Integration of Heterogeneous Digital Libraries with Semi-automatic Mapping and Browsing: From Formalization to Specification to Visualization

Rao Shen[1], Naga Srinivas Vemuri[1], Ananth Raghavan[1], Marcos André Gonçalves[2], Divya Rangarajan[1], Weiguo Fan[1], Edward A. Fox[1]

[1]Digital Library Research Laboratory
Virginia Tech
Blacksburg, Virginia USA 24061

[2]Department of Computer Science
Federal University of Minas Gerais
Belo Horizonte – MB Brazil 31270-901

{rshen, nvemuri, ananthr, mgoncalv, divyar, wfan, fox}@vt.edu

ABSTRACT

In this paper, we formalize the digital library (DL) integration problem and propose an overall approach based on the 5S framework. We apply 5S to domain-specific (archaeological) DLs, illustrating our solutions for key problems in DL integration. We use ETANA-DL as a case study to describe the process of semi-automatically generating a union catalog and a unified browsing service in an archaeological DL. A visual schema mapping tool is developed for union catalog creation. A pilot user study aids tool evaluation. Our approach is further validated through application of a general browsing component to two integrated DLs.

Categories and Subject Descriptors

H.3.7 [**Information Storage and Retrieval**]: Digital Libraries

General Terms

Design, Theory, Experimentation

Keywords

Integration, Interoperability, 5S Theory, Schema Mapping, Visualization, Browsing Component

1. INTRODUCTION

Digital Libraries (DLs) are transforming research, scholarship, and education. DL research challenges exist at both the fundamental technology level and at the large-scale integration level. A decade of government and private funding of DL research projects has led to important results at the fundamental technology level. Successes with large-scale integration are arguably less evident [17]. Even the notion of "DL integration" is ambiguous in the sense that different approaches and proposed solutions exist. Work on DL integration focuses mostly on three issues [12]:

1) Distribution: e.g., geographical distribution of DL data;

2) Heterogeneity: differences at both the technical level (e.g., hardware platform, operating system, programming language, etc.), and the conceptual level (e.g., different understanding and modeling of the same real-world entities);

3) Autonomy: the extent to which the components are self-sufficient or operate as components in a larger hierarchy.

By "DL integration" we mean hiding distribution and heterogeneity, while at the same time enabling and making visible component autonomy (at least to some degree).

Many DLs belonging to different autonomous organizations were developed independently without thought of open and easy automated access to their data and functionality. The inability to seamlessly and transparently access knowledge across DLs is a major impediment to knowledge sharing. Hence, a goal of DL integration is to provide unified knowledge from island-DLs.

Challenges to DL integration are a direct result of DL characteristics. DLs are complex information systems due to their inherently interdisciplinary nature, both with regard to application domains and technologies involved in building the systems. Concerning the latter, we must integrate findings from disciplines such as hypertext, information retrieval, multimedia services, database management, and human-computer interaction [7]. Hence, an integrative theory for DL is needed; [10] summarizes key early work on the 5S framework and our theory for DLs.

Interoperability is the most important issue when integrating heterogeneous DLs [1, 25, 30]. Since DL interoperability has many dimensions [25, 26] and has been the subject of many initiatives, the needs for DL integration are well known. Therefore, there are many opportunities to contribute. While numerous efforts have looked into the issues of interoperability amongst heterogeneous DLs, most developed their own approaches in an ad hoc and piecemeal fashion. This paper formalizes the DL integration problem and proposes an overall approach based on the 5S framework [10]. We then apply our framework to integrate domain-specific (archaeological) DLs, illustrating our solutions for key problems in DL integration.

Section 2 describes related work. Section 3 formalizes the DL integration problem. Section 4 presents our overall approach. Section 5 discusses how to generate a union catalog and integrated browsing service using our proposed approach for an integrated domain–specific DL in the field of archaeology. Section 6 concludes the discussion.

2. RELATED WORK

Related work concerning DL interoperability is summarized in a concept map (Figure 1). The two main approaches to interoperability are the intermediary-based and the mapping-based approach [27], which are both interrelated. The former depends on the use of intermediary mechanisms such as mediators, wrappers, agents, and ontologies. The mapping-based approach attempts to construct mappings between semantically related information sources. It is usually accomplished by constructing a global schema and by establishing mappings between the global schema and the local schemas. Approaches based on intermediaries may rely on mapping knowledge, domain-specific knowledge, or rules established by mapping-based approaches.

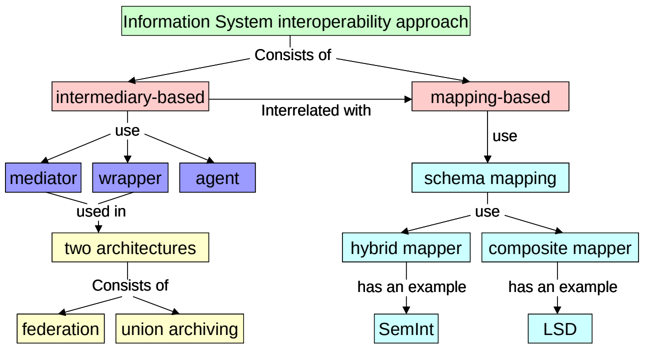


Figure 1. A concept map for related work

Within the intermediary-based approach are two architectures to deal with integration: federation and union archiving. Federation involves DLs sending search criteria to multiple remote repositories (e.g., using Z39.50 [21]). Results are gathered, combined, and presented. Federation is a more expensive mode of operation in terms of network and search system constraints; each repository has to support a complex search language and fast real-time response to queries. Union archiving involves gathering or harvesting data from sources and loading into a centralized data store. There are several schemes for harvesting data from heterogeneous sites, such as Harvest [2], OAI (Open Archives Initiative) [18], and SRU (Search/Retrieve URL Service) [22].

Other efforts that have explored interoperability amongst heterogeneous DLs include Dienst [3], InfoBus [24], and NDLTD [6]. Dienst, underlying the original NCSTRL (Networked Computer Science Technical Reference Library), provides for communications with services in a distributed DL. InfoBus is based on a hardware bus metaphor and was implemented with CORBA distributed object technology. It employed federation and high-level descriptions for mapping between different metadata standards. Humans developed mappings between metadata attributes of these standards. NDLTD provided semantic interoperability by adapting MARIAN [11] as mediation middleware. The MARIAN object-oriented data model is based on a semantic network of explicit nodes and links organized into a hierarchy of classes. This helps to join diverse harvested data into a single collection view for the user.

