Towards Lightweight Digital Library Integration

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
The Digital Library Integration Infrastructure (DLII) provides a systematic lightweight approach for integrating digital library collections and services. Digital library systems generally require minimal or no changes to their code. Users see a totally integrated environment. They use their digital library system just as before. They also see extra link anchors. Selecting one generates a list of links to relevant metainformation (structural, content-based and knowledge-sharing relationships, and metadata). DLII generates the vast majority of supplemental link anchors and metainformation links automatically through the use of relationship rules. This paper presents the concept of metainformation, describes the DLII infrastructure and architecture, and explains how systems can integrate into the infrastructure. This research's primary contribution is providing a relatively straightforward, sustainable infrastructure for integrating digital library collections and services.

Keywords
digital library integration, metainformation, service integration, automatic link generation, National Science Digital Library

1. INTRODUCTION
The Digital Library Integration Infrastructure (DLII) provides lightweight digital library integration through automated linking. DLII supplements collections by linking them automatically to relevant services and related collections. DLII supplements services by automatically giving relevant elements in collections (and other services) direct access to them. Users see a totally integrated environment, using their system just as before. However, they will see additional link anchors. When clicking on one, DLII generates a set of supplemental links. DLII filters and rank orders this set to user preferences and tasks. DLII provides a systematic approach for integrating digital library systems (and by extension, any other system with a Web interface). Our approach is "lightweight" or "non-intrusive" and relatively "uncoupled" because integration requires little or no change to a system's source code. Collections and services can still operate independently of DLII after integration [Barrett et al. 1996].

Following are some examples of integration by linking. Multiple collections could be "virtually" integrated when a concept described in one (some) is linked to further explanations, examples, and teaching resources for that concept in other collections. Any collection could share a service enabling users to create a guided tour over elements in that collection. Services can execute over multiple collections; the guided tour could include documents from several collections. Services also can apply over other services. For example, one could create a guided tour leading users through the steps of a peer review process, and then annotate it.

Analysts also can take advantage of DLII to connect information and services in new ways within a single collection or service virtually, with very low effort using relationship rules (described in §3.3.2).

A major longer-term research goal is developing a structure for providing users with comprehensive *metainformation* [Catanio et al. 2004]. Metainformation includes the structural relationships (links based on element type), content-based relationships, knowledge-sharing relationships, and metadata around an element of interest. Combined, the metainformation goes a long way towards establishing the full semantics for (the meaning of and context around) a system'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 described in §3.3.2 by element type (i.e., link skeletons are pre-specified for any space mission, any document, any stock, etc.)

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 (LinkFinderPlus, SFX, 1Cate, etc.) using OpenURL protocols and Digital Object Identifiers (DOI) are used to bridge the gaps between the "silos of information" of separate library database systems. Link resolvers primarily link citations to accessible copies of the cited resource (although they also may link citations or citation elements to general web searches) [Collins & Ferguson, 2002; Vogt, 2003]. DLII links among citation and non-citation sources as well as other information elements (e.g., headlines, photo captions, key words in paragraphs, geographic

