Digital Library Curriculum Development

Module: 5-a: DL architectures

Draft, 9/13/2008

1. Module name

DL architectures

2. Scope

This module covers architectures of digital libraries (DLs) such as federated architecture, distributed architecture, and service-oriented architecture, which appear in the DL literature.

Note: Implementation details are not included here.

3. Learning objectives

a. Be able to distinguish different types of DL architectures mentioned in the scope description section.

b. Be able to write specifications for a DL of a particular type of architecture, given a particular context and a particular set of needs.

4. 5S characteristics of the module

DL architectures are derived from the more conceptual DL models, which contain all the 5Ss as their components. Therefore, these architectures address all the components of the 5S Framework.

a. Streams: there are specifications about what type of digital objects can be stored or provided by the DL architecture/models.

b. Spaces: storage space and user interfaces are present in the DL architectures/models.

c. Scenarios: There are series of steps to go through for each component of DL architectures, and models to communicate with other components; these constitute scenarios.

d. Societies: this is not strongly present here, however, the architects and the DL model developers represent the Societies component of the 5S Framework.

e. Structures: DL architectures and models have structures, sometimes in layers. Each component of them such as a repository also has a structure.

5. Level of effort required (in-class and out-of-class time required for students)

To achieve learning objectives:

a. Out-of-class time: 6-8 hours

   Reading the assigned papers, preparation for group presentations, creation of concept maps for different DL architectures and models

b. In-class time: 2.5 hours

   1 hour for the lecture, 1.5 hours for group presentations and Q/A session

6. Relationships with other modules

Figure 1. 5-a module's relationships with other modules

As is presented in Figure 1, this module has a strong relationship with 1-a (10-c): Conceptual frameworks, theories, definitions and 5-b: Application software. Knowing the conceptual framework and definitions will help with understanding of DL architectures and models.

By studying the three modules, 1-a(10-c), 5-a, and 5-b, students might be able to understand DLs from the conceptual level to the real-life systems level.

Therefore, it is recommended to teach/learn the above three modules in the order indicated by the arrows.

7. Prerequisite knowledge required

None

8. Introductory remedial instruction

None

9. Body of knowledge

- ***Topic: Principles of DL architecture***

1. All of these principles are covered in more depth in other modules. But it's worth covering them briefly here because all DL application development is based on these principles.
2. Kahn & Wilensky (1995) "provides a method for naming, identifying and/or invoking digital objects in a system of distributed repositories" (section 5, ¶ 1), and provided definitions of many important entities in DLs (e.g., digital object, handle, metadata, repository).
    a. DLs are merely one example of a digital information service, according to Kahn & Wilensky's framework.
3. Arms' (1995) General Principles for DL Architecture:
    a. The technical framework exists within a legal and social framework.
        i. E.g., music
            - Copyright owners want to maintain control over how content is used.
            - Specifications for how content can be used can be add-ons to the content.
                a. But that can be removed or altered.
            - Must build some control mechanism into the content itself
                a. E.g, DRM, watermark
                b. But that breaks down the wall separating architecture & content
            - Cripple functionality to comply w/ law?
                a. E.g., iTunes, Quality of Service
        ii. This Principle will be addressed in Module 9e: Intellectual Property.
    b. Understanding of digital library concepts is hampered by terminology.
        i. One possible solution: Invent new terms
            - E.g., sharium, Information Commons, etc.
        ii. Some terms persist: digital object, handle, metadata, etc.
        iii. Hopefully this problem has diminished as DLs have become more common & familiar.
        iv. One term is still ambiguous: DL.
    c. The underlying architecture should be separate from the content stored in the library.
        i. Separation of architecture & content
            - This is the premise of all markup languages: HTML, XML, etc.
            - HTML does this separation poorly.
        ii. Functionality can be general to all content or specific to types of content.
            - E.g., text can be edited, images usually can only be viewed (except with special tools like PhotoShop); different file types invoke different applications, etc.
            - This is the function of modules in programming.
        iii. "Characteristics that apply to all types of material"

        - Some characteristics are more or less universal: e.g., describable with the Dublin Core
    iv. This Principle will be addressed in Module 4-a: Metadata
d. Names and identifiers are the basic building block for the digital library.
    i. In the physical world:
        - e.g., SSN, house address, phone number
        - e.g., ISBN: not unique; gets reused
    ii. Online:
        - URLs: Not unique, not persistent
        - DOI (www.doi.org) & PURL (purl.oclc.org): Unique & persistent
e. Digital library objects are more than collections of bits.
    i. A stream of bits is what is downloaded.
        - But users interpret an intellectual object.
        - Can some of that interpretation be embedded into the structure of the bitstream?
    ii. This becomes a more complex problem with complex objects.
f. The digital library object that is used is different from the stored object.
    i. Architecture must distinguish between digital objects as they are:
        - created by an originator,
        - stored in a repository,
        - disseminated to a user.
    ii. Legal ramifications: copying & caching
g. Repositories must look after the information they hold.
    i. Archiving & long-term preservation
    ii. Need DOIs / PURLs
    iii. Migration to new formats
    iv. Prevent hacking & unauthorized changes to files.
    v. This Principle will be addressed in Module 8: Preservation
h. Users want intellectual works, not digital objects.
    i. Complex objects are mutable.
    ii. The same object can be part of multiple “works”.
        - E.g., an image as part of multiple documents.
        - E.g., the same paragraph or citation as part of multiple documents.

