# Schema Mapper: A Visualization Tool for Incremental Semi-automatic Mapping-based Integration of Heterogeneous Collections into Archaeological Digital Libraries: The ETANA-DL Case Study

by

**Ananth Raghavan**

A thesis presented to the faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

**Master of Science**
in
**Computer Science**

**Dr. Edward A. Fox - Chairman**
**Dr. Weiguo Fan**
**Dr. Chris North**

**May 4, 2005**
**Blacksburg, VA**

**Keywords: Archaeology, Digital Libraries, ETANA, Mapping, Megiddo, Schemas, Visualization**

# Schema Mapper: A Visualization Tool for Incremental Semi-automatic Mapping-based Integration of Heterogeneous Collections into Archaeological Digital Libraries: The ETANA-DL Case Study

**Ananth Raghavan**

**Committee Chairman: Dr. Edward A. Fox**

## Abstract

Schema mapping is a challenging problem. It has come to the fore in recent years; there are important applications like database schema integration and, more recently, digital library (DL) merging of heterogeneous data. With Schema Mapper we demonstrate a semi-automatic tool for schema integration that combines a novel visual interface with an algorithm-based recommendation engine.

We use ETANA-DL, a digital library developed to support integration of data from Near Eastern archaeology sites, as we explore integrating new collections. Schemas are visualized as hyperbolic trees, thus allowing more schema nodes to be displayed at one time. Matches to selections are recommended to the user, which makes the mapping operation easier and faster. Once the user has completed the mapping operation, a wrapper (XSLT Style Sheet) is created automatically with the mappings which can be applied to transform source XML files into target XML files.

Schema Mapper allows editing the target schema as part of the process of incremental enrichment of the target schema. This involves operations like adding a node as a child, renaming a node, and deleting a node. The changes to the target schema also can be logged to disk.

Schema Mapper has been applied to integrate the Megiddo Collection successfully into the ETANA-DL Union Catalog. It also has been applied for data-level mapping, to ensure consistency of data representation to the users who access the information through services provided by the DL. Formative evaluation and a preliminary comparison with MapForce suggest that Schema Mapper may be usefully employed for schema mapping.

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Fox for the support he has provided me during the entire course of my graduate studies at Virginia Tech. I have benefited a great deal from his timely guidance, and my work would never have been a success without his constant encouragement.

I am thankful to Dr. Fan for helping me get started on this project, and correcting my course as necessary. I also am thankful to Dr. North, for his valuable suggestions towards conducting my usability study and for helping me improve my thesis.

I thank Rao for her constant support and patience in answering every question that I had during the implementation of this project. Special thanks go to Srinivas for his support, guidance, assistance whenever I ran out of ideas, and for helping me review my thesis.

I thank Marcos, for his constant support, encouragement, and helping me outline my thesis. I also thank all the members of the Digital Library Research Laboratory (DLRL), for their valuable suggestions during the prototyping stages. The interactions that I have had with many of you over the last two years have been wonderful learning opportunities. I wish all of you good luck for the future.

I thank the numerous directors of excavations and their staff who not only allowed us to use their primary data but also attended meetings and answered queries. Without the contributions of these and others, my thesis work would have been more difficult and protracted.

I am thankful to my parents for constantly encouraging me to be a better person and for giving me this wonderful opportunity to study here in Tech. I also am thankful to my sister for being supportive and a constant source of inspiration. I would not be who I am today, without their blessings.

Last but not the least; I am thankful to God for everything He has given me in life.