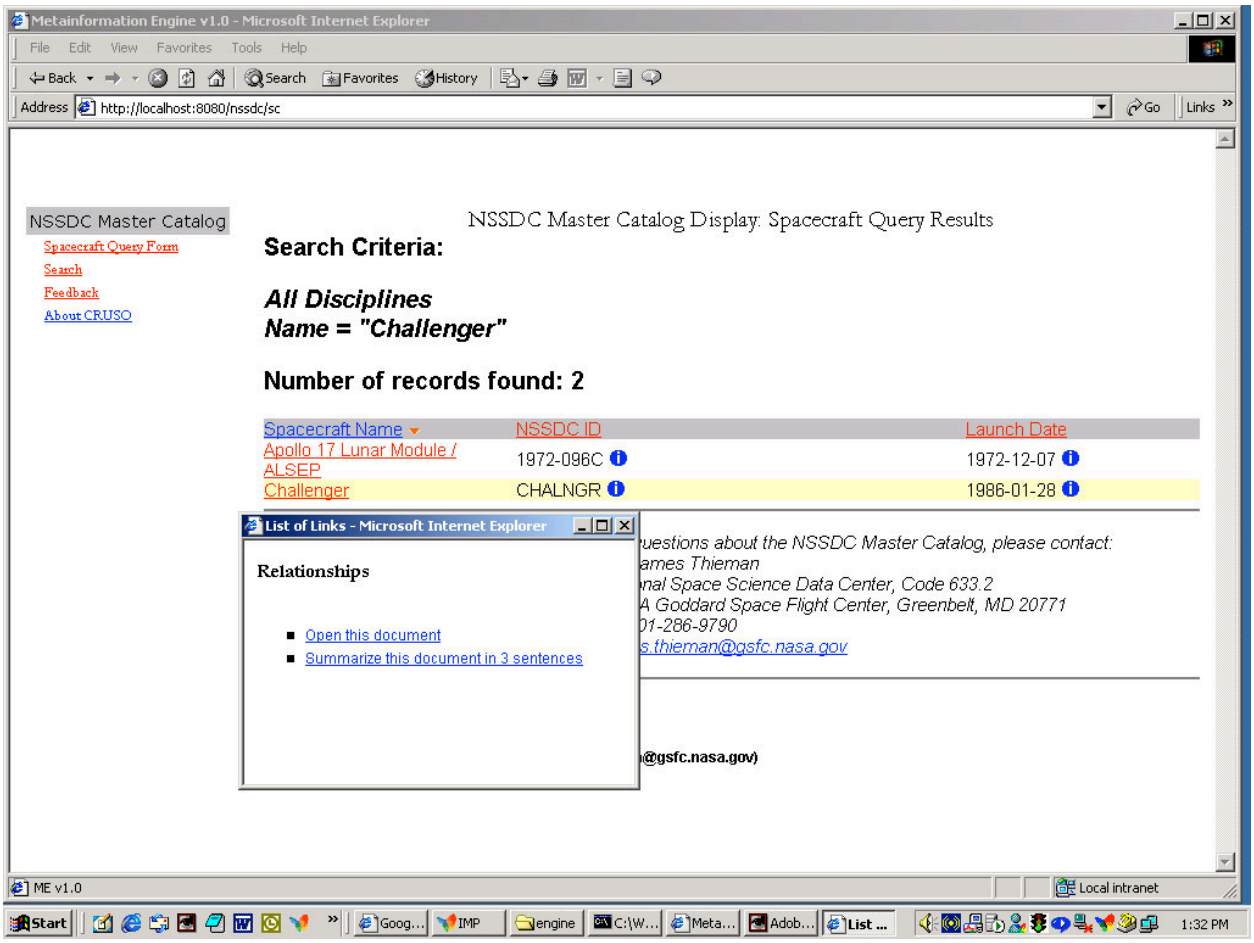

Figure 1: A screenshot of our current DLII prototype, integrating two independent digital library systems: NASA's National Space Science Data Center (NSSDC) Master Catalog and the Arizona Document Summarizer.

names)[1]. (Furthermore, we could integrate these existing link resolver systems as *metainformation providers* and *metainformation requesters* (described below), bringing their services to additional users and enriching them with the other types of links that DLII can provide.)

DLII's approach provides value over the cross-database linking provided by library vendors such as Serials Solution, EBSCO, WebFeat, and others. DLII provides seekers with links at the point of need within the native mode of the system or database being used that lead directly to related materials in an entirely independent system, without intervening steps or proprietary software.

Digital library systems should find several benefits to integrating through DLII:

- DLII virtually enlarges the size of a collection and the "feature set" or services that a system provides through links to related information and relevant services that external systems provide.
- DLII brings more users to a system because it has related information or relevant services.
- Users also become aware of other systems through seeing links to them within other systems. Similarly, DLII causes users to be aware of the kinds of information and services that are available, because of these links.
- DLII streamlines individual systems by providing direct access through links among a single system's information and functions, sparing the user from navigating through a possible series of menus.
- DLII's lightweight approach is cheaper in time and resources than other integration approaches, with minimal or no changes to a system's documents or code.

Figure 1 illustrates the current DLII research prototype [http://hynic.njit.edu/], integrating two digital library systems: NASA's National Space Science Data Center (NSSDC) Master Catalog [http://nssdc.gsfc.nasa.gov/] and the Arizona Document developed at the University of Arizona [Marshall et al. 2004]. Users make queries into the space science database from a query form. The NSSDC "wrapper" (one possible approach to integration—see §3.3.1) parses the query result, identifying NSSDC documents and launch date elements. DLII added

[1] The NISO OpenURL standard recently has become a standard for non-citation linking [Van de Sompel & Beit-Arie 2001], an approach that DLII and predecessor systems have taken from the very start [Bieber & Kimbrough 1992; Bieber 1998]. The next release of the DLII engine will use OpenURL as its internal as well as external standard protocol.

supplemental anchors on the document for these elements (indicated by the circled "i" in the 2nd and 3rd columns). The second column contains NSSDC document identifiers. The third column contains launch dates. The user clicked on a document anchor. DLII then inferred the list of links shown from its base of relationship rules for the kind of element selected. When the user clicks on the document identifier "CHALNGR," DLII generates a list of two links for document identifiers (for elements of type "document"). The first will prompt the NSSDC system to display this document. The second will prompt DLII to pass the CHALNGR document to the external Arizona Document Summarizer system. Clicking on (an element of type) launch date generates a separate list of links to services relevant to that kind of element.

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 [Silverstein 2003], the Atmospheric Visualization Collection [Klaus et al. 2002], and the Earth Science Picture of the Day system [Ruzek et al. 2002], as well as MapQuest.

We begin in §2 by further explaining the notion of metainformation. In §3 we present our general approach to integrating systems and DLII's integration infrastructure. §4 reviews the literature in systems and digital library integration, as well as linking engines. We conclude in §5 by discussion our plans for future research, our contributions and a vision of fully integrated digital library systems.

We note that the focus of this paper is on digital library integration. As such we shall not detail DLII's use of lexical analysis and collaborative filtering, even though this research makes contributions in each of these areas. Lexical analysis is used to for finding content-based relationships (see §2.2). Collaborative filtering customizes (prunes and rank orders) the set of links presented to users [Im & Hars 2001, Zhang & Im 2002].

2. METAINFORMATION

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 [Galnares 2001].

2.1 Structural Relationships

Structural relationships apply to an entire class of elements in an information domain. Structural relationships are inherent to the design or "structure" of the system. A database entity-relationship diagram, for example, contains structural links. The sets of links shown in Figure 1 represent structural relationships that apply to any document. Clicking on any launch date would generate a different set of links specific to that kind of element.

Structural links can connect the equivalent element, such as the same author or subject. They also can connect related elements (such as teaching materials on a particular subject or documents with a common author) or characteristics of an element (such as an author's address and background). The connected elements can be in the same or different systems. Thus a user may follow a link from an element in a specific system to a related element in a completely different system.