- ***Topic: DSpace Architecture***

Content based on
http://www.dspace.org/index.php?option=com_content&task=view&id=145#overview
This covers the popular DSpace system. Other related systems not covered include Greenstone and Fedora; these are similar to DSpace or the systems and architectures covered below.

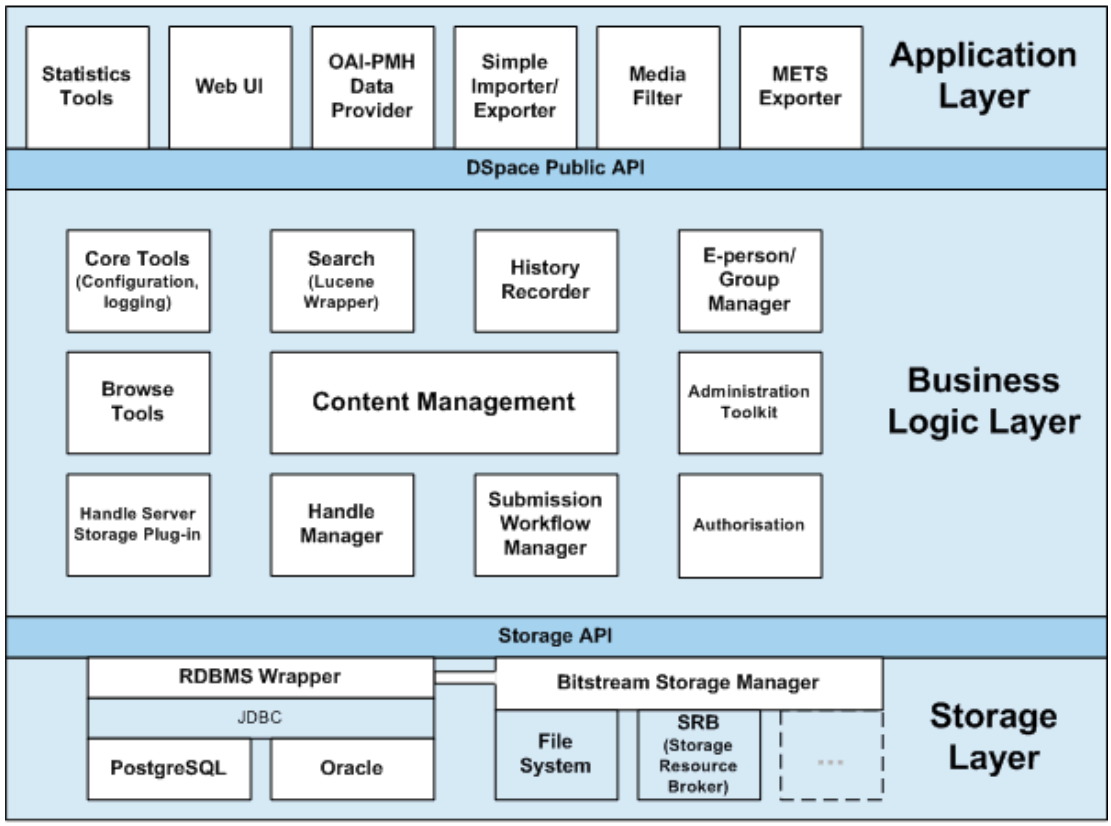

Figure 2. DSpace System Architecture

1. DSpace architecture overview

    i. DSpace architecture has three layers and two APIs as shown in Figure 2.
        i. Storage layer stores digital objects and their metadata in databases and file systems.
        ii. Business logic layer includes key operations such as searching and browsing services. It also manages users and authorization;; it includes broad suites of services to access and preserve content.
        iii. Application layer allows users to access DL system through its user interface. DSpace communicates with the outside world through this layer.
    j. Each layer only invokes the layer below it. For example, the Application layer cannot access the Storage layer directly.

k. Each component in the Storage layer and the Business Logic layer has a defined API. Storage API (Public API) is a combination of APIs for components in the Storage Layer (Business Logic Layer).

2. More details on components of the three-layers

    l. Storage Layer
        i. RDBMS: DSpace stores information about the user groups, organization and metadata of content, workflows, authorization, and indices.
        ii. Bitstream Store: content is stored two ways - in the server's file system, or using the Storage Resource Broker (SRB), which is a robust storage manager with a backup feature.
    m. Business Logic Layer
        i. Search: it is based on the Lucene search engine.
        ii. Browse: provides dates, authors, titles, and subjects.
        iii. History recorder: it captures significant changes in DSpace.
        iv. Administration toolkit: provides command line tool to create admins and configure DSpace system
        v. Workflow system: it models item's status in the system such as submitted, step-1, step-2, archive, etc., so that people with different responsibilities can correct and view the item's status.
        vi. Content management: it provides means to read or modify content in the storage layer.
        vii. Core classes: are basic classes used throughout DSpace.
        viii. Authorization system: it enforces policies, which are attached to resources. Policies also restrict responsibilities/actions of the users.
    n. Application Layer
        i. Web user interface: end-users access DSpace through the web-browser-based interface. It is the most used component in this layer.
        ii. OAI-PMH data provider: it provides a platform for metadata harvesting.
        iii. Item importer/exporter: the importer bypasses workflows assigned to a collection unless a flag is given specifically. The exporter can export a single item or a collection of items.
        iv. Media filter: it converts content to another new form of content. For example, a thumbnail image can be created from an item containing an image by applying filters.
        v. Registration: it is an alternate way to incorporate items, their metadata, and bit streams into DSpace.
        vi. Sub-community management
        vii. METS export tools (experimental): it exports DSpace items to a file system with the metadata in METS format

