Lightweight Integration of Documents and Services

Nkechi Nnadi and Michael Bieber
Information Systems Department
College of Computing Sciences — New Jersey Institute of Technology
University Heights, Newark, NJ 07102, USA
http://is.njit.edu/dlsi/

ABSTRACT
This research's primary contribution is providing a relatively straightforward, sustainable infrastructure for integrating documents and services. Users see a totally integrated environment. The integration infrastructure generates supplemental link anchors. Selecting one generates a list of relevant links automatically through the use of relationship rules.

Categories and Subject Descriptors
H.3.7 [**Information Storage and Retrieval**]: Digital Libraries – *systems issues*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *architectures*

General Terms: Design

Keywords: automatic link generation, metainformation, service integration, relationship rules

INTRODUCTION
This research provides a general method for integrating documents and services within Web systems through linking the interrelated elements and functions. While our approach is a general one that should apply to any Web application, thus far we have applied it primarily to digital libraries.

The Digital Library Integration Infrastructure (DLII) automatically generates links for digital library documents to related documents and services. Services include searching, providing annotations and peer review. Figure 1 presents an example from our current prototype, which can be accessed from our Web site. In addition to NASA's National Space Science Data Center (NSSDC) and the Arizona Document Summarizer, we currently have three preliminary, partial integrations of digital library systems within the National Science Digital Library [http://www.nsdl.org]. These include the AskNSDL "ask an expert" service, the Atmospheric Visualization Collection, and the Earth Science Picture of the Day system, as well as MapQuest, amazon.com and the NJIT library.

DLII supplements *documents* by linking them automatically to relevant services and related collections. DLII supplements *services* by automatically giving relevant objects in collections (and other services) direct access to these services. Users see a totally integrated environment, using their system just as before. However, they see additional link anchors, and when clicking on one, DLII will present a list of supplemental links. The next release of DLII will filter and rank order this set of generated links to user preferences and tasks.

The DLII infrastructure provides a systematic approach for integrating digital library systems, and by extension, any other document-based system with a Web interface. Systems generally require no changes to integrate with DLII.

DocEng'04, October 28-30, 2004, Milwaukee, Wisconsin, USA.

INTEGRATION INFRASTRUCTURE
Figure 2 presents the DLII integration infrastructure. To integrate a system that *requests* information (where DLII provides links), an analyst must write an integrator. To integrate a system that *provides* services (to or through which DLII-provided links lead), an analyst must declare relationship rules and/or register glossaries/thesauri.

(1) *Develop an Integrator:* An integrator sends DLII a preliminary list of elements eligible for linking within its system's documents and screens. DLII uses "structural analysis" (relationship rules) to generate links for each element.

Several integrator approaches exist. External approaches, such as wrappers and content analysis operate solely on the collection or service's output, and therefore require no changes to the system itself. This is especially useful for retrofitting DLII support to an existing system.

Internal approaches can generate element information as the screen or document is being produced, and either embed the information within it (e.g., using XML) or provide this information separately, perhaps through an application programming interface (API).

(2) *Declare Relationship Rules:* Relationship rules specify the "structural relationships" for automatically generating links for recognized object types within the system being integrated.

(3) *Register Glossaries/Thesauri:* DLII's lexical analysis engine uses a unified glossary of terms from participating systems to find key phrases associated with the glossary entries over the entire document or screen content. These become supplemental content-based links (not shown in Figure 1).

Most other systems could be integrated in the same manner as digital library collections and services.

DLII is a loosely coupled system, where various components communicate with each other via messages that conform to a well-defined standardized internal protocol. (Future versions will use the OpenURL standard [Van de Sompel & Beit-Arie 2001].) This approach allows new components to be developed and added without affecting existing components and functionality.