Structural links often can be thought of as services. Often the destination system will need to execute a command (e.g., a query) to retrieve, calculate or otherwise generate the destination element. This command will be associated with the link's *relationship rule* and executed if the user selects that link (see §3.3.2). Structural links generally are not found using search.

2.2 Content-based Relationships

Content-based relationships also contribute to understanding the context around an element of interest. While equally important, they are not necessarily fixed in the structure of the related systems. Instead they are based on the display content. For textual content, content-based relationships typically are found using one or more of the many lexical analysis techniques ranging from simple keyword search to cluster analysis. In §3.4 we briefly describe the approach that DLII currently uses, but many others are possible.

For multimedia content, much research is underway to automatically identify enough of the non-textual content to determine relationships. When alternate text tags provide metadata (such as the keywords on a photograph) or other identifying markup is available, then content-based relationships can be inferred for these surrogate representations.

Ontologies [McGuiness et al. 2002; Fensel et al. 2002], such as those being developed as part of the Semantic Web, could be used to generate structural relationships among key terms, found either through content-based or structural analysis. Part of understanding the context of an element could be to see how it fits within a web of related terms. A system supporting ontologies could easily be integrated within the DLII architecture and add such links to the list that DLII generates. The ontologies also can be used in a more traditional way to find related terms for expanding keyword search and other content-based approaches.

2.3 Knowledge-sharing Relationships

Knowledge-sharing relationships can be provided by authors within digital library systems. But they especially encourage users within the digital library's community to interact with each other and participate within the digital library, thus promoting the ideal from the hypermedia research community of the "reader as author" [Burton et al. 1995; Conklin 1987; Cotkin 1996; Miller 1995; Nielsen 1995].

Knowledge-sharing services allow people to create new kinds of metainformation that help customize the digital library for themselves and for others, conveying some knowledge they have about an element of interest. These can simply be personal annotations by users to themselves as a reminder or organizing device. Alternatively, they can be devices for sharing knowledge with others. (Each feature can have access

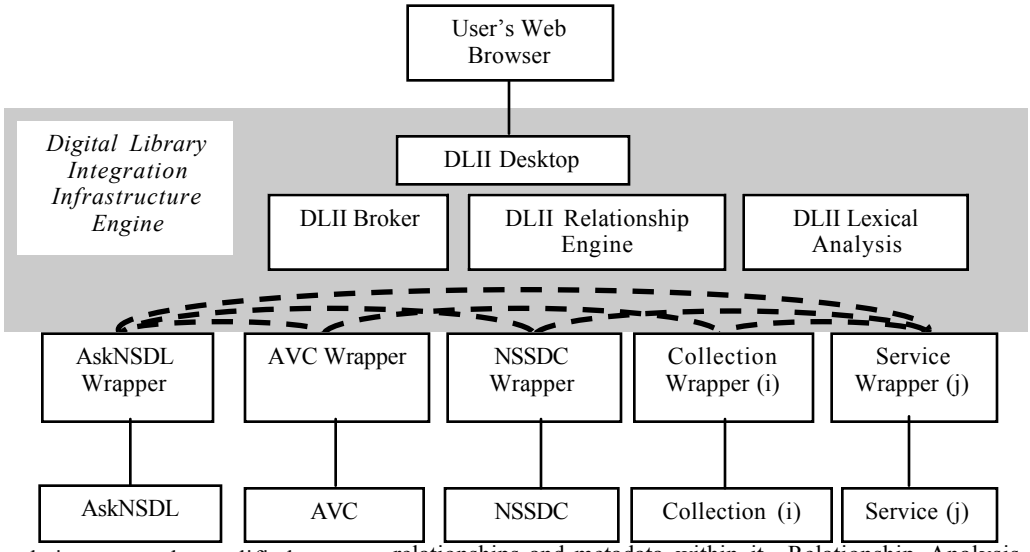

Figure 2: DLII High-level Architectural Overview. DLII comprises the shaded area. The dashed paths indicate that once integrated, digital library systems can share features through DLII links automatically. The digital libraries also continue to operate independently of the DLII. Wrappers, one possible integration approach, are included here to indicate our ideal that systems can integrate without change to their internal code. Other approaches are possible.

permissions specified, e.g., for being created, modified, deleted, linked to, and commented upon by an individual, work group or the general public.) Knowledge-sharing features include user-declared links, metadata, comments, discussions, bookmarks (favorites), overview maps, trails and guided tours, among others [Bieber et al. 2002].

In future research, we shall be incorporating existing knowledge-sharing services within DLII, as well as developing our own. We shall incorporate appropriate access permissions.

2.4 Metadata

Metadata is quite an active research topic in its own right, and mostly beyond the scope of this paper. Using *metadata rules* (see §3.3.2), DLII will allow any system to provide metadata for elements within another system.

In some of our earlier prototypes we experimented with displaying the metadata in a separate frame from that containing structural, content-based and knowledge-sharing links. When the user clicked on an element's link anchor, we would display the metadata for that element as well as generate a list of links to related items and services. However, we often found it difficult to decide whether to represent some information, such as an element's description or an annotation as metadata or a supplementary link. Future research includes both determining the best way to display metadata and deciding what should be displayed as metadata and what should be displayed as a structural relationship.

3. INTEGRATION APPROACH AND ARCHITECTURE

In this section we describe the general steps for integrating a system with the DLII infrastructure, and present an overview of our architecture.

3.1 Identifying Metainformation

System developers may wish to start by determining the kinds of metainformation their systems can provide. We have been developing a software engineering technique for analyzing an information domain and determining the relationships and metadata within it. Relationship Analysis is a systematic and rigorous elicitation technique to discover the relationship structure of the problem domain [Yoo and Bieber 2000; Yoo et al. 2004].