- ***Topic: Federated Architecture – Networked Digital Library of Theses and Dissertations (NDLTD) with MARIAN as a mediation middleware***

Content is based on: Gonçalves, Marcos A., Robert K. France, Edward A. Fox. 2001. MARIAN: Flexible Interoperability for Federated Digital Libraries, in *Proceedings of the Fifth European Conference on Research and Advanced Technology for Digital Libraries (ECDL2001)*, Darmstadt, Germany, September 4-9, 2001, pp. 173-186 (http://www.dlib.vt.edu/reports/ecdl2001_8.pdf).

After several years, however, when OAI-PMH became more widely used, OCLC began to run a Union Catalog, harvesting from OAI data providers, and integrating that information with WorldCat records; VTLS and Scirus run services atop the Union Catalog at present.

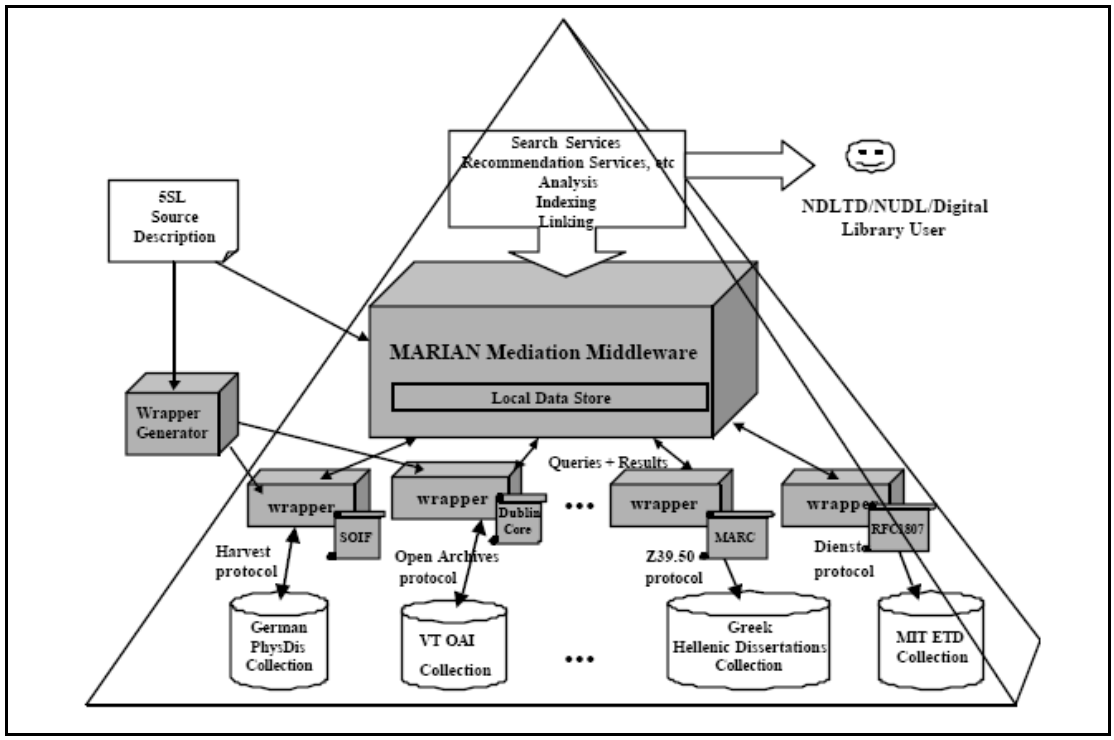

Figure 3. The NDLTD Union Archive Architecture

1. NDLTD overview
    a. An international consortium of universities, libraries, and other institutions focused on electronic theses and dissertations.
    b. A networked (federated) system that consists of multiple autonomous information systems (see the different collections in the bottom of Figure 3 such as VT OAI, MIT ETD, German PhyDis in member institutions), which are distributed in different locations and provide heterogeneous services.
    c. Although the information is searched and retrieved from various member institutions, a unified and transparent view of information sources is provided to DL patrons.
    d. Characteristics that complicate interoperability across member systems

i. Autonomy: member institutions manage most of their services for their scholars (e.g., the Virginia Tech library administers its ETD collection mostly for its own patrons, but at the same time participates in NDLTD).
ii. Decentralization: member institutions are not asked to report updates on their data to a central coordinator.
iii. Minimal interoperability: members should provide Unique Resource Names (URNs) and metadata records for all data but are not required to support the same standards/protocols.
iv. Heterogeneity: member institutions might use different languages, metadata standards, protocols, and character codings; they differ in user characteristics, preferences, and capabilities. This is one of the big challenges in NDLTD.
v. Massive amount of data and dynamism: new member institutions are constantly added and they provide input data continuously.