Schema mapping is typically performed manually – a tedious, time-consuming, error-prone, and expensive process. This led to aids to automate the process, such as Microsoft BizTalk Schema Mapper (http://www.microsoft.com/biztalk/) and Altova Mapforce (http://www.altova.com/products_mapforce.html). While fully automating the mapping process, to automatically generate wrappers, is generally infeasible, it is possible to partially automate the process, reducing human effort. A hybrid mapper uses multiple mapping criteria. Composite mappers are more flexible, combining multiple mapping results produced by different algorithms, including hybrid mappers [29]. A hybrid mapper typically uses a hard-wired combination of particular techniques that are executed simultaneously or in a fixed order. However, a composite mapper allows selection from a repository of modular mappers, and can extend the system when additional mappers are needed. SemInt [20] uses a hybrid mapper, and LSD [4] develops a composite mapper. In a future paper we will describe our approach to composite mapping, based on machine learning.

While many research projects developed semantic mediators and wrappers to address interoperability [23, 31, 43], few tackled the problem of (partially) automating production of these mediators and wrappers (which contain specific domain knowledge, such as mappings between source schema and the integrated schema). We develop a visual schema mapping tool within a formal framework to semi-automate schema mapping.

3. PROBLEM FORMALIZATION

Formalizing DL integration facilitates the development, comparison, and evaluation of solutions; makes clear to users what a solution means; and helps users evaluate the applicability of a solution. Furthermore, it allows us to leverage special-purpose techniques for the DL integration process. In this section, we first define inputs to the DL integration problem based on the 5S framework [10], and then explain the meaning of the output.

Notation: Let $DL_1, DL_2, \ldots, DL_i, \ldots, DL_n$ be n independent digital libraries; let Id_i be a unique identifier of DL_i; let C_{ij} be the j-th collection of DL_i; let $C_i = \bigcup_{j=1}^{m} C_{ij}$, where m is the total number of collections of DL_i; let $UnionC = \bigcup_{i=1}^{n} C_i$ be a union collection of the n DLs; let H be a set of universally unique handles.

Following [10] we have $DL_i=(R_i, DM_i, Serv_i, Soc_i)$, where R_i is a network accessible repository, supporting some type of harvesting protocol to expose its metadata; DM_i is a set of metadata catalogs for C_i; $Serv_i$ is a set of services, and Soc_i is a society.

- Definition 1: A Union Repository (***UnionRep***) of the n DLs is a DL repository ([10]) with a *getDL_Id* function:

  UnionRep *= (CollSet, getDL_Id, get, store, del), where*

  1) $CollSet \subseteq 2^{\{UnionC\}}$;

  2) $getDL\_Id: UnionC \rightarrow \{Id_1, Id_2, \ldots, Id_i, \ldots, Id_n\}$ maps a digital object *do* to the DL it belongs to.

  3) $get: H \rightarrow UnionC$ maps a handle h to $do=get(h)$;

4) *store: UnionC* $\times$ *CollSet* $\rightarrow$ *CollSet* maps *(do,* $\tilde{C}$ *)* to the augmented collection *{do}* $\cup$ $\tilde{C}$;

5) *del: H* $\times$ *CollSet* $\rightarrow$ *Collset* maps *(h,* $\tilde{C}$ *)* to the smaller collection $\tilde{C}$ *-{get(h)}*;

- Definition 2: A Union Catalog ***UnionCat*** *=DM*$_{UnionC}$ is a metadata catalog for *UnionC*.
- Definition 3: Minimal Union Services ***(MinUnionServ)*** = *{harvesting, mapping}* $\cup$ ($\bigcup_{i=1}^{n} Serv_i$). The *harvesting* service provides a mechanism to gather metadata from DL_i; the *mapping* service supports transforming information organized by local schema to information structured according to the global schema. The *harvesting* service is formally defined in [9]; the *mapping* is defined as follows (see definitions 4-7):
- Definition 4: A schema is a structure ([10]) with a domain D of data types (e.g., strings, numbers, dates, etc.). **schema** = *((V, E), L, F, D, M)*, where *(V, E)* is a graph with vertex set *V* and edge set *E*, *L* is a set of label values, *F* is a labeling function $F: (V \cup E) \rightarrow L$, and *M* is a function $M: V \rightarrow D$.
- Definition 5: Given a schema *((V, E), L, F, D, M)*, its element set = $\{(v, F(v))\} \cup \{(e, F(e))\}$.
- Definition 6: 1-1 mapping

  Let *S* and *T* be two element sets, of *S_Schema* and *T_Schema,* respectively. 1-1 mapping is a function: $M_{1\text{-}1}$: $S{\times}T \rightarrow Sim$, where $\forall sim \in Sim,\ 0 \le sim \le 1$. A tuple (*s, t, sim*) indicates element *s* of *S* is similar to element *t* of *T* with confidence score *sim*. The higher a confidence score, the more semantically similar are *s* and *t*.
- Definition 7: complex mapping

  Let *S* and *T* be two element sets, of *S_Schema* and *T_Schema,* respectively; let *O* be a set of operators that can be applied to elements of *S* and *T* according to a set of rules *R* to construct formulas; and let $Formu_s$ and $Formu_t$ be two sets of formulas constructed from the elements of *S* and *T,* using *O*. Complex mapping is a function: $M_{n\text{-}n}$: $(S \cup Formu_s) \times (T \cup Formu_t) \rightarrow Sim$, where $\forall sim \in Sim,\ 0 \le sim \le 1$.
- Definition 8: A Union Society ***UnionSoc*** = $\bigcup_{i=1}^{n} Soc_n$
- Definition 9: A Minimal Union Digital Library integrated from *n* DLs (see start of Section 3) is given as a four-tuple:

  MinUnionDL=(R_{union}, DM_{union}, Ser_{union}, Soc_{union}), where R_{union}, DM_{union}, Ser_{union}, Soc_{union} are Union Repository, Union Catalog, Minimal Union Services, and Union Society. A Union DL is a superset of a ***MinUnionDL***. "Integrated DL" and "Union DL" will be used interchangeably in this paper.
- Definition 10: DL Integration Problem Definition

  Given *n* individual digital libraries ($DL_1, DL_2, \ldots, DL_n$), each defined as described above, to integrate the *n* DLs is to create a Union DL.

4. APPROACH

4.1 Architecture of Integrated DL

As above (definition 9), an integrated DL is a 4-tuple consisting of a union repository, a union catalog, union services, and a union society. There are three popular integration architectures to deal with regarding the first two components of the definition, namely: 1) a centralized union catalog along with a centralized union repository; 2) a centralized union catalog for a decentralized union repository; and 3) a middle ground between the above two extremes of the spectrum, i.e., a centralized union catalog with a partially centralized union repository.

Decision on the architecture to be used to develop an integrated DL is based on 1) what contents (metadata, digital objects, or both) the DLs to be integrated would like to share; and 2) what the integrated DL wants to harvest. The former relates to copyrights and publication rights. The latter may consider issues such as scalability, consistency, and preservation.