One begins by identifying the elements of interest for which one wants to provide structural relationships and metadata. For existing systems, one can look at screen shots to identify the elements of interest that a user might want to request metainformation about. When designing a new system, one can identify elements of interest (entities) from the use cases, a standard part of a formal systems analysis. For each element of interest, one asks a domain expert a series of questions to elicit characteristics about it and the relationships around it.

Relationship Analysis fills a major gap in today's software engineering techniques—the lack of a rigorous and comprehensive process to explicitly capture the relationship structure of the problem domain. Whereas other analysis techniques lightly address the relationship discovery process, Relationship Analysis provides the only systematic, domain-independent analysis technique focusing exclusively on a domain's relationship structure. Further description of Relationship Analysis lies outside the scope of this paper.

3.2. Integration Steps

Metainformation requesters send their display screens (documents, service interaction forms and results, or anything else to be displayed on the user's Web browser) for DLII to supplement the elements of interest with link anchors (akin to the "i" icon in Figure 1). The possible elements of interest within the screens/documents must be identified in some manner to DLII. It will then overlay a supplemental link anchor for each that it has at least one metainformation link available in a *relationship rule*. Later, when a user clicks on one of the supplemental link anchors, its underlying link sends a message to DLII. DLII generates a customized list of relevant links to documents and services for the corresponding structural, content-based and knowledge-sharing relationships and metadata. These links will send the appropriate URL or commands to the

corresponding *metainformation provider* systems to produce this metainformation.

The same system can be both a metainformation requestor and provider.

To integrate a digital library system that will provide metainformation, an analyst must register structural, knowledge-sharing and metadata *relationship rules*. Metainformation providers should also register any glossaries or thesauri that DLII's lexical analysis system can use to identify content-based relationships.

3.3 DLII Components and Architecture

Figure 2 presents a high-level architecture for DLII. We describe the major aspects that metainformation requesters and providers provide in this section, and the rest of the architecture in the section that follows.

3.3.1 Identifying Elements of Interest

Several approaches exist for identifying elements of interest within a metainformation requestor's screens/documents. External integration techniques, 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 integration techniques can generate element information as the screen/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).

DLII requires the set of element identifiers, element types and their locations within screens. Currently DLII has its own internal format for these specifications. The next release of the DLII engine will use OpenURL as its internal as well as external standard protocol. In addition, DLII's lexical analysis system will parse the display content using content-based analysis to identify additional elements of interest.

To date we have primarily used external wrappers, as we have been integrating existing systems over which we have no development access. We also are developing an interface to enable DLII to add links to systems using a Web services

3.3.2 Relationship Rules

Relationship rules capture the services that each system can provide for any kind of element. Relationship rules specify the structural, knowledge-sharing and metadata relationships for recognized element types within the system being integrated. This defines the integration that occurs virtually along the dashed paths in Figure 2. When a user selects the link anchor over an element of interest, DLII determines many of its links to related information and services from the relationship rules that metainformation providers have declared. (Content-based analysis determines the rest of the links.)

Relationship Analysis provides a systematic methodology for analyzing an information domain to determine its structural relationships and metadata. Integrators then can write relationship rules for each type of relationship or metadata found during the analysis.

Structural Relationship Rules

Each structural 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. For example, in Figure 1, the relationship rule underlying the second link would include the following parameters:

a) the element type (in this case "document")

b) the link display label ("Summarize document in 3 sentences")

c) any relationship metadata (as opposed to element metadata, which has its own rules; relationship metadata describes the relationship or link. It could include a link behavioral type (e.g., "query" or "computational"), semantic relationship type (e.g., "detail"), keywords, (which are useful for filtering links), and so forth [Oinas-Kukkonen 1998]

d) the destination target application system (the "Arizona Document Summarizer")

e) the exact command(s) to send to the destination system ("*http://nano.bpa.arizona.edu/nanoport/Summarizer.jsp?url=X&length=3*" where X is the document URL)

f) any relevant conditions for including this relationship (including the user types and tasks that would find this useful, level of expertise required, access restrictions, and so forth)

To emphasize the core idea behind relationship rules: because they operate at the "class" or "kind of element" level, each relationship rule works for every element of that class or kind. This means that the rule just described applies to any "document" displayed by any digital library collection or service.

In our current DLII implementation, 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 1992, 1994], but have not yet re-implemented, instead concentrating on other functionality. 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.

Metadata Rules

Most element metadata is structural, i.e., the same parameters exist for each instance of an element class. The goal behind metadata rules is to represent all kinds of structural metadata for an element class, especially when parameters for the same element can be gathered from different systems.

For example, in Figure 1, one hypothetical metadata rule that could underlie the document element would include the following parameters:

a) the element type (in this example, "document")

b) the metadatum display type ("author")

c) any metadata about this metadatum (semantic type ("name"), keywords, etc., for this metadatum itself)

d) the system providing this metadatum (the NSSDC's metadata repository)

e) the exact command(s) to send to this metadata provider

f) any relevant conditions for including this metadatum (including the user types and tasks that would find this useful, level of expertise required, access restrictions, and so forth)

Knowledge-sharing Relationship Rules

Knowledge-sharing services can be integrated through relationship rules. Whenever the user selects an element of interest, or a span of new text, the appropriate functionalities could be available.