2. Union Archive vs. Federated Search: design decisions for federated architecture
    a. Union Archive
        i. Information is periodically collected from each member institution
        ii. Processed and merged with information from other institutions
        iii. Loaded into the central union archive
        iv. User queries are searched within the union archive without interacting with individual member institutions
        v. ADVANTAGES
            - Adequate performance at query time
        vi. DISADVANTAGES
            - Information might not be up-to-date (unless information collection period is short)
            - Concerns about data quality (multiplicity of representation, errors in data, imprecision) and consistency
    b. Federated Search
        i. There is no central archive. Data stays in each member institution.
        ii. User entered queries are passed to each member institution at query time.
        iii. ADVANTAGES
            - It is guaranteed to retrieve up-to-date information.
            - There is no data replication (whereas the union archive contains copies of original information from member institutions).
        iv. DISADVANTAGES
            - More sophisticated strategies for query optimization and data fusion (of retrieved information from each member institutions) are needed.
            - Low performance compared to union archive approach
        v. Other considerations
            - Network availability, latency, amount of data to transfer, etc. should be considered (e.g., some member institutions

might have low network availability. This worse-case situation decides the overall performance of the federated DL system).

c. Union Archive approach is used in NDLTD due to:
    i. Complexity of query translation in heterogeneous environment (e.g., different languages, protocols, etc.)
    ii. Poor and inconsistent network connectivity in some member institutions
    iii. Variability in server load and administration.

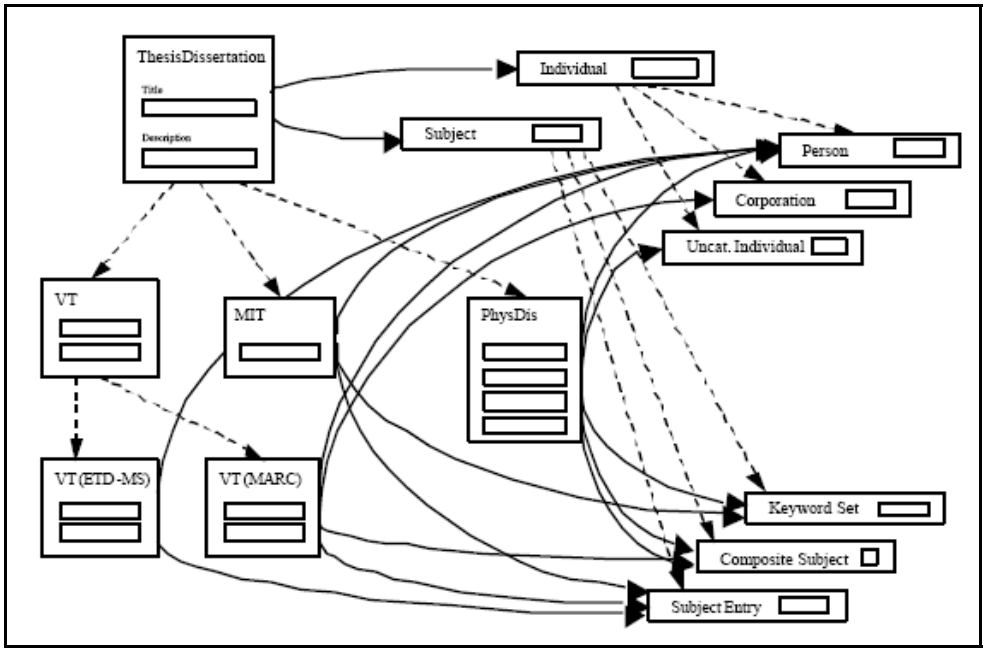

Figure 4. A data fusion approach - semantically similar object classes overlap

3. NDLTD Union Archive Architecture – please see Figure 3
    a. The MARIAN Mediation Middleware acts as a tool supporting interoperability. It provides NDLTD users a uniform interface to search and browse collections in member institutions, which use different protocols, formats, and systems.
    b. Like other union archive approaches, it has mechanisms to:
        i. Harvest data from various sources
            - To overcome the differences in member institution systems, special software modules such as wrappers (shown in Figure 3) are used.
            - To apply those wrappers, more information about the data collections, such as capabilities of remote collections and their internal document structures, should be known to the NDLTD system.
            - 5SL language can describe collection specific information and it is used to semi-automatically generate wrappers (see

5SL Source Description and Wrapper Generator in Figure 3).

ii. Combine (fuse) those harvested data for use (shown in Figure 4)
   - Each metadata standard has different classes such as Subject, Individual, Headings, Keywords, etc.
   - Merge those classes that have similar semantics (meanings); separate them if they have different semantics.
   - This approach is an attempt to reduce redundancy of data while keeping the collected data flexible enough to be modified in changing circumstances.