Having both a centralized union catalog and a centralized union repository in an integrated DL can guarantee adequate performance at information seeking time. No burden is placed on the remote DLs to retrieve results. Storing digital objects in the integrated DL redundantly can help preservation. However, delivery of the most current information to users cannot always be guaranteed. Changes to the metadata and digital objects by the individual DLs need to be propagated to the integrated DL.

Assumed for a decentralized union repository is that the metadata contains links for concrete realization of digital objects. Its main disadvantage is that retrieval of digital objects relies on remote DLs. CITIDEL [28] is a DL that has a centralized catalog and decentralized repository; sustainability of the centralized portion of such a system also can be a challenge.

A partially decentralized union repository may store the digital objects that will not be changed frequently. The architecture of ETANA-DL [32-34] consists of a centralized catalog and partially decentralized repository.

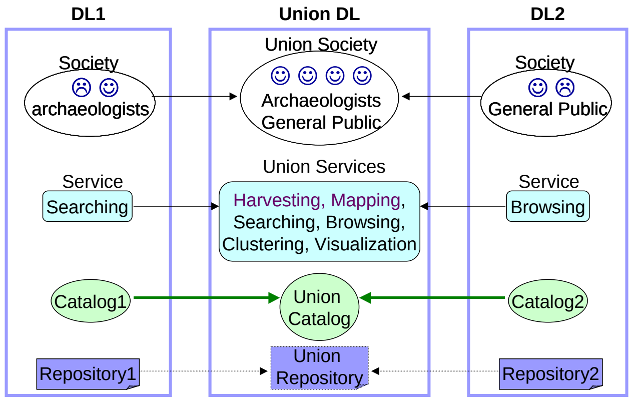


Figure 2. Architecture of a Union DL

Figure 2 shows the architecture of a union DL (with centralized catalog and repository) integrated from two DLs. To create a centralized catalog, the union DL must provide a harvesting service and a mapping service. Beside these two, the integrated union DL must provide all the services supported by the two DLs (i.e., searching and browsing), and other services (i.e., clustering and visualization). The visualization service integrates searching, browsing, and clustering. CitiViz [14] is an example of such an integrated service. It provides a visual interface to CITIDEL. Search results can be either clustered according to inter-document similarity or classified by predefined classes. Grouped documents are displayed in several ways to help browsing.

The union services illustrated in Figure 2 satisfy users of the two DLs, i.e., archaeologists and the general public. The user society in the integrated DL is a union of the users of the two DLs.

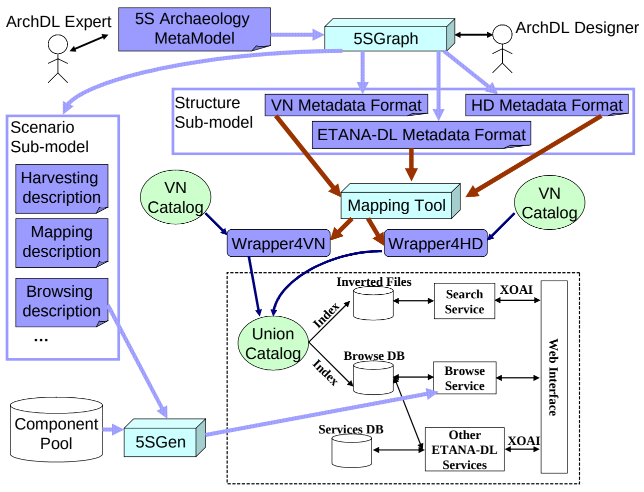


Figure 3. Generation of an Integrated Archaeological DL

5. UNION DL GENERATION CASE STUDY

With a better understanding of Union DLs (integrated DLs), we may use DL models based on the 5S framework to facilitate the process of building high quality integrated DLs.

We use a union Archaeological DL (ArchDL), ETANA-DL [32-34], as a case study. Figure 3 shows the process diagram for the generation of an integrated archaeological DL. At the bottom of Figure 3, a box with a dashed border describes the architecture of ETANA-DL. The centralized union catalog is indexed in two formats: as inverted files to provide Information Retrieval (IR) services, and as relational database to provide DB-supported services. The searching service uses the inverted files, whereas the browse service uses the relational DB to provide the dynamic, multidimensional browsing service. Future browsing and searching services may use both IR and DB support infrastructure. Other services may rely on geographic information system solutions, custom DBs, and the indexed archaeological data in the relational DB. Some of the services in the current ETANA-DL are pre-existing ODL [37] components, which communicate with each other and the web-interface using XOAI [36].

Though the current ETANA-DL prototype makes our ideas understandable, our objective for next generation implementation is to use the 5S framework and tools to cover the process of union ArchDL generation, including requirements gathering, conceptual modeling, rapid prototyping, and code generation. The 5SSuite tool we are developing consists of 5SGraph [41, 42], 5SGen [15, 16], and the visual mapping tool described in Section 5.1.1. It helps develop an integrated DL prototype as a proof-of-concept to justify and evaluate our DL integration approach.

An ArchDL metamodel encoded in 5SL [8] is developed based on its formal definition [35]. This metamodel is fed to the 5SGraph modeling tool (Figure 3). The ArchDL designer interacts with the 5SGraph tool to model the ArchDLs to be integrated (Virtual Nimirin (http://www.cwru.edu/affil/nimrin/menu/nimrin.htm) [5, 39], Halif DigMaster (http://www.cobb.msstate.edu/dig/lahav/) [13]], etc.), and the union DL (i.e., ETANA-DL). Each resulting ArchDL model contains a structure sub-model and a scenario sub-model as well as the other three sub-models (i.e., stream, space, and society sub-models). Metadata format is described in the structure sub-model, whereas services are described in the scenario sub-model. The mapping tool then semi-automatically creates a wrapper for each individual ArchDL. Each wrapper transforms the metadata catalog of its ArchDL to one conforming to the union metadata format. The results are stored in the union catalog. Section 5.1 describes the process to semi-automatically generate a union catalog; Section 5.2 shows how to produce an integrated browsing service.

5.1 Union Catalog Generation

Figure 4 shows how the metadata catalogs from two archaeological DLs, Virtual Nimrin [5, 39] and Halif DigMaster [13], are integrated into a union catalog.

Metadata catalogs from the two DLs are harvested by the Union DL. Each DL catalog conforms to its own metadata format (local schema), which is fed into the mapping tool together with the global schema. Two wrappers are automatically generated to transform the two catalogs conform to the global schema, for storage in the union catalog.

Figures 3 and 4 show the mapping tool playing a key role in catalog integration. Our visual schema mapping tool helps further automate the mapping process, so users interact with the system and provide feedback. It helps users find semantic relationships between schemas, exploiting human vision and spatial cognition.