# Table of Contents


**1. INTRODUCTION ..... 1**

1.1. PROBLEM STATEMENT ..... 1
1.2. APPROACH ..... 2
1.3. HYPOTHESIS ..... 3
1.4. CONTRIBUTIONS ..... 3
1.5. THESIS OUTLINE ..... 3

**2. BACKGROUND AND RELATED WORK ..... 5**

2.1. INTRODUCTION TO DIGITAL LIBRARIES AND CONCEPTS ..... 5
*2.1.1. What is a Digital Library? ..... 5*
*2.1.2. Examples of Digital Libraries ..... 6*
2.1.2.1. MARIAN ..... 6
2.1.2.2. NDLTD ..... 7
2.2. ETANA-DL ..... 8
*2.2.1. ETANA-DL Architecture ..... 8*
*2.2.2. ETANA-DL Services ..... 9*
2.3. SCHEMA MAPPING ..... 10
*2.3.1. Need for Schema Mapping in ETANA-DL ..... 10*
*2.3.2. Overview of Schema Mapping Tools ..... 10*
*2.3.3. Schema Visualization Approaches ..... 13*

**3. SCHEMA MAPPER ..... 16**

3.1. SYSTEM REQUIREMENTS ..... 16
*3.1.1. Core Requirements ..... 16*
*3.1.2. Schema Mapper Features ..... 17*
3.2. ARCHITECTURE ..... 20
3.3. IMPLEMENTATION ..... 23

**4. MEGIDDO COLLECTION CASE-STUDY ..... 25**

4.1. INTRODUCTION TO MEGIDDO COLLECTION ..... 25
4.2. INCREMENTAL APPROACH TO DATA INTEGRATION ..... 27
4.3. SAMPLE MAPPINGS ..... 29
*4.3.1. Flint Tool Collection ..... 29*
*4.3.2. Vessel Collection ..... 32*

**5. DATA-LEVEL MAPPING ..... 36**

5.1. WHAT IS DATA-LEVEL MAPPING ..... 36
5.2. XML FORMAT FOR DATA-LEVEL MAPPING ..... 37
5.3. SAMPLE MAPPINGS ..... 39

**6. USABILITY EVALUATION ..... 43**

6.1. CLAIMS ANALYSIS ..... 43
6.2. EXPERIMENTAL SET-UP AND USER DEMOGRAPHICS ..... 45

6.3. BENCHMARK TASKS, RESULTS AND DISCUSSIONS ..... 45
*6.3.1. BT 1* ..... *45*
*6.3.2. BT 2* ..... *49*
*6.3.3. BT 3* ..... *52*
*6.3.4. BT 4* ..... *54*
6.4. SUMMARY OF USABILITY EVALUATION ..... 55
**7. CONCLUSIONS AND FUTURE WORK ..... 56**
7.1. CONCLUSIONS ..... 56
7.2. FUTURE WORK ..... 57
**REFERENCES ..... 59**
**APPENDIX A ..... 62**
**VITA ..... 66**