- ***Topic: A Service-Oriented Architecture (SOA) for CiteSeer***

Content is based on: Petinot, Y., Giles, C. L., Bhatnagar, V., Teregowda, P. B., Han, H., and Councill, I. 2004. A service-oriented architecture for digital libraries. In *Proceedings of the 2nd international Conference on Service Oriented Computing* (New York, NY, USA, November 15 - 19, 2004). ICSOC '04. ACM, New York, NY, 263-268. DOI= http://doi.acm.org/10.1145/1035167.1035205

1. Current CiteSeer architecture overview (please see Figure 5)

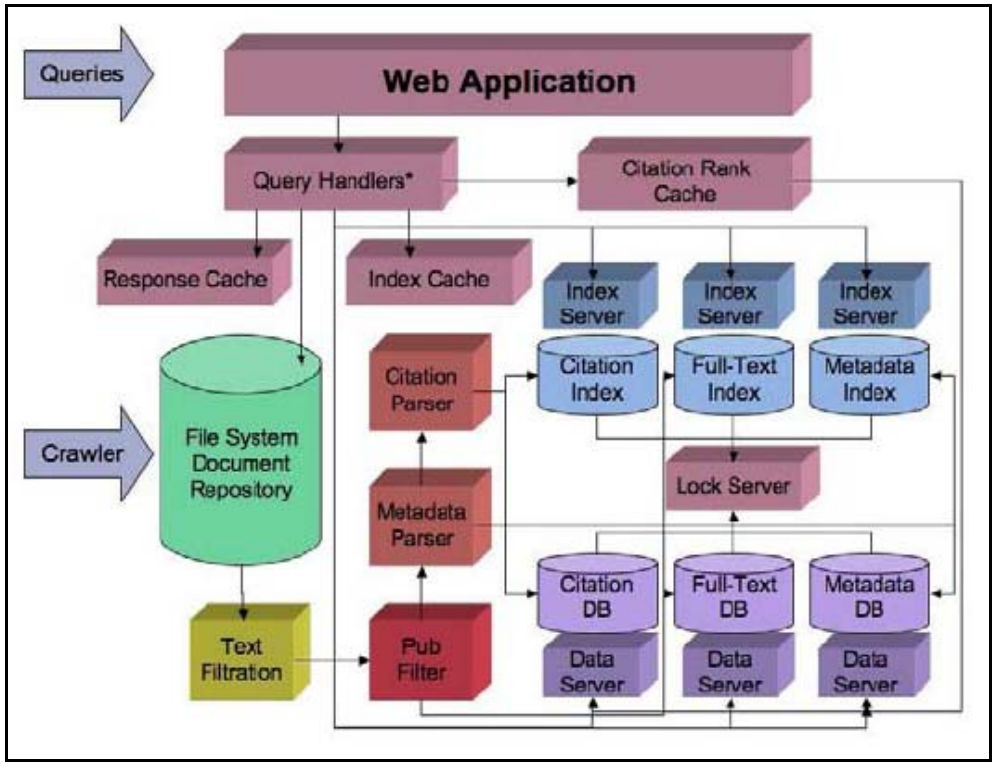

Figure 5. Current CiteSeer architecture

(Excerpt from CiteSeerX - A Scalable Autonomous Scientific Digital Library, http://clgiles.ist.psu.edu/papers/www2006-citeseerx.pdf)

a. CiteSeer (http://citeseer.ist.psu.edu/) is a scientific literature digital library and search engine that provides citations and cached papers in the fields of computer science and information science.
b. It consists of three basic components
    i. A focused crawler
    ii. Document archive and related indices
    iii. Query interface
c. How it works
    i. By a focused crawler, raw documents in PDF and PostScript formats are collected into a File System Document Repository.
    ii. After filtration of academic documents, data such as citations, full-text and metadata are extracted from them using autonomous citation indexing and stored into relevant indices and databases. References in research papers are linked to each other for providing navigation and evaluation services.
    iii. Through a web application, users can search the index containers and databases.
    iv. Lock Server, which is located between the Full-Text Index container and Full-Text DB, is used to lock the table or index level during updates.