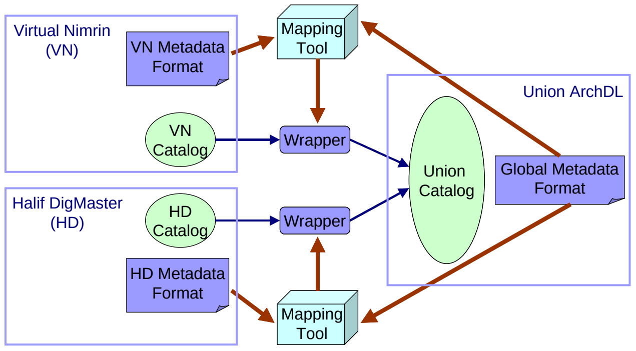


Figure 4. Union metadata catalog generation

5.1.1 Visual Mapping Tool

The ETANA-DL combines data from excavation projects like Nimrin, Umayri, and Lahav, about artifacts like Bones, Seeds, and Figurines. It merges the data into a global repository. As these archaeological collections may be stored in different formats, merging them involves mapping diverse structures of data to form a global representation encompassing all collections.

The process of schema mapping in the first ETANA-DL prototype was through code specific to each local schema, based on writing a specific mapping component for every new archaeological database to be integrated. Efficiency and reusability were low. Visual schema mapping can be of great help in such situations.

Schema mapping, so far, has been approached either from an algorithmic point of view or a visualization point of view for which commercial tools like MapForce (http://altova.com/products_mapforce.html) and BizTalk Mapper (http://microsoft.com/biztalk) are available. Clio [40] is another tool that uses reasoning about queries to suggest mappings between heterogeneous data sources and a target schema. However, most of these tools use a cumbersome outline view to display the hierarchical schemas. Through our visual mapping tool Schema Mapper, we solve key problems associated with the process of schema mapping by means of effective visualization of the schemas as hyperbolic trees and by using visual feedback to establish mapping relationships.

Hyperbolic trees provide context by laying out the hierarchy in a uniform way on a hyperbolic plane and mapping this plane onto a circular display region. The "fish eye" focus + context technique supports visualizing and manipulating large hierarchies [19]. In terms of navigation, studies have shown that hyperbolic trees help people to locate information 62% faster as compared to standard navigation methods [40]. Hyperbolic trees can be better by an order of magnitude as compared to traditional tree representation techniques, in terms of number of nodes displayed [41].

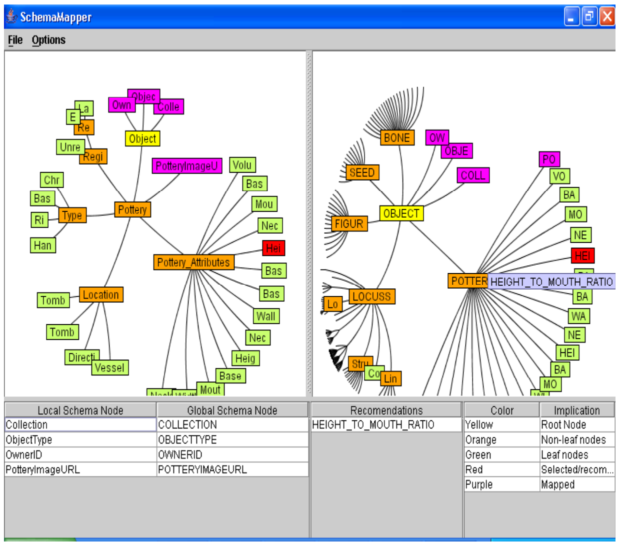
5.1.1.1 *Visualization System Overview*

The Graphical User Interface (GUI) serves as the main point of interaction between the users and our tool. Figure 5 shows a screenshot of the Schema Mapper GUI. The local and global schemas are visualized as hyperbolic trees, respectively, on the left and right. The hyperbolic tree representation has more nodes on the screen than a linear tree representation used in tools such as MapForce™. As each node in the hyperbolic tree representation is quite small and cannot show the entire name of the schema node it represents, the schema node name is available as a tool tip on each node.

Mapped nodes are shown in purple to distinguish them from unmapped ones. Further we use color to distinguish the root node (yellow), the non-leaf nodes (orange), and the leaf nodes (green). The color legend is shown in the bottom right hand corner of the screen. Only the leaf nodes in an XML schema contain data, hence only they can be mapped directly. Therefore, we do not allow mapping of non-leaf nodes. Non-leaf nodes can only be added as children of other nodes. The nodes which are selected or recommended appear red in color.

The GUI contains a recommendation table, with a list of all the recommended nodes for a selected local schema node. It also contains a mapping table which contains a summary of all the mappings in the current mapping session.

Through the GUI one can edit the global schema. This includes deleting or renaming a node, or adding a sub-tree of the local schema as a child of a node in the global schema. The user also is given the capability to undo mappings between nodes. The local schema is only to be mapped to the global schema, and so cannot be edited. After mapping all the nodes, a user can save the mappings through the Save menu option. This generates a style sheet with the mappings made in the current session.



Local Schema Node	Global Schema Node
Collection	COLLECTION
ObjectType	OBJECTTYPE
OwnerID	OWNERID
PotteryImageURL	POTTERYIMAGEURL

Recomendations
HEIGHT_TO_MOUTH_RATIO

Color	Implication
Yellow	Root Node
Orange	Non-leaf nodes
Green	Leaf nodes
Red	Selected/recom...
Purple	Mapped

Figure 5: Screenshot of Schema Mapper

5.1.1.2 *Architecture and Implementation*

A key design consideration, for future scalability and maintainability, was for the tool to be as componentized as possible. Figure 6 illustrates Schema Mapper's architecture.

The **Visualization Component** contains logic for generating hyperbolic trees and the different coloring representations to depict root, leaf, non-leaf, mapped, recommended, and unmapped nodes. Currently, the component displays schemas as hyperbolic trees. This representation scheme is extensible; other visualization techniques can support customization.

The **Recommendation Component** helps during mapping by recommending global schema nodes which are potential matches for a particular local schema node. When the user clicks on the local schema node to be mapped, the Component applies a name-based matching algorithm to find a set of potential matches (nodes) from the global schema. It returns these nodes to the Visualization Component to show in an appropriate manner. The Visualization Component highlights these recommended nodes in a different color (red) and also updates the recommendation table located in the middle of the bottom panel to display all the recommended nodes. This ensures that the user is aware of all the recommendations.

The user might choose one of these recommendations or a totally different node to map to. Once the user selects a global node, the two mapped nodes will change to purple, and the names of the nodes just mapped will be added to the mapping table located in the far left side of the bottom panel.