The following hypothetical relationship rule could underlie a hypothetical "view comments" service link. It would be valid for every type of element, including documents. A condition check would confirm whether any comments already exist for this element, in which case it would be included in the list of links.

a) the element type ("generic_element")

b) the link display label (concatenate("view comments on this", element_type))

c) any relationship metadata (behavioral type = "command", semantic type = "annotation")

d) the destination system ("NSDL Core Annotation Service")

e) the exact command to send to the destination system (e.g., display_annotations(element_ID))

f) any relevant conditions for including this function (*check_condition("NSDL Core Annotation Service", existence_check("annotations", element_ID)) = true*)

3.4 DLII Architecture

DLII is a loosely coupled system, where various components communicate with each other via messages that conform to a well-defined standardized internal protocol. This approach allows new components to be developed and added without affecting existing components and functionality.

Figure 2 shows an overview of the DLII architecture. The core DLII "engine" 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 et al. 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 integration technique (wrapper, etc.) to parse displays and return the internal identifiers that services would use for elements of interest.

We next describe the information flow within DLII for a digital library collection utilizing wrappers. Many collections have a well-defined document format, and thus we can write a wrapper to identify their structural elements. Alternatively, within the wrapper we could specify a template for each publication source within the collection (newspaper, journal, conference, etc.) that has its own consistent layout. Then if we can identify the source, we can apply its predefined template.

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 wrapper. The collection's wrapper parses the document to identify possible elements of interest. First, the wrapper does a structural analysis; in the case of a research article, it can easily identify the title, author, publication, sections, figures, etc. If the article included XML markup, further elements could be identified easily. The wrapper forms an XML message in DLII's internal format containing the document and all elements identified, along with their object types, which it passes to DLII. DLII's lexical analysis engine uses a unified glossary of terms from participating collections and services to find key phrases associated with the glossary entries. DLII's Relationship 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 2, the DLII Relationship Engine would use the relationship rule presented earlier to generate a query to the Arizona Document Summarizer.)

4. RELATED RESEARCH

In this section we present related research concerning systems integration, as well as digital library integration and linking engines.

4.1 Systems Integration

In order to provide value to a user, systems often need to share application data or services. System Integration (SI) enables the cooperation of multiple software modules. It encompasses a host of activities that are aimed at accessing data and programming logic in an environment characterized by distributed heterogeneous systems. The goal of SI is to utilize various autonomous systems in concert so that they support the achievement of a common goal, by providing an integrated set of data and services [Barrett 1996]. It is often difficult to successfully carry out systems integration with systems that were developed independently with no thought to future integration [Nilsonn, 1990]. In what follows, we contrast DLII's lightweight integration approach to some common middleware architectures for systems integration.

Many systems are not designed to provide open and easy access to their data or functionality. This is often the problem faced when systems to be integrated belong to different autonomous organizations, as is the case with some digital libraries. Intra-organizational integration provides an opportunity to enforce some degree of standardization or compliance to the systems that must be integrated. However, integration architectures for inter-organizational systems are not able to impose such constraints. They therefore must rely on a mechanism that allows systems to cooperate without the need for compliance with a rigid set of standards or protocols. This is why loosely coupled or lightweight systems integration approaches in distributed, heterogeneous, and independent collections of systems like digital libraries are needed to provide ubiquitous information to users.

The technology that provides integration between systems is often referred to as *middleware*. Integration technology is the mechanism that allows communication and the transfer of data among systems. It also allows one system to initiate actions on another system. This is the "technical" aspect of system integration that focuses on the applications and protocols utilized to enable communication between systems [Nilsonn et al. 1990]. Middleware solutions provide standard programming interfaces and protocols that mask the complexity of networks and lower-level protocols [Bernstein, 2000]. The various types of middleware include transactional, message-oriented, procedural, and object-component or object broker middleware [Emmerich 2000].

Transactional middleware applies the concepts of transactions from databases. DLII does not provide support for transactions, as it is unnecessary at this point in integrating digital libraries.

Message oriented middleware enables clients and servers to communicate using messages and message queues. The client sends a message, which includes the service parameters, to the server component by inserting it into a message queue. The server responds to the client request with a reply-message containing the result of the service execution [Emmerich, 2000, Cummings, 2002]. Message systems rely on an asynchronous communication mode, which is not appropriate for timely information on the World Wide Web. In DLII, on the other hand, service invocation and fulfillment is handled by relying on the HTTP request/response model. When a client makes a request, DLII routes it to the appropriate service and returns the results of that service to the requester. (However, DLII in some ways is similar to the Java Messaging Service (JMS) publish and subscribe mechanism.)

Procedural middleware utilizes remote procedure calls (RPC). RPC requires a rigid architecture where the client system (metainformation requester) is tightly coupled to the service provider because of the need to hard-code the provider's server interface. When the server interface changes, the client system must also change and implement the new interface definition. In DLII the client is unaware of which server fulfils its request. Instead it simply registers to receive services that operate on an element of interest. The service provider's integration technique (e.g., its wrapper) is responsible for handling changes to the server's system. If the service provider changes its interface then its integration technique must be adapted appropriately. However, all these modifications are transparent to the client who may still invoke the service by clicking on a DLII link.

In the object and component architecture or object broker architectures, services advertise themselves in the service registry. Clients query the registry for service details and interact with the service using those details. This approach does not provide support for relationships between information items or objects (i.e., elements of interest). Instead it focuses on matchmaking between client and servers based on the functional interfaces described in the service registry. In contrast, the DLII approach is organized around structural and object relationships among elements of interest. The equivalent of a service description registry in DLII is the relationship rules repository, which specifies the services that are available for various element types rather than the description of parameters to be sent and methods to be initialized. When a service is requested, the link DLII generates is responsible for invoking the service is activated. (This may pass through an external component such as a wrapper or invoke the service directly.) This eliminates the need to exchange interface descriptions between client and server. Instead the integration technique handles the complexity of this interaction.