# List of Figures

Figure 1: MARIAN-DL Architecture (adapted from [10]) ..... 7
Figure 2: ODL-based NDLTD Architecture (adapted from [14]) ..... 7
Figure 3: ETANA-DL Architecture (adapted from [2]) ..... 8
Figure 4: Web Interface for ETANA-DL ..... 9
Figure 5: SpotFire (adapted from [23]) ..... 11
Figure 6: Files and Folders Scenario Visualized using Snap (adapted from [23]) ..... 11
Figure 7: XML-XML Schema Mapping using MapForce ..... 12
Figure 8: Windows Explorer View ..... 13
Figure 9: TreeMaps ..... 14
Figure 10: Hyperbolic Tree Representation ..... 15
Figure 11: Screenshot of Schema Mapper Indicating the Different Colors ..... 18
Figure 12: Architecture of Schema Mapper ..... 20
Figure 13: Process of Integrating Local Archaeological Data into ETANA-DL ..... 25
Figure 14: Megiddo Site Organization ..... 26
Figure 15: Screenshot of Schema Mapper Representing the Megiddo Collection Format (left hand side screen) and the ETANA-DL Global Format (right hand screen) ..... 27
Figure 16: Initial Set of Mappings Suggested By Recommendations Based on Rules ..... 29
Figure 17: Adding Flint Tool as a Child of OBJECT in the Global Schema ..... 30
Figure 18: Description Node in the Global Schema Renamed to DESCRIPTION and the User Choosing to Save Mappings ..... 31
Figure 19: Flint Tool Style Sheet Generated by Schema Mapper ..... 32
Figure 20: View Only Top-level Leaf Nodes Option ..... 33
Figure 21: Name Change Recommendation by Schema Mapper ..... 34
Figure 22: ETANA-DL Global Schema after the Integration of the Megiddo Collection. ..... 35
Figure 23: Sample XML File Conforming to Format Required by Schema Mapper ..... 38
Figure 24: Schema Mapper Representing Local XML File for Data Mapping ..... 38
Figure 25: Local and Global XML Files Containing the Data for CulturalPhrases Element. ..... 40
Figure 26: PERSIAN Recommended (in red color) for Selection PER because of Mappings History ..... 41
Figure 27: BYZANTINE Recommended for Selection BYZANTIN because of Edit Distance Algorithm ..... 41
Figure 28: Stone Added as a Child and Renamed as STONE in the Global XML file ..... 42
Figure 29: Output Tab Delimited Text File Containing the Data-Level Mappings ..... 42
Figure 30: Ratio (In Percentage) of Time Taken using Schema Mapper to that using MapForce for BT 1 ..... 47
Figure 31: Ratio (In Percentage) of Re-orient Actions using Schema Mapper to Scrolls using MapForce ..... 47
Figure 32: Ratio (In Percentage) of Time Taken using Schema Mapper to Time Taken using MapForce and XML Spy Combined for BT 2 ..... 51
Figure 33: Ratio (In Percentage) of Time Taken to Identify Mappings using Schema Mapper to that using MapForce ..... 53

# List of Tables

Table 1: Mappings History Table for Tag Element CulturalPhrases..... 39
Table 2: Claims, Trade-offs, and Method to Explore These Trade-offs..... 45
Table 3: BT 1 Results ..... 46
Table 4: BT 2 Results ..... 50
Table 5: BT 3 Results ..... 53

# 1. INTRODUCTION

## 1.1. Problem Statement

During the past several decades, archaeology as a discipline and practice has increasingly embraced digital technologies and electronic resources. Vast quantities of information produced by archaeological research are being generated, stored, and processed using customized monolithic information systems. But migration or export of archaeological data from one system to another is a monumental task that is aggravated by peculiar data formats and database schemas. Hence achieving interoperability between these systems becomes extremely difficult.

This issue could be solved by having an Archaeological Digital Library (DL): a single unifying system that manages heterogeneous information from several sites, while providing services that are tailored for the archaeological domain. One of the key issues identified here is managing the heterogeneous information. Different archaeological sites excavate different kinds of objects and each object has certain peculiar characteristics associated with it. This, coupled with the fact that different branches of archaeology have special methods of classification, contributes not only to the heterogeneity of the content but also to the heterogeneity of the input formats that need to be handled.

Merging these different formats into a global format can be handled by using the intermediary-based approach, which employs wrappers, ontologies, mediators, and agents. Yet, while many research projects developed semantic mediators and wrappers to address the interoperability issue, few tackled the problem of (partial) automatic production of these mediators and wrappers (through a mapping-based approach).

The mapping-based approach attempts to construct mappings between semantically related information sources. It is usually accomplished by constructing a global schema representing the global format, plus local schemas for local formats, and by establishing mappings between the local and global schemas. Existing tools for schema mapping approach the process either from a visualization or from an algorithmic perspective with few looking to integrate both techniques. The problem thus boils down to generating a wrapper by enabling mappings between local and global schemas and providing visualization to the mapping process that allows for a better mapping experience to the user, both in terms of usability and efficiency of the mapping activity.

## 1.2. Approach

Our approach to the problem presented above is to build a tool, "Schema Mapper" [1], that allows us to integrate heterogeneous formats of data into one global format and then harvest the data into the digital library (DL). We use the data to be integrated into ETANA-DL [2], an Archaeological DL, as input to the tool. The main output from the tool is an XSLT [3] style sheet (wrapper) containing mappings from the local to the global schemas, which can be applied to the data conforming to the local formats and can transform them into the global format.