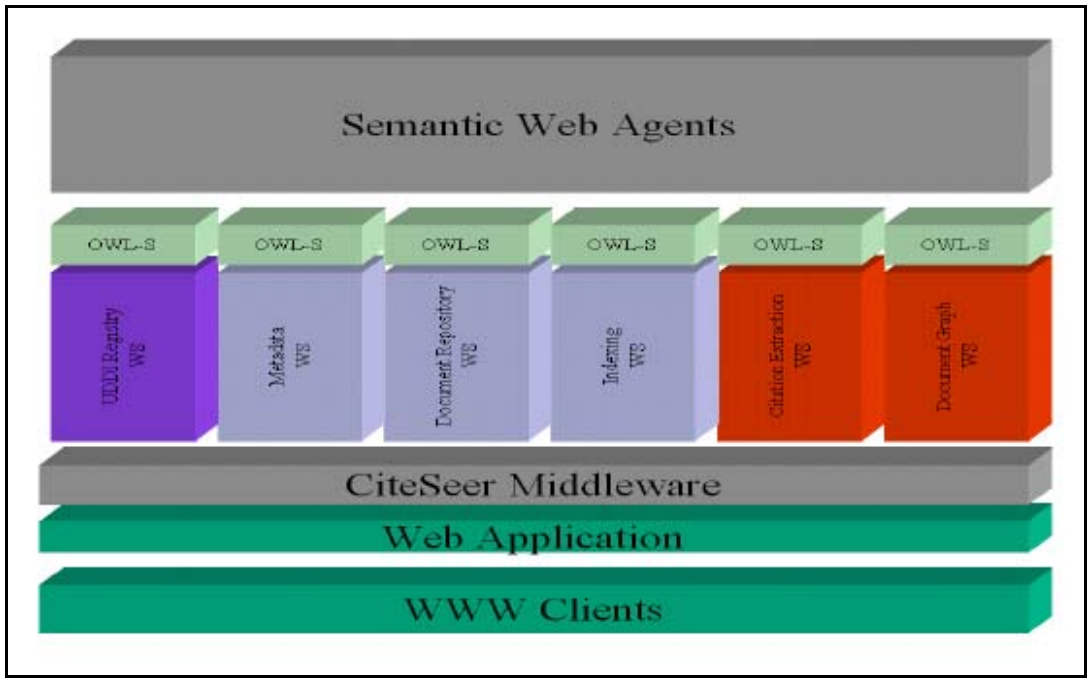

Figure 6. SOA-based CiteSeer Architecture

2. New CiteSeer architecture based on SOA

    a. Current CiteSeer architecture in Figure 5 might be seen as a monolithic system architecture, where several elementary services such as indexing,

filtering, citation extracting, citation ranking are coordinating with each other to achieve tasks.

b. We might think of those services as Web Services and re-organize the architecture around them.
c. Web Services in the new architecture can be categorized into two parts with an additional service, a UDDI registry service, where all the Web Services are registered.

    i. Part 1 - Fundamental CiteSeer services

        - Citation Extraction Service: it provides autonomous citation parsing functionality.
        - Citation Graph Service: it maintains a citation graph, which is a directed graph where nodes represent documents and edges represent citation relationships, for example, cites/cited-by relationships.

    ii. Part 2 - Utility services in most information systems

        - Indexing Service: It provides functionality such as inverted file and mapping elementary tokens to documents. In CiteSeer, two different kinds are used independently to index documents and their citations.
        - Metadata Service: It is equivalent to providing the OAI-PMH service.
        - Electronic Repository Service: It is aware of duplicate documents. Federation of this service might be provided as a solution for copyright management and mirroring concerns.
        - Electronic Conversion Service: It converts an electronic format to another. If the other format conversion services are made available on the Internet, CiteSeer can support those conversion services immediately.
        - Duplicate Identification Service: It compares the similarity between two documents to check the amount of overlap. This service might be used to discover an alternate URI.

    iii. Service Registration (UDDI registry service)

        - CiteSeer Web Services can be used by Semantic Web agents without problems.
        - However, for human users to access those Web Services through the Web application and CiteSeer Middleware, a meeting point, where Web Services are advertised and discovered, is necessary.
        - The UDDI registry Web Service does this job.

d. To transform Web Services into Semantic Web Services, Ontology Web Language for Services (OWL-S) is used to express the Web Service processes.
e. In summary, Semantic Web agents have access to individual Web Services expressed by OWL-S and the human users also have access to those through the Web Application layer and Middleware layer that has access to the UDDI registry Web Service.

- ***Topic: Component-based DL architecture (e.g., Open Digital Libraries)***

The first clear argument to shift to components for DLs was developed during the doctoral research of Hussein Suleman. The most complete reference is his dissertation: Hussein Suleman, "Open Digital Libraries", Nov. 2002, Virginia Tech Computer Science PhD dissertation, Blacksburg, VA, USA, http://scholar.lib.vt.edu/theses/available/etd-11222002-155624/