Mappings selected by a user are stored temporarily in a data structure by the **Mapping Component**. After the user decides to commit the mappings, the **XML Generation Component is called** to generate the mapping style sheet. There are two main generator parts:

1. *XSLT Style Sheet Generator*: When the user decides to save the mappings, the XSLT Style Sheet Generator gets the

temporary data structure with the mappings. It parses through the local and global schemas, and generates XSLT code that indicates the mapping of the local schema nodes to the appropriate global schema nodes. Once the style sheet has been generated, this component calls the XML File Generation Component to generate the output XML files.

2. *XML File Generator*: This takes in the Style Sheet and local XML file and creates an Output XML file with data from the local XML file, in the global XML format.

Schema Mapper is coded in Java. The hyperbolic tree library source code is available for free academic use at sourceforge.net.

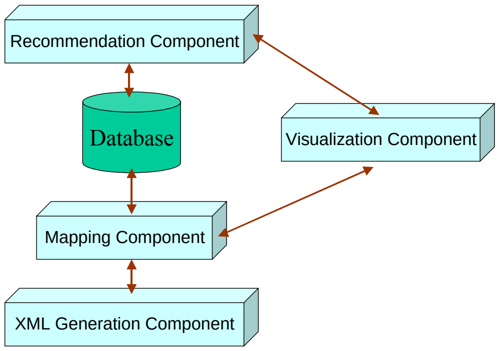


Figure 6: Schema Mapper Architecture

5.1.1.3 Evaluation

A **pilot study** was conducted to test the usefulness of the tool, especially with respect to visualization of the schemas as hyperbolic trees. We designed tasks to test the usefulness of the visual editing capabilities of Schema Mapper. Accordingly, the pilot tester was assigned two tasks, and was asked to repeat the same tasks on MapForce™, a commercial tool by Altova.

Task 1 required the user to map six nodes from a local schema to the global schema. Task 2 required the user to add a sub-tree from the local schema to the global schema and then do the mapping of the same six nodes as in Task 1.

Quantitative Results: For comparison, we measured the time taken to complete Task 1 using MapForce and using Schema Mapper. Table 1 shows actual times. SchemaMapper consistently outperformed MapForce in terms of task completion time. Also the users found that using lines to join nodes for mapping in MapForce was harder than clicking in Schema Mapper.

Another quantitative measure was the number of times users had to scroll to find nodes in MapForce versus the number of times they had to reorient the hyperbolic trees in Schema Mapper. Table 2 shows the comparison of counts required using MapForce and Schema Mapper, respectively.

Table 1: Times for Benchmark Task 1 using MapForce and Schema Mapper

User	Time using MapForce (seconds)	Time using Schema Mapper (seconds)
1	4.22	1.53
2	3.13	1.25
3	1.38	1.25
4	3.48	2.00
5	1.45	1.22

Table 2: Comparison of number of scrolls and number of reorient actions between MapForce and Schema Mapper

User	Number of scrolls using MapForce	Number of reorient actions using Schema Mapper
1	16	5
2	0	3
3	12	2
4	10	3
5	10	2

Every user scrolled many more times in MapForce than they reoriented the hyperbolic trees in Schema Mapper except for one user (User 2). However, this user was not able to complete the task and hence did not scroll at all in MapForce. This user declared the task as complete after mapping only three out of the six nodes required for Task 1 in both tools. Overall, the mapping process took more time in MapForce than in Schema Mapper. Thus, these quantitative results for Schema Mapper were positive.

Qualitative Results: All users strongly preferred the ability of Schema Mapper to allow editing from within the same tool. MapForce requires them to edit the schema using a different tool and then come back and reopen the schema again to continue mapping. Also, most users found the amount of scrolling involved in finding nodes in MapForce annoying and preferred to use the hyperbolic tree navigation technique offered by Schema Mapper.

Providing recommendations to the user proved to be very useful and strongly preferred. Users saved time looking for nodes in the global schema to map to by using the recommendations. However, the quality of the recommendations themselves and the mappings that result have yet to be evaluated.

A surprising result was that although until now hyperbolic trees have never been used for the purpose of schema mapping, users were not confused by the representation, perhaps because of the color scheme used to distinguish between the various nodes.

One negative comment provided by some users was that although the full name of a node in the hyperbolic tree was provided as a tool tip, the truncated names displayed as node labels proved to be a little confusing, which led to some mapping errors.

Some of the users suggested that we include the data type of the schema nodes as tool tip information. One of the users also suggested showing the reasoning behind the recommendation of the nodes. This would lead to the user accepting the recommendation with more confidence than otherwise. Other suggestions included: being able to re-align the hyperbolic tree in such a way that the maximum number of recommended nodes could be seen in the global schema, and the ability to move nodes within the global schema.

Overall, all of the users were very enthusiastic about Schema Mapper and preferred Schema Mapper over MapForce for simple one-to-one schema mapping purposes.

5.2 Union Service Generation
ETANA-DL supports many integrated DL services, though some are not componentized, e.g., the multi-dimensional browsing service. We are automating the generation of browsing services by developing a component for our component pool. When a browsing description specified in the scenario sub-model is fed into 5SGen (Figure 3), using this powerful component we can automatically generate a suitable browsing service.

5.2.1 Browsing Service
Open Digital Libraries (ODLs) [36] are built based on principles and philosophies derived directly from the Open Archives Initiative (OAI) [18]. Services for annotating, browsing, recommending, and searching were developed as ODL components, which could be plugged into open digital libraries. Integrated digital libraries such as ETANA-DL [32-34] and NDLTD [6] have been developed using these components.

Besides the ODL Browse component, related work that closely resembles our approach in principle is Bainbridge et al.'s metadata based classification browsing service for GreenStone [2]. They developed an extensible and dynamically configurable DL architecture which consists of a Receptionist module, and a Browse service similar in spirit to that of the browse interface module and the browsing engine in our browsing component. Similarly, Sumner et al. [38] developed a programmatic interface that uses dynamically generated components in constructing the conceptual browsing interfaces for digital libraries.

Digital objects in ETANA-DL are various archaeological data, e.g., figurine images, bone records, locus sheets, and site plans. They are organized by different hierarchical structures (e.g., animal bone records are organized based on: sites where they were excavated, temporal sequences, and animal names). By navigational dimension, we mean a hierarchical structure used to browse digital objects. Navigational dimensions of ETANA-DL can be built from taxonomies existing in botany and zoology, or from classification and description of artifacts by archaeologists.

The prior ODL browsing component doesn't provide the capability to navigate within a collection of digital objects through multiple navigation dimensions such as topic, geographical space, temporal sequence, etc. The prior browsing implementation for ETANA-DL also does not support incremental updates, i.e., whenever new digital objects are harvested into the Union Catalog it needs to re-index all records. Finally, that implementation is not extensible, i.e., it needs to be modified whenever a new navigational dimension or hierarchical structure is introduced. This is troublesome when integrating multiple DLs, since each of them may support its own browsing service and navigational dimensions, all of which need to be supported in the integrated DL. To address these issues, we designed a new automatic, extensible, and flexible browsing component.