4.2 Digital Library Integration and Linking Engines

Very little research appears to exist concerning digital library integration, which is quite different from content interoperability. Similarly, few "engines" exist for the automatic generation of link anchors and links.

SDLIP

SDLIP (Simple Digital Library Interoperability Protocol) acts as search middleware for digital libraries. It provides a means for applications to access data from information sources easily. It handles the transport of queries and results, as well as negotiation of parameters for searches. A client may interact directly with an information source using SDLIP if that information source provides SDLIP access. Otherwise, the client utilizing SDLIP interacts with a proxy that communicates with the information source in its native

protocol or API, and the proxy then performs the required translation and conversion between client and information source [Paepcke et al. 2000].

In contrast, DLII does not implement or specify any protocol that requires compliance by clients and service providers (or their proxies). Client applications communicate with DLII through HTTP by clicking on a link to a service or data. DLII routes the client's request to the integration technique that communicates with the service. The technique is then responsible for communicating with the provider using its native protocols and APIs, and converting the results of the interaction in a format the client can understand (typically HTML or XML).

SFX

SFX [Van de Sompel 1999a,b,c; Van de Sompel & Beit-Arie 2001a,b] is one of the only approaches we have found that are similar to DLII. Both SFX and DLII can generate a set of context-sensitive links. SFX operates primarily within citation environments. When the user clicks on an SFX-button (anchor) inserted by a citation, SFX looks up destination collections that contain the cited resource from a set of registered collections. SFX uses the OpenURL protocol to record metadata parameters for citations [Van de Sompel et al. 2000]. While primarily used with traditional on-line library resources, SFX's approach could relatively easily be generalized to include links to other kinds of services and collections, and its developers are beginning to explore this [Van de Sompel & Beit-Arie 2001b].

Collections and services can automatically insert SFX-buttons into citation lists. It seems that they primarily generate these anchors at the same time they generate query results from a search. In contrast, DLII's approach requires few or no changes to the collection or service. (Nothing precludes applications from post-processing documents as DLII does to include SFX-buttons; we just have found no published evidence that any do this yet.) DLII also could integrate SFX as a citation service (or supplemental link resolver service).

LinkFinder, LinkFinder Plus [Samuels 2001] and 1Cate are link server products similar to SFX.

Other Linking Approaches

A few commercial products have appeared which generate anchors and links, such as NBC-Interactive's now defunct QuickClick [http://www.quickclick.com] and Microsoft's discontinued Smart Tags. QuickClick finds its links in two ways. First, it has a list of predefined keywords that it recognizes and a set of relevant links for each, such as company names. Second, it finds relevant links through a standard keyword search. Smart Tags [Kaminski 2001] similarly is content-based, utilizing a predefined set of words it recognizes or words that match a particular expression such as the format for legal citations. It provides not only links to sites on the Web, but also the ability to be customized to invoke services on applications. However, the Smart Tags recognizer is embedded in the user's Web browser. Both concepts are much less powerful than what we propose. In addition, unlike Smart Tags we provide an independent layer, separate from both the client's browser and the system providing information or services.

DLII also differs from most hypermedia linking engines [Anderson 1997, Carr et al. 1998, Davis et al. 1992, Grønbæk & Trigg 1999], which rely on user-declared manual linking or keyword search.

A few hypermedia engines execute independently of an application with minimal modifications to it, and provide the application's users with hypermedia support. Few approaches provide transparent hypermedia integration as DLII. Notable projects include Microcosm's Universal Viewer [Davis et al. 1994], Freckles [Kacmar 1993, 1995], the Distributed Link Service [Carr 1995] and the OO-Navigator [Garrido & Rossi 1996; Rossi et al. 1996].

Microcosm's Universal Viewer and Freckles seamlessly support an application's other functionality but provide only manual linking. OO-Navigator comes the closest to our approach, providing a seamless hypermedia support for computational systems that execute within a single Smalltalk environment [Garrido and Rossi 1996]. This approach meets our goal of supplementing Smalltalk applications with hypermedia support without altering them. Our approach applies to any Web-based service or collection.

Lastly we note that several Web sites automate linking for database applications. For example, electronic shopping applications will fill a template such as a catalog page, adding specific links for whichever product appears there. Several digital library query engines similarly add links to query results returned from a database. DLII provides a generalized approach that will function beyond database applications. Also, anchors within Web applications tend to have a single link, while anchors within DLII-enhanced systems typically produce a list of relevant links, each representing a different relationship on the anchor's element of interest.

5. FUTURE RESEARCH ISSUES

In the course of this research we anticipate resolving several issues. These include:

- *User Interface Studies:* We shall conduct user interface studies to determine the best ways to display metadata, as well as link anchors, links and lists of links to metainformation
- *Where to store service data:* If a service has been designed for a particular collection, can it now store service data (e.g., comments and peer reviews) for other collections? If not, the service's integration technique may have to store data for the external ones.
- *Service customization:* Some services may need customization to serve new audiences. Several digital library services, such as annotations and guided tours, would work over most collections and services [Harnad, 1996, 1999]. Other services need to be customized to particular communities. One example is peer review. Different communities require different numbers of reviewers, use different methods for assigning reviewers, handle the evaluation of reviewer comments differently, etc. These also may vary depending on the element type, such as a teaching resource versus a journal article. Therefore, any general peer review service [Harnad,1997] would have to accommodate all of these differences. This leads to the opportunity in our future research to

develop a general approach to developing customizable services.