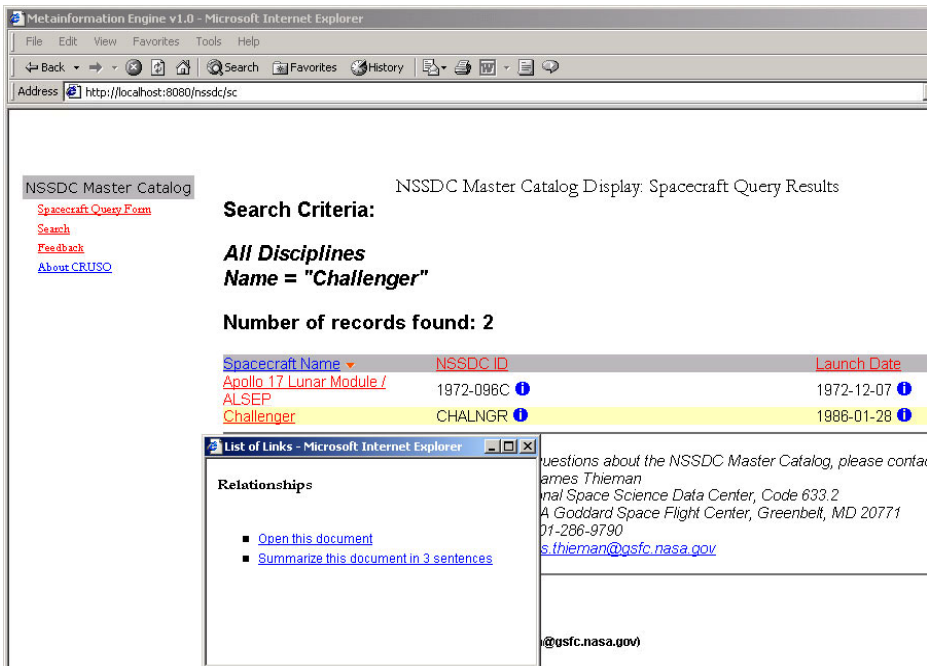

Figure 1: A screenshot of our current DLII prototype, integrating two independent digital library systems: NASA's National Space Science Data Center (NSSDC) Master Catalog and the Arizona Document Summarizer.

DLII is open source, written in JAVA. It currently runs on a LINUX machine using an Appache server.

The core DLII "engine" shown in Figure 2 consists of four primary components:

- The *Desktop* translates the displayable portion of DLII's internal messages, from the standard internal XML format to a format that can be displayed to a user via a Web browser (or other kind of user interface) and vice versa.
- The *Broker* enables the communication between the DLII engine modules. All DLII messages pass through the Broker, which then redirects them to the appropriate component.
- The *Relationship Engine* maps the system data and relationships to links at run-time. The Relationship Engine maintains a repository of relationship and metadata rules. When a screen is being sent to the DLII Desktop for display, the Relationship Engine retrieves all relevant rules for each element in that screen. The Desktop then converts the elements to link anchors and the relationships to links.
- The *Lexical Analysis engine* is based on Wu's Noun Phrase Extractor [Wu & Chen 2003]. It identifies key phrases, which it then compares to those in the registered thesauri and glossaries. Links to and/or within the thesauri' and glossaries' entries are added to the list of links that the Relationship Engine generates for that key phrase's element. (In future research, we shall incorporate other forms of content analysis for non-textual elements.) As lexical analysis is not a focus of this paper, we shall not describe it further.

A key characteristic of many elements of interest is that their identifiers are not the same as their display content. For example, a document or book may have a title that most people use, but several underlying services will match information and operations to its internal document identifier or ISBN. It is the job of the integrator to return the internal identifiers that services would use for elements of interest.

Information flows through DLII as follows. Assume the user asked a digital library collection to display a document. The collection's retrieval function will pass the document to its integrator. The integrator forms an XML message in DLII's internal format containing the document, along with the set containing the identifier, types and location within the document of its "elements of interest". The integrator sends this message to DLII. DLII's lexical analysis engine uses a unified glossary of terms from participating collections and services to find key phrases within the document's content associated with the glossary entries. DLII's Relationship

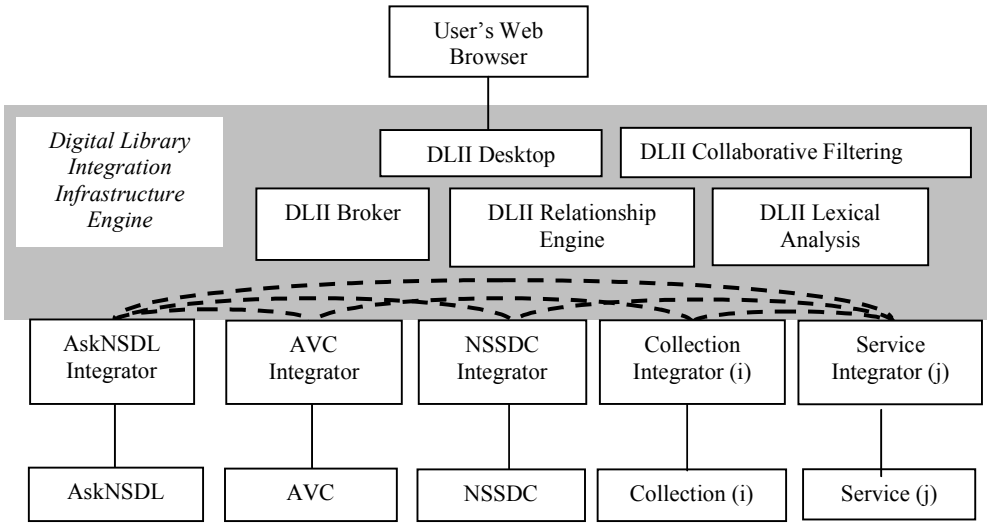

Figure 2: DLII Architecture. DLII is within the shaded area. The dashed paths indicate that once integrated, collections and services can share features through DLII links automatically. Integrated systems also continue to operate independently of DLII.