5.2.1.1 Architecture of the Browsing Component
Figure 7 illustrates the architecture of the new browsing component. Its three main sub-components are the *browsing database maintenance module* (for creation and updating of the browsing database), the *browsing engine,* and the *browsing interface module*. These last two support browsing interaction.

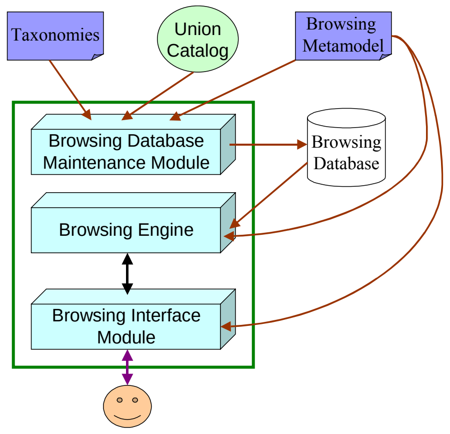


Figure 7. Architecture of the Browsing component.

All the three processing modules work based on the input provided by the browsing meta-model – an XML document that encodes the details of all navigational dimensions. The browsing meta-model of ETANA-DL can be viewed at http://feathers.dlib.vt.edu/~etana/browse/etanabrowse.xml.. In this case all hierarchical dimensions correspond to attributes of the ETANA-DL objects.

5.2.1.2 Implementation
The browsing component automates the browsing service for any integrated DL by using specific tailored meta-models, thus being able to be reused by any DL. The browsing database maintenance module is developed using Java, while the browsing engine and the browsing interface module are developed using Java Servlets. The browsing database is supported by a MySQL DBMS.

The browsing engine and the browsing interface module support the browsing interaction task. From the digital library perspective, in a browsing scenario, the end user makes two types of requests: a *navigational request* and a *display results request*. The browsing engine receives HTTP requests from the browsing interface module and responds to those requests with XML results corresponding to a *navigational response* or a *display results response*. The navigational response gives a description of the current navigation context, while the display results response gives identifiers of all digital objects that can be reached from the current navigation state, limited to the number of digital object records that can be displayed in a results page. Either the navigational request or the display results request received by the browsing engine is automatically converted into a corresponding SQL query and is queried on the browsing database. Each request is associated with a parameter that encodes the type of request being received. The rest of the parameters encode those values selected by the user for each hierarchical level while navigating through dimensions. These parameters are used to restrict the selection of next lower level values for each dimension. If it is a display results request, an additional parameter that encodes number of records per page is used.

The browsing interface module uses an XSLT style sheet to render the XML-based responses returned by the browsing engine and presents that to the user. In the case of a display results response, it provides a brief description of the DL object records based on the identifiers returned by the browsing engine. If the returned records are of different object types, appropriate style-sheets corresponding to those objects will be selected and applied while displaying the results of the browsing.

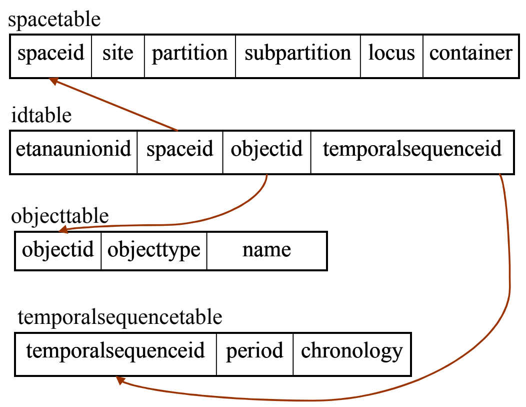


Figure 8. Browsing Database Relational Schema for the ETANA-DL Generated by the Browsing Component

The *browsing database maintenance module* creates a browsing database based on the browsing meta-model. It is responsible for populating and updating the database. The ID table in the database has a union ID that uniquely identifies a DL object as its primary key, and a foreign key referencing the primary key of each dimension from remaining attributes of the union table (Figure 8). We introduced some redundancy into the database to speed up the working of the browsing service. The populating and updating tasks are performed with the help of XPath expressions specified for each dimension present in the meta-model. Figure 8 gives the relational schema of the browsing database for ETANA-DL as generated by the browsing database maintenance module. There is a corresponding table for each of the dimensions: space, object, and temporal sequence. Finally, there is an ID table referencing the primary key of each dimension.

5.2.1.3 Testing, Experimentation, and Evaluation

To test the reusability and capability of the browsing component to support browsing services for multiple integrated DLs, we tested the component with two integrated DLs: ETANA-DL and NDLTD. Browsing services for these two DLs are available at http://feathers.dlib.vt.edu:8080/ETANA/servlet/BrowseInterface and http://feathers.dlib.vt.edu:8080/ndltd/servlet/BrowseInterface.

The tables listed below illustrate the extent of the re-usability achieved in current browsing component versions of ETANA-DL and NDLTD. We use the software metric Lines of Code (LOC) in analyzing the reusability of our browsing component. The row *Tailoring* corresponds to the number of lines of (XML) code for the metamodel. The row *Non-componentized* corresponds to the number of lines used to implement the browsing interface module, since it is currently not completely automated.

Table 1. Analysis of the reuse of the browsing component to implement browsing services for ETANA-DL and NDLTD using LOC metric

	ETANA-DL (LOC)	**NDLTD (LOC)**
Tailoring (Metamodel)	54	28
Non-Componentized (Interface)	156	159
Reused from Component	995	995
Total	1205	1182
% to reuse to implement service	82	84

With the help of the browsing component, we are able to minimize the effort associated with building a multi-dimensional browsing service. If a new dimension is introduced, or changes are made regarding hierarchical levels of a dimension, only the browsing meta-model has to edited, and the browsing database maintenance module has to be re-run. Currently, incremental updates are possible by calling an appropriate method in the browsing database maintenance module. In our future work, the database maintenance module will be extended, so it can automatically perform the task of updating whenever there is a modification in the union catalog; we are making efforts to completely automate the browsing interface module. Also, the scalability of the architecture in the presence of a large union collection will be studied. Further, we plan to develop other services, such as comparing DL objects in a domain independent manner, with the help of similar meta-models.

5.2.1.4 Quality of Browsing

We propose "knowledge-gain" as an indicator for the quality measurements of the integrated browsing service as follows.