- *Dynamic Service Support:* Some services may be designed to work over static documents in a collection. They now may have the opportunity to serve collections or services where documents are generated in response to a user request, such as placing a comment over a list of documents returned by a search query. When this query result is regenerated at a later time, we should be able to locate the comment over the same element(s). Zhang [2005] discusses such "dynamic" service features further.
- *Service Combination:* We hope to explore creating new services from existing ones. For example, we could string together the summarizer service with a standard Web language translation service to create a new "summarize in my language" service. Selecting a text passage would first have DLII direct the Arizona Document Summarizer to summarize it, and then the translation service to translate the summary.

6. CONCLUSION

DLII research makes several contributions:

- an architecture for lightweight integration of digital library systems through linking
- a systematic approach to integrating digital library collections and services, with other appropriate collections and services
- applying the concept of metainformation to digital libraries for specifying the structural, content-based and knowledge-sharing relationships (as well as metadata) around elements of interest

This research provides a new means for making digital library services interoperable. Developers should consider designing their collections and services in a way that makes them available to a broad community:

- Developers should program them in a way that makes it easy to integrate. (In many cases this simply means providing a large amount of metadata in their documents and screens using XML, which already is a trend.)
- Developers should make services easy to customize.
- Developers should check whether existing external services satisfy their requirements and can be used (possibly customized and potentially through DLII), thus sparing time and expense.

DLII should facilitate a virtual restructuring of digital libraries into broad "federated" digital library spaces constructed from numerous interrelationships. Elements will reside within a rich context of metainformation that help users understand and work with them. This provides a ripe environment for organizations and individual people to develop small, specialized collections and services, which automatically become part of the federated space and accessible to those they can benefit. DLII extends the boundaries of how we think about and interact with digital libraries.

ACKNOWLEDGMENTS

We wish to thank Hsinchun Chen and Byron Marshall at the University of Arizona; Gail Hodge, Janet Ormes and Walt Truszkowski at NASA's Goddard Space Flight Center; Chris Klaus at the Argonne National Labs; Ken Anderson at the University of Colorado at Boulder; Martin Ruzek at the Universities Space Research Association; and Joanne Silverstein, Jeff Rubin and Mark Frantz at the University of Syracuse and Internet Consulting Services, Inc. for their assistance in integrating their systems with DLII. Anirban Bhaumik of OuterForce Systems helped design and implement the DLII architecture. We gratefully appreciate partial funding support for this research by the United Parcel Service, NJIT, and the National Science Foundation under grants IIS-0135531 and DUE-0226075.

REFERENCES

Anderson, K. (1997). Integrating Open Hypermedia Systems with the World Wide Web. Hypertext'97 Proceedings, ACM Press, New York, NY, 157-166.

Barrett, D.J., et al. (1996). A Framework for Event-Based Software Integration, ACM Transactions on Software Engineering and Methodology, 5(4).

Bernstein, P.A., Middleware: A model for distributed systems services, Communications of the ACM, February 1996, 39(2).

Bieber, Michael (1998), "Hypertext and Web Engineering," ACM Hypertext'98 Proceedings, ACM Press, Washington, D.C., June 1998, 277-278.

Bieber, Michael, Douglas Engelbart, Richard Furuta, Starr Roxanne Hiltz, John Noll, Jenny Preece, Edward Stohr, Murray Turoff and Bartel Van De Walle. (2002). Towards Virtual Community Knowledge Evolution," Journal of Management Information Systems 18(4), 11-36.

Bieber, Michael and Steven O. Kimbrough (1992), On Generalizing the Concept of Hypertext, Management Information Systems Quarterly, 16(1), 77-93.

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

Burton, John K., D. Michael Moore and Glen A. Holmes (1995). Hypermedia Concepts and Research: An Overview. Computers in Human Behavior 11: 345-369.

Carr, L. A., De Roure, D., Hall, W., Hiil, G. (1995). The Distributed Link Service: A Tool for Publishers, Authors and Readers. Proceedings of the 4th International World Wide Web Conference, Boston, Massachusetts, December 11-14, 1995.

Carr, L. A., Hall, W.; Hitchcock, S. (1998). Link Services or Link Agents?, Proceedings of ACM Hypertext '98, Pittsburgh PA, 113-122.

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.

Conklin, Jeff (1987). Hypertext: a Survey and Introduction, IEEE Computer 20(9), 17-41.

Cotkin, George (1996). 'Hyping the Text': Hypertext, Postmodernism and the Historian. American Studies 37: 103-116.

Cummings, F. (2002). Enterprise integration: an architecture for enterprise application and systems integration, New York: John Wiley & Sons.

Davis, H., W. Hall, I. Heath, G. Hill, and R. Wilkins. (1992). Towards an integrated information environment with open hypermedia systems, Proceedings of the ACM Conference on Hypertext (Milan, Nov. 1992) 181-190.

Davis, H., S. Knight and W. Hall (1994). Light Hypermedia Link Services: A Study of Third Party Application Integration," Proceedings of the Fifth ACM Conference on Hypermedia Technologies, Edinburgh, Scotland, September 1994, 41-50.

Emmerich, W. (2000) Software Engineering and Middleware: A Roadmap, Proceedings of the conference on the future of software engineering.

Fensel, D., Van Harmelen, F., Horrocks, I., McGuinness, D., Patel-Schneider, P., OIL: An Ontology Infrastructure of the Semantic Web, IEEE Intelligent Systems, Vol. 16(2), 2002

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

Garrido, A.; Rossi, G. (1996). A framework for extending object-oriented applications with hypermedia functionality, The New Review of Hypermedia and Multimedia 2, 25-41.