Engine adds link anchors for each element to a copy of the document, which is then passed to the user's Web browser for display. When the user selects any of these DLII anchors, the Relationship Engine uses the relationship rules to generate a filtered list of links, which it passes back to the Web browser. When the user selects one of these links, the appropriate set of commands associated with its relationship rule is passed to the associated collection or service. (For the second link in Figure 1, the DLII Relationship Engine would use the relationship rule presented below to generate a query to the Arizona Document Summarizer.)

Much More than Lexical Analysis

DLII generates the majority link anchors and links *automatically* through structural analysis using relationship rules. (Lexical analysis supplements these structural links.) If a system can operate on an element, DLII generates a link leading directly to this system's service. For example, given a discussion thread about a document, any time that document's identifier or title appears, DLII automatically detects this and adds an anchor over the document identifier or title.

Relationship rules define which relationships (links) should be available for which kinds of elements. Each relationship rule represents a single relationship for a single element class. As elements can have many relationships, each element class can have several relationship rules. Each element instance triggers the same set of relationship rules, assuming conditions are satisfied for each. In Figure 1, two relationship rules triggered for the "document" element (or more rules triggered, but DLII's collaborative filtering produced this customized list).

Because they operate at the "class" or "kind of element" level, each relationship rule works for every element of that class. E.g., the rule below applies to any "document" element found within any screen or document displayed.

As an example, in Figure 1, the relationship rule underlying the second link would include the following parameters:

- the element type (in this case "document")
- the link display label ("Summarize document...")
- relationship metadata (semantic type, keywords, etc., useful for filtering)
- the destination collection or service (in this case the "Arizona Document Summarizer")
- the exact command(s) to send to the destination system *("<http://keats.ecom.arizona.edu:8080/ebizport/serlet/ebizport/Summarizer.jsp?url=X&length=3*" where X is the document URL)
- any relevant conditions for including this relationship (including access restrictions)

Relationship rules are stored in an XML database. A recent research project called *xlinkit* [http://www.xlinkit.com] is the only system we know doing something similar. They express relationship rules in first-order logic, which we actually did in an early prototype [Bieber & Kimbrough 1994]. In future versions of DLII we hope to go back to this more flexible and powerful format, and will consider using *xlinkit* within an extended version.

DISCUSSION

A major longer-term research goal is developing a structure for providing users with comprehensive *metainformation* [Catanio et al. 2004, Galnares 2001]. The notion of metainformation expands on what people typically consider metadata. Whereas metadata often describes characteristics of an element of interest, surrounding relationships often point to other entities or documents, as well as to functions (services) that can be executed over aspects of that element. Metainformation includes structural relationships, content-based relationships, user-declared knowledge-sharing relationships, as well as the metadata around an element of interest [Catanio et al. 2004, Galnares 2001]. Combined, the metainformation goes a long way towards establishing the full semantics for (the meaning of and context around) a document's elements.

DLII's approach is entirely different from federated search and metasearch. The vast majority of DLII links are not found through searching. Instead they are specified through structural relationships. These are pre-specified through the relationship rules by element type.

In many ways, DLII is a link resolver service, in that it generates a set of relevant links to information resources, and when the user selects one, DLII forwards appropriate commands and parameters to have that information presented to the user. Link resolvers primarily link citations within traditional library systems to accessible copies of the cited document [Collins & Ferguson 2002; Vogt 2003]. DLII links among citation and non-citation sources as well as other document elements (e.g., headlines, photo captions, key words in paragraphs, geographic names). (Furthermore, we could integrate these existing link resolver systems as *metainformation providers* and *metainformation requesters*, bringing their services to users of other document systems and enriching them with the other types of links that DLII can provide.)

ACKNOWLEDGMENTS

We gratefully acknowledge support by the NSF under grants IIS-0135531 and DUE-0226075, and the UPS Foundation. DLII is part of the National Science Digital Library project.

REFERENCES

Bieber, Michael and Steven O. Kimbrough (1994), On the Logic of Generalized Hypertext, Decision Support Systems 11, North Holland, 241-257.

Catanio, Joseph, Nkechi Nnadi, Li Zhang, Michael Bieber and Roberto Galnares, "Ubiquitous Metainformation and the 'What You Want When You Want It' Principle," forthcoming in the Journal of Digital Information, 2004.

Collins, Maria D. and Christine L. Ferguson (2002). "Context-sensitive Linking: It's a Small World After All," *Serials Review*, 28(4), 267-282.

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

Van de Sompel, Herbert and Oren Beit-Arie (2001). Generalizing the OpenURL Framework beyond References to Scholarly Works: The Bison-Futé Model, D-lib Magazine 7(7/8).

Vogt, Sjoerd (2003). "Resolving the links," *Information Today*, 20(4), 25-26.

Wu, Yi-Fang and Xin Chen (2003). Extracting Features from Web Search Returned Hits for Hierarchical Classification. Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE'03), June 23 - 26, 2003, Las Vegas, Nevada, USA, page 103-108.