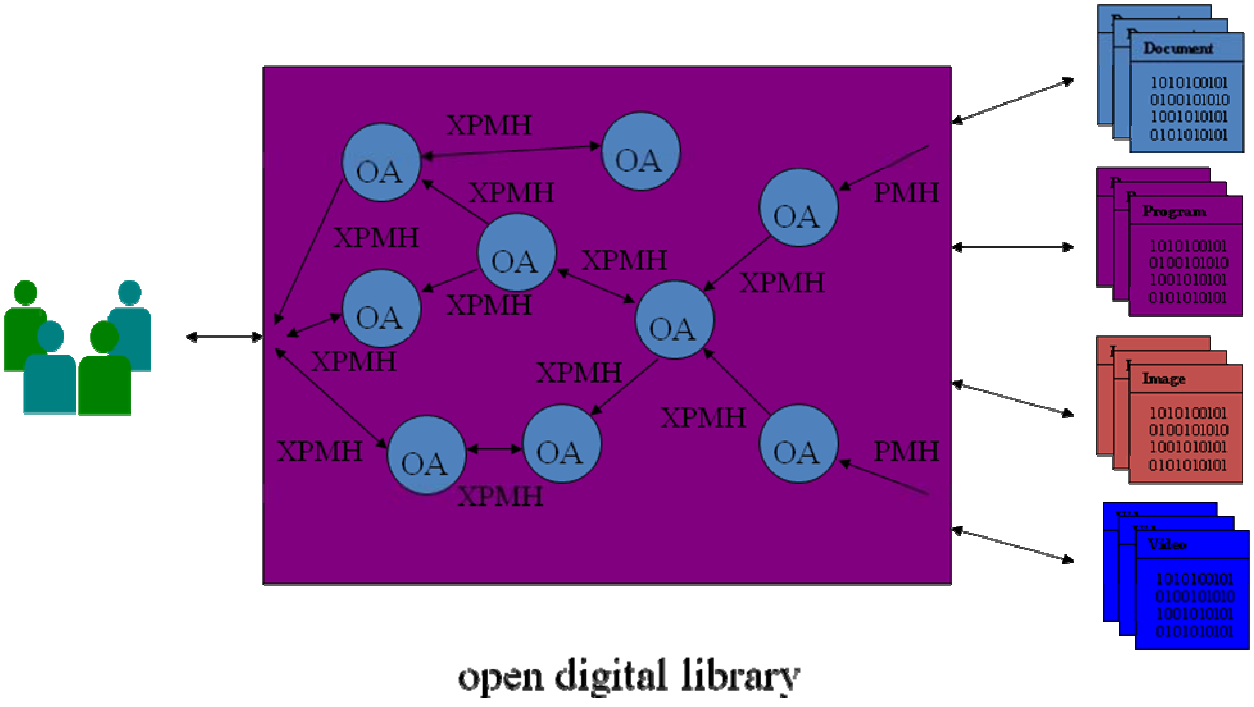

Figure 7. Open Digital Library Architecture

1. ODL architecture overview (please see Figure 7)

Components can be connected together so a DL can be constructed from modules. Figure 7 shows how an Open Digital Library (ODL) can connect users (see left) with content collections (see right), building upon OAI-PMH. Note that this work coincided with work on MARIAN but pre-dates the development of SOA, as explained above in connection with CiteSeer. The right side shows a relatively standard application of OAI-PMH to get to the metadata, and thence the data. The

center shows the modules. The left shows that one or more of the modules connects with the users.

In the middle, we see modules connected with XPMH. Suleman defined an extension of OAI-PMH and called it XPMH. It allowed modules to get input from other modules in similar fashion to OAI harvesting, and to provide output that could be used by other modules through a type of OAI data providing.

Some of the modules included: Filter, ODLUnion, ODLSearch, ODLBrowse, and ODLRecent. The DL-in-a-box project made these available, so DL architects could connect these key pieces of a DL together, along with other modules, to build a tailored system, such as under control of 5SGen (5S-based generator – see Rohit Dilip Kelapure, "Scenario-Based Generation of Digital Library Services", June 2003, Virginia Tech Computer Science MS thesis, Blacksburg, VA, USA, http://scholar.lib.vt.edu/theses/available/etd-06182003-055012/). The WS-ODL framework (see Roberto et al., in Resources), extended and updated ODL.

10. Resources

- Principles of DL architecture
  - Readings for students
    - Arms, W. Y. (1995). Key Concepts in the Architecture of the Digital Library. *D-Lib Magazine*, 1(1). http://dx.doi.org/cnri.dlib/july95-arms
    - Kahn, R., & Wilensky, R. (1995). A Framework for Distributed Digital Object Services. http://dx.doi.org/cnri.dlib/tn95-01
- Federated (networked) architecture
  - Readings for students
    - Marcos André Gonçalves, Robert K. France, and Edward A. Fox. "MARIAN: Flexible Interoperability for Federated Digital Libraries." *Research and Advanced Technology for Digital Libraries: [Proc. of] 5th European Conference, ECDL-01 (Darmstadt, Germany: 4-9 Sept. 2001)* Springer, 2001, pp. 173-186. http://www.dlib.vt.edu/reports/ecdl2001_8.pdf
- Distributed architecture
  - Readings for students
    - Rodger J. McNab, Ian H. Witten, Stefan J. Boddie, "A Distributed Digital Library Architecture Incorporating Different Index Styles (1997)," Advances in Digital Libraries. (Discusses MG and MR, related to the Greenstone system.) URL: http://www.cs.waikato.ac.nz/~ihw/papers/RM-IHW-SB-Adistributeddl.pdf
- Service Oriented Architecture (SOA)
  - Readings for students

        - Yves Petinot, C. Lee Giles, V. Bhatnagar, Pradeep B. Teregowda, Hui Han, Isaac G. Councill, "A Service-Oriented Architecture for Digital Libraries (2004)," (related to CiteSeer) at http://www.personal.psu.edu/staff/i/g/igc2/papers/petinot04service.pdf
- (Optional) Component-based DL architecture: Open Digital Libraries (ODL)
    - Reading for students
        - Hussein Suleman and Edward A. Fox (2001), A Framework for Building Open Digital Libraries, D-Lib Magazine, Volume 7 Number 12 at http://www.dlib.org/dlib/december01/suleman/12suleman.html
    - More resources for students (optional) and instructors
        - Virginia Tech's Open Digital Libraries project page, which contains project description, publications, tools, components and protocols and other resources of the component-based DL architecture. URL: http://oai.dlib.vt.edu/odl/
        - (WS-ODL: Overcoming ODL's limitations) Pablo A. Roberto et al., "A Web Services-Based Framework and a Wizard Tool for Building Componentized Digital Libraries" at http://www.lbd.dcc.ufmg.br:8080/colecoes/sbbd/2006/007.pdf
- (Optional) Readings for other architectures
    - William P. Birmingham (1995), "An Agent-Based Architecture for Digital Libraries," D-Lib Magazine article (emerging from the University of Michigan DLI-1 project) at http://www.dlib.org/dlib/July95/07birmingham.html
    - Carl Lagoze, Sandy Payette, Edwin Shin, Chris Wilper (2005), "Fedora: An Architecture for Complex Objects and their Relationships," *Journal of Digital Libraries, Special Issue on Complex Objects*, 6(2): 124-138. DOI: 10.1007/s00799-005-0130-3. Preprint available at http://arxiv.org/abs/cs/05010122
    - "NSDL Library Architecture: An overview" (2005) at http://*onramp.**nsdl**.org/eserv.php?pid=onramp:19&dsID=**nsdl**_arch_**overview**.pdf*
    - Xiaoming Liu, et al. (2002), "A Scalable Architecture for Harvest-Based Digital Libraries: The ODU/Southhampton Experiments," D-Lib Magazine 8(11) article at http://www.dlib.org/dlib/november02/liu/11liu.html