Let *BrowseS(DL$_i$)* be a local schema of a digital library *DL$_i$*; let *BrowseS(UnionDL)* be a global schema of a union digital library *UnionDL*; let *Path(DL$_i$)* and *Path(UnionDL)* be the number of all possible navigation paths provided by *DL$_i$* and *UnionDL*, respectively. Then

$$\textbf{knowledge-gain}_{\textbf{browse}} = \frac{Path(UnionDL) - \sum_{i=1}^{n} Path(DL_i)}{\sum_{i=1}^{n} Path(DL_i)}$$

Example: Let DL_1 and DL_2 be two DLs supporting browsing of representations of excavated animal bones (Figure 9). DL_1 allows browsing bones by site organization; DL_2 allows browsing by bone names. DL_{Union}, the integrated DL built from these two DLs, supports browsing both by site organization and bone name. A navigation path denoted as "A→B→C" means a user navigates starting from A, through B, and ending with C.

The possible navigation paths for DL_1 (by site organization) are:

- Site1;
- Site1→Partition;
- Site1→Partition→Sub-partition;
- Site1→Partition→Sub-partition→Locus;
- Site1→Partition→Sub-partition→Locus→Container;
- Site1→Partition→Sub-partition→Locus→Container→Artifact;

Hence, *Path(DL$_1$) =6;*

The possible navigation paths for DL_2 (by bone name) are:

- Bone;
- Bone→BoneName;

Hence, *Path(DL_2) =2.*

The possible navigation paths for DL_{Union} are:

- All the possible navigation paths of browsing animal bones excavated from $Site_1$ and $Site_2$ by site organization;
- All the possible navigation paths of browsing by bone name;
- All the possible navigation paths of browsing by both site organization and bone name, e.g.,

  Site1→BoneName; Site1→Partition →BoneName; …

Hence, Path(DL_{Union})=(6+6+2)+4*6*2=62, therefore,

knowledge-gain$_{\text{browse}}$ = (62-(6+2))/(6+2) = 6.75

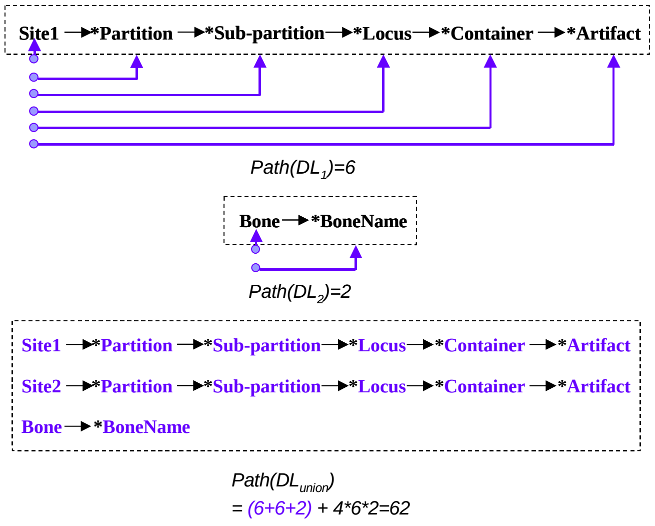

Site1 →*Partition →*Sub-partition→*Locus→*Container →*Artifact
Path(DL1)=6
Bone→*BoneName
Path(DL2)=2
Site1 →*Partition →*Sub-partition→*Locus→*Container →*Artifact
Site2 →*Partition →*Sub-partition→*Locus→*Container →*Artifact
Bone→*BoneName
Path(DLunion)
= (6+6+2) + 4*6*2=62

Figure 9. Example of calculating knowledge-gain by browsing

6. CONCLUSION AND FUTURE WORK

We formalize the DL integration problem and propose an overall approach based on the 5S framework. We then apply our framework to integrate domain-specific (archaeological) DLs, illustrating our solutions for key problems in DL integration. An integrated Archaeological DL, ETNA-DL is used as a case study to justify and evaluate our DL integration approach. To generate a union catalog and integrated browsing service in ETANA-DL, we model it based on the 5S framework, develop a visual schema mapping tool evaluated by a pilot user study, and implement a general browsing component validated through application to another integrated DL (NDTLTD).

The visual schema mapping tool and the browsing component are available at http://feathers.dlib.vt.edu/~etana/integration for testing and sharing. Future work will include improving them for more applications, and developing assessment measurements for domain specific integrated DLs, such as archaeological DLs.

7. ACKNOWLEDGMENTS

This work is funded in part by the National Science Foundation (ITR-0325579). Marcos Gonçalves was supported by an AOL fellowship and has support from CNPq. We thank Douglas Clark, Joanne Eustis, James W. Flanagan, Paul Gherman, Andrew Graham, Larry Herr, Paul Jacobs, Douglas Knight, Oystein LaBianca, David McCreery, and Randall Younker for their support. We also thank all the colleagues in the Digital Library Research Laboratory at Virginia Tech.

8. REFERENCES

[1] Adam, N.R., Atluri, V. and Adiwijaya, I. SI in digital libraries. Communications of the ACM, 43 (6): 64-72.

[2] Bowman, C.M., Danzig, P.B., Hardy, D.R., Manber, U. and Schwartz, M.F. The Harvest information discovery and access system. Computer Networks and ISDN Systems, 28 (1-2): 119 - 125.

[3] Davis, J.R. and Lagoze, C. NCSTRL: Design and deployment of a globally distributed digital library. JASIS, 51 (3): 273-280.

[4] Doan, A., Domingos, P. and Halevy, A.Y., Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In Proc. SIGMOD Conference (2001), 509-820.

[5] Flanagan, J.W., McCreery, D.W. and N.Yassine, K. Tell Nimrin: preliminary report on the 1995 excavation and geological survey, department of antiquities of Jordan, Amman, 1996.

[6] Fox, E.A., France, R.K., Gonçalves, M.A. and Suleman:, H., Building Interoperable Digital Library Services: MARIAN, Open Archives and NDLTD. In Proc. SIGIR (2001), 451-451.

[7] Fox, E.A. and Marchionini, G. Toward a Worldwide Digital Library - Introduction. Commun. ACM, 41 (4): 28-32.

[8] Gonçalves, M.A. and Fox, E.A., 5SL: a language for declarative specification and generation of digital libraries. In Proc. JCDL (2002), 263-272.

[9] Gonçalves, M.A. Stream, Structure, Space, Scenarios, and Societies (5S): A Formal Digital Library Framework and Its Applications, Ph.D. Dissertation, Dept. Comp. Sci., Virginia Tech, 2002, http://scholar.lib.vt.edu/theses/available/etd-12052004-135923

[10] Gonçalves, M.A., Fox, E.A., Watson, L.T. and Kipp, N.A. Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries. ACM Transactions on Information Systems (TOIS), 22 (2): 270 - 312.

[11] Gonçalves, M.A., Mather, P., Wang, J., Zhou, Y., Luo, M., Richardson, R., Shen, R., Xu, L. and Fox, E.A., Java MARIAN: From an OPAC to a Modern Digital Library System. In Proc. SPIRE, 2002, 194-209.