Schema Mapper partially automates the wrapper production process. This is because archaeological data classification depends on a number of vaguely defined qualitative characteristics that are open to personal interpretation. Hence mapping from the local format to the global format requires the expertise of a human who is familiar with the archaeology domain. The automation comes from the fact that the user does not have to write any code for producing the wrapper. This widens the target audience to include non-computer scientists. The target audiences for Schema Mapper, who constitute the "societies" in the 5S framework [4], thus are archaeologists or digital librarians familiar with the archaeology domain, who would want to integrate new collections into the global collection, through data harvesting after conversion from the local format.

Requirements elicitation is an important part in the development of any software. Accordingly, a rapid prototyping approach was adopted for both requirements gathering as well as testing the feasibility of our approach. We describe the design and implementation of our tool as well as the research issues that we explored during the development process. We also describe certain applications (apart from wrapper generation), like schema editing and data mapping, that we discovered Schema Mapper could be used for, as a result of its flexible design.

By integrating the Megiddo collection [5], we show that our approach successfully integrates heterogeneous data into the ETANA-DL Union Catalog. The effectiveness of Schema Mapper will be realized, as and when more collections need to be integrated into the DL. As an essential part of Schema Mapper is the Graphical User Interface (GUI), we also conducted a usability evaluation, to justify our claims about the usability of the tool as well as to perform a comparative evaluation with traditional visual mapping tools. This comparison also deals with the efficiency of the tools in dealing with an actual scenario for data integration.

## 1.3. Hypothesis

We hypothesize that:

a) The problem (described in Section 1.1) of integrating heterogeneous data into a Union Catalog of a DL

b) can be partially automated through our approach (described in Section 1.2) of building a flexible GUI-based mapping tool which, through human interaction during the mapping process, generates a wrapper that then could be applied to the data to achieve the necessary transformation and be ready for harvesting into the Union Catalog.

## 1.4. Contributions

The main contributions of this work are:

- A scalable, flexible, and manageable approach to integrating heterogeneous data from different sources into a Union Catalog.
- An approach that combines visual representation and an algorithm based recommendation engine that allows for better mapping experience.
- An approach that allows active participation of non-computer scientists in the mapping process.
- An easy approach to schema editing during the mapping process.
- An approach that allows for data-level mapping to be achieved.

## 1.5. Thesis Outline

Chapter 2 presents background on digital libraries, archaeological information systems, the ETANA-DL, schema mapping tools, and various schema visualization approaches.

Chapter 3 presents requirements, architecture, and the design of Schema Mapper. It describes the algorithms used by the recommendation engine. It also discusses several key implementation issues.

Chapter 4 describes the actual process of mapping with the Megiddo Collection Case Study. It describes the incremental approach to mapping, as well as gives examples of the mapping process involved in integrating two different types of artifacts, belonging to the Megiddo Collection, into the Union Catalog format, using Schema Mapper.

Chapter 5 describes the application of Schema Mapper to achieve data-level mapping. It describes the motivation behind data mapping, and presents examples of two different data files that need to be mapped to corresponding global files.

Chapter 6 describes the usability evaluation of Schema Mapper.

Chapter 7 details the conclusions and possible extensions of Schema Mapper.

# 2. BACKGROUND AND RELATED WORK

In this chapter, we put together the background information related to this work. Readers are introduced to various concepts related to digital libraries (DLs) and examples of DLs. We proceed to describe the ETANA-DL and the services that it provides, to give a better understanding of the overall context in which Schema Mapper is required. We also give a brief overview of existing schema mapping tools and why these would not serve our specific purpose. We also detail the different schema visualization approaches that were explored, and the pros and cons of each of these approaches.