11. Concept maps (created by students)

Note: IHMC Cmap Tools is an open source client tool to create concept maps. CmapServer enables the users to collaborate and share concept maps anywhere on the internet. Both software can be downloaded freely for educational purposes from http://cmap.ihmc.us/download/index.php

12. Exercises / Learning activities

- Students form groups
- A specific DL architecture is assigned to each group
- Each group presents a DL architecture overview with its design ideas (in which situations the architecture is most appropriate), advantages, disadvantages, etc.

a. Group analysis of individual architectures

Prior to the class session, organize the students into groups. Each group will be assigned to investigate/analyze one of the architectures listed above:

- Federated architecture
- Distributed architecture
- Service-oriented architecture
- (Optional) Architecture not mentioned in the lecture such as Component-based DL architecture, peer-to-peer-based DL architecture, etc.

Each group will be expected to read more than just the assigned class readings, in order to fully understand the assigned architecture. At the next class period, each group will give presentation about an overview of what it has learned about the situations in which the architecture is most appropriate, its advantages and disadvantages, etc.

[Note for instructor: Rather than lecturing about the architecture, this learning activity could be expanded, so that the students are presenting the primary material. If that approach is taken, then the instructor should be prepared to fill in the gaps or correct any misstatements made in student group presentations.]

13. Evaluation of learning objective achievement

The group presentations described in section 12 could be graded to evaluate students' learning

- Group presentations might be evaluated in terms of their comprehensiveness (did they include the important features and characteristics of the architecture?), their clarity (did they explain the architecture in a way that it could be distinguished from the other architectures?), and the quality of the presentation (e.g., slide quality, presentation style, use of time, and Q/A session).

14. Glossary

Universal Description, Discovery and Integration (**UDDI**) is a platform-independent, XML-based registry for businesses worldwide to list them on the Internet. UDDI is an open industry initiative, sponsored by OASIS, enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet.

DC: The Dublin Core metadata element set is a standard for cross-domain information resource description. It provides a simple and standardized set of conventions for describing things online in ways that make them easier to find. The Dublin Core is widely used to describe digital materials such as video, sound, image, text, and composite media like web pages. Implementations of the Dublin Core typically make use of XML and can be Resource Description Framework based. The Dublin Core is defined by NISO Standard Z39.85-2007. (Based on excerpt from wikipedia.org)

OAI-PMH (*Open Archives Initiative Protocol for Metadata Harvesting*) is a protocol developed by the Open Archives Initiative. It is used to harvest (or collect) the metadata descriptions of the records in an archive so that services can be built using metadata from many archives.

Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with, and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

SOAP (Service Oriented Architecture Protocol) is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework that more abstract layers can build on.

Web Services have measurable preconditions and expectations.

Web Ontology Language (**OWL**) is a family of knowledge representation languages for authoring ontologies, and is endorsed by the World Wide Web Consortium. OWL is considered one of the fundamental technologies underpinning the Semantic Web, and has attracted both academic and commercial interest. (Excerpt from wikipedia.org)

OWL-S is a simplified ontology scheme, within the OWL-based framework of the Semantic Web, for describing Semantic Web Services. It will enable users and software agents to automatically discover, invoke, compose, and monitor Web resources offering services, under specified constraints (Excerpt from W3C at http://www.w3.org/Submission/OWL-S/)

15. Additional useful links

Interactive demo

a. Scottish Distributed Digital Library (2008) at http://scone.strath.ac.uk/sddl/index.cfm

b. Distributed Digital Library of Mathematical Monographs at http://mathbooks.library.cornell.edu:8085/Dienst/UIMATH/2.0/Search
c. European Digital Library (example of federated system) at http://www.edlproject.eu/

16. Contributors

Seungwon Yang (Virginia Tech, development), Dr. Edward A. Fox (Virginia Tech, development), Liz Liddy (Syracuse University, evaluation), Jerome McDonough (University of Illinois at Urbana-Champaign, evaluation), Ingeborg Solvberg (Norwegian University of Science and Technology-NTNU, evaluation), project members at the University of North Carolina at Chapel Hill provided additional comments.