[12] Hasselbring, W. Information System Integration: Introduction Commun. ACM, 43(6): 32-38.

[13] Jacobs, P.F. Ancient World, Digital World: Excavation at Halif. ejournal Ariadne (27), 2001.

[14] Kampanya, N., Rao Shen, S.K., North, C. and Fox, E.A., Citiviz: A Visual User Interface to the CITIDEL System. In Proc. ECDL, 2004, 122-133.

[15] Kelapure, R., Gonçalves, M.A. and Fox:, E.A., Scenario-Based Generation of Digital Library Services. In Proc. ECDL 2003, 263-275.

[16] Kelapure, R.D. Scenario-Base Generation of Digital Library Service, Masters Thesis. Dept. Comp. Sci., Virginia Tech, 2003.

[17] Lagoze, C., Arms, W.Y., Gan, S., Hillmann, D., Ingram, C., Krafft, D.B., Marisa, R.J., Phipps, J., Saylor, J., Terrizzi, C., Hoehn, W., Millman, D., Allan, J., Guzman-Lara, S. and Kalt:, T., Core services in the architecture of the national science digital library (NSDL). In Proc. JCDL, 2002, 201-209.

[18] Lagoze, C. and Sompel, H.V.d., The open archives initiative: building a low-barrier interoperability framework. In Proc. JCDL, 2001, 54-62.

[19] Lamping, J., Rao, R. and Pirolli, P. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In Proc. SIGCHI, 1995, 401-408

[20] Li, W.-S. and Clifton, C. SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. Data & Knowledge Engineering, 33 (1): 49 - 84.

[21] Lynch, C.A. The Z39.50 Information Retrieval Standard D-Lib Magazine, 3(4), 1997, http://www.dlib.org/dlib/april97/04lynch.html

[22] Morgan, E.L. An Introduction to the Search/Retrieve URL Service (SRU) Ariadne Magazine, 2004, http://www.ariadne.ac.uk/issue40/morgan/

[23] Nelson, M.L., Maly, K. and Zubair, M. Interoperable Heterogeneous Digital Libraries, Old Dominion University Dept. Comp. Sci., technical report TR-98-07 1998, http://www.cs.odu.edu/~techrep/techreports/TR_98_07.ps.Z

[24] Paepcke, A., Baldonado, M., Chang, C.-C.K., Cousins, S. and Garcia-Molina, H. Building the InfoBus: A Review of Technical Choices in the Stanford Digital Library Project. IEEE Computer, 32 (2): 80-87.

[25] Paepcke, A., Chang, K.C.-C., Garcia-Molina, H. and Winograd, T. Interoperability for Digital Libraries Worldwide. Commun. ACM, 41 (4): 33-43.

[26] Paepcke, A., Chang, K.C.-C., Garcia-Molina, H. and Winograd, T. Search Middleware and the Simple Digital Library Interoperability Protocol D-Lib Magazine, 6(3), 2000, http://www.dlib.org/dlib/march00/paepcke/03paepcke.html

[27] Park, J. and Ram, S. Information systems interoperability: What lies beneath? ACM Transactions on Information Systems (TOIS), 22 (4): 595 - 632.

[28] Perugini, S., McDevitt, K., Richardson, R., Pérez-Quiñones, M.A., Shen, R., Ramakrishnan, N., Williams, C. and Fox, E.A., Enhancing usability in CITIDEL: multimodal, multilingual, and interactive visualization interfaces. In Proc. JCDL, 2004, 315-324.

[29] Rahm, E. and Bernstein, P.A. A Survey of Approaches to Automatic Schema Matching. VLDB Journal, 10 (4): 334-350.

[30] Ram, S., Park, J. and Lee, D. Digital Libraries for the Next Millennium: Challenges and Research Directions. Information Systems Frontiers, 1 (2): 75-94.

[31] Ram, S. and Ramesh, V. Information Sharing among Multiple Heterogeneous Data Sources Distributed across the Internet. In Proc. HICSS 1998, 504.

[32] Ravindranathan, U. Prototyping Digital Libraries Handling Heterogeneous Data Sources - An ETANA-DL Case Study, Masters Thesis. Dept. Comp. Sci., Virginia Tech, 2004, http://scholar.lib.vt.edu/theses/available/etd-04262004-153555

[33] Ravindranathan, U., Shen, R., Gonçalves, M.A., Weiguo Fan, E.A.F. and Flanagan, J.W. ETANA-DL: a digital library for integrated handling of heterogeneous archaeological data. In Proc. JCDL, 2004. 76-77.

[34] Ravindranathan, U., Shen, R., Gonçalves, M.A., Weiguo Fan, E.A.F. and Flanagan, J.W., Prototyping Digital Libraries Handling Heterogeneous Data Sources - The ETANA-DL Case Study. In Proc. ECDL, 2004, 186-197.

[35] Shen, R. Apply the 5S Framework in Integrating Digital Libraries. Dissertation Proposal, Virginia Tech, 2004

[36] Suleman, H. Open Digital Libraries, Ph.D. Dissertation, Dept. Comp. Sci., Virginia Tech, 2002, http://scholar.lib.vt.edu/theses/available/etd-11222002-155624

[37] Suleman, H. and Fox, E.A., Designing Protocols in Support of Digital Library Componentization. In Proc. ECDL, 2002, 568-582.

[38] Sumner, T., Bhushan, S., Ahmad, F. and Gu, Q. Designing a language for creating conceptual browsing interfaces for digital libraries. In Proc. JCDL, 2003. 258--260.

[39] West, D., Finnegan, M., Lane, R.W. and Kysar, D.A. Analysis of Faunal Remains Recovered from Tell Nimrin, Dead Sea Valley, Jordan, final report, 1996.

[40] Yuan, L.L., Miller, R.J., Haas, L.M. and Fagin, R. Data-Driven Understanding and Refinement of Schema Mappings. In Proc. SIGMOD, 2001. 485-496.

[41] Zhu, Q. 5SGraph: A Modeling Tool for Digital Libraries, Masters Thesis. Dept. Comp. Sci., Virginia Tech, 2002, http://scholar.lib.vt.edu/theses/available/etd-11272002-210531

[42] Zhu, Q., Gonçalves, M.A., Shen, R., Cassel, L. and Fox, E.A., Visual Semantic Modeling of Digital Libraries. In Proc. ECDL, 2003, 325-337.

[43] Zubair, M., Maly, K., Ameerally, I. and Nelson, M.L., Dynamic Construction of Federated Digital Libraries. In Proc. WWW9 Conference, 2000, 56-57.

Columns on Last Page Should Be Made As Close As Possible to Equal Length