the expected syntax.

## State of TE

The *TeachEngineering* Digital Library went live on the 15th January, 2005. It is currently hosted by Open Source Lab at Oregon State University, Corvallis.

The curriculum in *TeachEngineering* is organized by Subject Areas, Curricular units, Lessons and Activities.

Some of these items are standalone, and others are organized hierarchically with activities comprising lessons, which in turn comprise units, which fall under one or more subject areas.

At the time of this writing, there are 13 Subject Areas, 20 Curricular Units, 111 Lessons and 201 Activities. 44 users registered themselves with *TeachEngineering* and many more are accessing *TeachEngineering* content from places such as Japan, Australia, Canada and other places around the world. *TeachEngineering* content is indexed by al the well known web crawlers and its metadata are harvested by *NSDL*. Activities are accessed more than lessons and curricular units. *TeachEngineering* is supported on all major browsers.

The *TeachEngineering* Web Spider tracks link information from all the four contributors:

**Table 1: Results of Spider check of *TeachEngineering* links (05/31/2005)**

| Contributor | Success Links | Error Links |
|---|---|---|
| University of Colorado | 2556 | 24 |
| Colorado School of Mines | 67 | 0 |
| Worcester Polytechnic | 236 | 6 |
| Duke University | 431 | 4 |

The TeachEngineering usage statistics can be found at www.teachengineering.com/awstats. The American Society for Engineering Education Annual Conference and Exposition is going to be held in Portland, Oregon from June 12th – 18th, 2005. Teacher workshops will be conducted during the conference to test the usability of TeachEngineering Digital Library.

## TE Architecture Evaluation

The *TeachEngineering* Digital Library has several components that are in common with many other digital libraries. However, there is an important difference including:

- The Document templates and authoring process are unique when compared to that of other digital collections. The content of *TeachEngineering* in completely in XML and it gives a lot of flexibility for updating the template libraries and authoring process. Adding and removing content from an XML document is easy.

- Having version control taking over the repository is a good way of handling multiple versions of documents in *TeachEngineering*. In general, curriculum developers of other libraries are typically submitting their content either as an attachment or email, which is then edited, moderated and manually added into the repository.

- The Document Spider keeps track of the links from every document and creates a log so that any broken link can be fixed as soon as it is identified. This automated tool runs periodically and stores its content in a relational database. This way, *TeachEngineering* is makes sure that all parts of a document are available. These data are very helpful for search and retrieval.

- The Metadata Generator automatically generates metadata and also checks the validity of the document's content. An automatic emailer generates email to the curriculum developers listing all the errors that occurred in their branch in the repository. This is a neat and clean process which avoids *TeachEngineering* administrators having to check metadata daily.

- *TeachEngineering* strictly follows open standards. Most of the software involved in developing the *TeachEngineering* Digital Library is open sourced.

- *TeachEngineering* also followed current software engineering practices like having a bug tracking tool, *Bugzilla*, to track the bugs. *Awstats* monitors the traffic on the *TeachEngineering* website.

Some disadvantages identified in the architecture are:

- Many other digital libraries don't have a specific document structure. They support multiple types of documents while *TeachEngineering* doesn't. *TeachEngineering* is forcing the curriculum developers format their content to can fit into its templates.

- The Rendering Engine of *TeachEngineering* is not generic. Rendering the XML documents is built by feeding XML tags statically. Therefore, change in the XML templates, the rendering engine will not propagate those changes. The source code of the rendering engine should be changed to be in sync with the document templates.

## Conclusion

The *TeachEngineering* Digital Library is a team effort. It is quite interesting that *TeachEngineering* has all the common components that are present in several other digital libraries.

At the time of this writing, there is no best software architecture. The architectures like DSpace and Fedora are considered stable but can only be used to serve specific communities. While the collection developers can decide which component to use from already developed ones, they may extend the existing components to serve their purpose. This involved lot of work if the components from already developed ones come up with newer versions.

I wish this report would of help for current and future collection developers nationally and internationally.

# References


1. Rene Reitsma, Brandon Whitehead, Venkata Satya Gokul Suryadevara (2005), *Digital Libraries and XML-Relational Data Binding.* Dr.Dobb's Journal; April, 2005; 42-47.

2. Kumar, A., Sahai, R., Chavez, R., Schwertner, N. (2004) *Architecting an Extensible Digital Repository.* Proceedings of the 2004 ACM/IEEE Joint Conference on Digital Libraries (JCDL), Tuscon, AZ, June 7-11; ACM; 2-10.

3. William.C Janssen, Kris Poppat, *UpLib: A Universal Personal Digital Library System*, DocEng'03, November 20 – 22, 2003, Grenoble, France.

4. Janee, G., Flew, J. (2002) *The ADEPT Digital Library Architecture.* Proceedings of the 2002 ACM/IEEE Joint Conference on Digital Libraries (JCDL), Portland, OR, July 14-18; ACM; 342-350.

5. Flew, J., Freeston, M., Freitas, N., Hill, L., Janee, G., Lovette, K., Nideffer, R., Smith, T., Zheng, Q. (1998) *The Alexandria Digital Library Architecture.* Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries; Springer-Verlag; 61-73.

6. Tansley, R., Bass, M., Stuve, D., Branschofsky, M., Chudnov, D., McClellan, G., Smith, M. (2003) *The DSpace Institutional Digital Repository System: Current Functionality.* Proceedings of the 2003 ACM/IEEE Joint Conference on Digital Libraries (JCDL), Houston, TX, May 27-31; ACM; 87-97.

7. Borbinha, J., Freire, N., Neves, J., *BND: The Architecture of a National Digital Library.* Proceedings of Joint Conference on Digital Libraries, June 7-11, 2004, Tucson, Arizona; ACM 21-22.

8. Millman, D. (2000). *Columbia University Digital Library Architecture and Services*, Academic Information Systems; January, 2000.

9. Lagoze, C., Hoehn, W., Millman, D., Arms, W., Gan, S., Hillman, D., Ingram, C., Kraft, D., Marisa, R., Phipps, J., Saylor, J., Terrizzi, C., Allan, J., Guzman-Lara, S., Kalt, T. (2002) *Core Services in the Architecture of the National Science Digital Library (NSDL).* Proceedings of the 2002 ACM/IEEE Joint Conference on Digital Libraries (JCDL), Portland, OR, July 14-18; ACM; 201-209.

10. *The Open Archives Initiative for Metadata Harvesting*; http://www.openarchives.org/OAI/openarchivesprotocol.html

11. *National Information Standards Organization Z39.50 Information Retrieval Protocol* http://www.loc.gov/z3950/gateway.html.

12. McNab, R.J., Witten, I.H., Bodie, S.J. (1998) *A Distributed Digital Library Architecture Incorporating Different Index Styles.* Proceedings of the Advances in Digital Libraries Conference; IEEE; 36-45.

13. Reddy, R., Ager, T., Chellappa, R.,Croft, W.B., Brown, B., Mendel, J., Shamos, M. (1999), *Digital Information Organization in Japan*; WTEC Panel on DIO in Japan; February, 1999.

14. Fuhr, N., Hansen, P., Mabe, M., Micsik, A., Solvberg, I. (2001), *Digital Libraries: A Generic Classification and Evaluation Scheme*; Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries; Springer-Verlag; 187 – 199, 2001.

15. The *TeachEngineering* Digital Library
http://www.teachengineering.com

16. The National Science Digital Library
http://www.nsdl.org