Grønbæk, K. and R. Trigg (1999). From Web to Workplace: Designing Open Hypermedia Systems, MIT Press.

Harnad, Stevan. "Free at Last: The Future of Peer-Reviewed Journals." D-Lib Magazine 5 (December 1999). http://www.dlib.org/dlib/december99/12harnad.html

Harnad, Stevan. "Implementing Peer Review on the Net: Scientific Quality Control in Scholarly Electronic Journals." In Scholarly Publishing: The Electronic Frontier, ed. Robin P. Peek and Gregory B. Newby, 103-118. Cambridge, MA: The MIT Press, 1996.

Harnad, Stevan "Learned Inquiry and the Net: The Role of Peer Review, Peer Commentary and Copyright." Antiquity 71 (December 1997): 1042-1048.

Im, Il and Alexander Hars, "Finding information just for you - Knowledge reuse using collaborative filtering systems," ICIS Conference, December 2001, New Orleans.

Kacmar, C. (1993). Supporting hypermedia services in the user interface, Hypermedia 5(2), 85-101.

Kacmar, C. (1995). A process approach for providing hypermedia services to existing, non-hypermedia applications, Electronic Publishing: Organization, Dissemination, and Distribution 8(1), 31-48.

Kaminski, C., Much Ado about Smart Tags, [on-line] http://alistapart.com/articles/smarttags/, July 2001.

Klaus, C.M., K. Andrew, G.G. Mace, "Atmospheric Visualization Collection: Developments in the NSDL", Journal of Digital Information, May 2002.

Marshall,B.,McDonald,D.,Chen,Hsinchun,Chung,Wingyan (2004). EBizPort: Collecting and Analyzing Business Intelligence Information. Journal of the American Society for Information Science and Technology (JASIST) Special Issue on Document Search Interface Design for Large-scale Collections and Intelligent Access, Forthcoming.

McGuinness, D., Fikes, R., Hendler, J., Stein, L., DAML+OIL: An Ontology Language for the Semantic Web, IEEE Intelligent Systems, Vol. 17(5), 2002.

Miller, J. Hillis. (1995). The Ethics of Hypertext. Diacritics 25.3: 27-39.

Nielsen, Jakob. (1995). Multimedia and Hypertext: The Internet and Beyond. Boston: AP Professional.

Nilsson, E.G.; Nordhagen, E.K.; Oftedal, G. (1990). Aspects of Systems Integration, Proceedings of the First International Conference on Systems Integration, 23-26 April 1990.

Oinas-Kukkonen, Harri (1998). What is in a link? Communications of the ACM. 7; 41(7): 98.

Paepcke, A., Brandriff, R., Janee, G., Larson, R., Ludaescher B., Melnik, S., Raghavan, S., Search Middleware and the Simple Digital Library Interoperability Protocol, D-Lib Magazine, Vol 6(3), March 2000.

Rossi, G., A. Garrido and S. Carvalho (1996). Design Patterns for Object-Oriented Hypermedia Applications. Book chapter in: Pattern Languages of Programs II. J. Vlissides, J. Coplien and N. Kerth, eds. Addison- Wesley, 177-191.

Ruzek, M., J. Foster, D. Bowler, A. Kerr, "The Earth Science Picture of the Day (EPOD)" Eos Trans. AGU, 81(48), Fall Meeting Supplement, F287, 2000.

Samuels, H. (2001). The State of Linking Today and Endeavors Linking Solutions- A White Paper. [on-line] http://www.endinfosys.com/ENCompass/linkingwhitepapers.pdf

Silverstein, Joanne, May We Help You Find Something? AskNSDL!, Knowledge Quest on the Web, American Library Association, January - February 2003. [on-line] http://www.ala.org/Content/NavigationMenu/AASL/Publications_and_Journals/Knowledge_Quest/Back_Issues_Archives/Volume_31/Ask_NDSL.htm

Van de Sompel, Herbert and Oren Beit-Arie (2001a). Open Linking in the Scholarly Information Environment Using the OpenURL Framework, D-lib Magazine 7(3).

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

Van de Sompel, Herbert and Patrick Hochstenbach (1999a). Reference Linking in a Hybrid Library Environment, Part 1: Frameworks for Linking, D-lib Magazine 5(4).

Van de Sompel, Herbert and Patrick Hochstenbach (1999b). Reference Linking in a Hybrid Library Environment, Part 2: SFX, a Generic Linking Solution, D-lib Magazine 5(4).

Van de Sompel, Herbert and Patrick Hochstenbach (1999c). Reference Linking in a Hybrid Library Environment, Part 3: Generalizing the SFX solution in the SFX@Ghent & SFX@LANL experiment, D-lib Magazine 5(10).

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).

Van de Sompel, Herbert, Hochstenbach, Patrick and Beit-Arie, Oren (2000). OpenURL syntax description. Technical Report. [on-line: http://www.sfxit.com/openurl/openurl.html]

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

Wu, Yi-Fang and Xin Chen: 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.

Yoo, Joonhee and Michael Bieber, "A Relationship-based Analysis," *Hypertext 2000 Proceedings*, San Antonio, ACM Press, June 2000.

Yoo, Joonhee, Joseph Catanio, Ravi Paul and Michael Bieber (2004). Relationship Analysis in Requirements Engineering, forthcoming in the Requirements Engineering Journal.

Zhang, Li (2005). Just-in-Time Hypermedia. Ph.D. Dissertation, New Jersey Institute of Technology, forthcoming.

Zhang, Yi and Il Im, "Recommender Systems: A Framework and Research Issues," Americas Conference on Information Systems, August 2002, Dallas.