## 2.1. Introduction to Digital Libraries and Concepts

### 2.1.1. What is a Digital Library?

Digital Libraries are organized collections of digital information. These combine the structuring and archiving capabilities of traditional libraries with the digital capabilities provided by computers. The concept of a Digital Library first came to light in an article published in the Atlantic Monthly in July 1945 [6], in which Vannevar Bush, describes "***a device in which an individual stores all his books, records and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility…***". This device, which he called Memex, has since given rise to much research in the area of digital libraries.

As the volume of research in this area increased, the meaning attributed to the term Digital Libraries became varied. In order to achieve a common understanding of what constitute digital libraries, the partner institutions in the Digital Library Federation (DLF) [7] came up with the following definition for Digital Libraries.

***Digital libraries are organizations that provide the resources, including the specialized staff, to select, structure, offer intellectual access to, interpret, distribute, preserve the integrity of, and ensure the persistence over time of collections of digital works so that they are readily and economically available for use by a defined community or set of communities.***

A more recent definition of Digital Libraries based on the 5S framework states that

***Digital libraries are complex data/information/knowledge systems that help: satisfy the information needs of users (societies), provide information services (scenarios), organize information in usable ways (structures), manage the location of information (spaces), and communicate information with users and their agents (streams) [8].***

In spite of the fact that there are different definitions for DLs, they share certain common characteristics. A summary of characteristics as provided by the Association of Research Libraries [9] is as follows:

- The digital library is not a single entity;
- The digital library requires technology to link the resources of many;
- The linkages between the many digital libraries and information services are transparent to the end users;
- Universal access to digital libraries and information services is a goal;
- Digital library collections are not limited to document surrogates: they extend to digital artifacts that cannot be represented or distributed in printed formats.

A summary of the advantages of a Digital Library is as follows:

- Dissemination of information: Information in a digital library is available to more people and the access to information is also faster than in traditional libraries. Physical location is not an issue as in traditional libraries and people can access this information from basically anywhere: homes, offices, schools, etc.
- Availability of services: DLs provide different kinds of services like Searching, Browsing collections, Comparison of digital objects, Recommendations based on personalization techniques, etc.
- Preservation: Archiving to ensure future use is another important aspect of digital libraries. Redundant storage of digitized information safeguards it from loss, damage, or catastrophe.

## 2.1.2. Examples of Digital Libraries

### 2.1.2.1. MARIAN

Multiple Access and Retrieval of Information with AnnotatioNs (MARIAN) [10] is an object-oriented digital library system with a modular architecture, flexible data model, and a powerful search mechanism [11]. MARIAN is a monolithic DL with a layered architecture [12]. The following figure highlights its architecture.

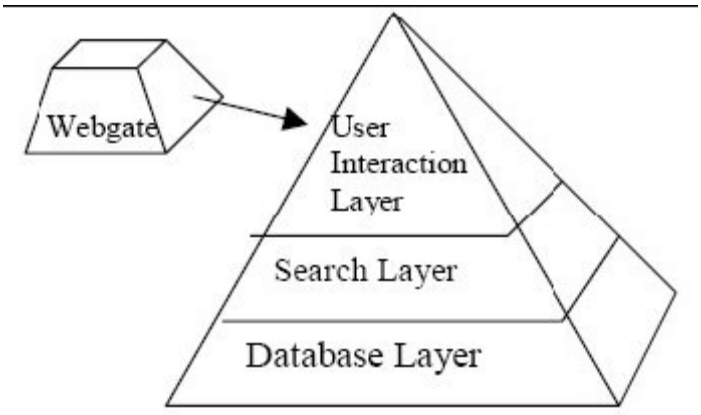


**Figure 1: MARIAN-DL Architecture (adapted from [10])**

### 2.1.2.2. NDLTD

The Networked Digital Library of Theses and Dissertations (NDLTD) [13] is an initiative to promote the adoption, creation, use, dissemination, and preservation of electronic analogues to traditional paper-based theses and dissertations. Individual universities or consortial agents maintain the collections. Each university that wants to share its theses and dissertations implements data providers to expose the related metadata in standard formats (e.g., Dublin Core). The metadata is then harvested into the Union Catalog of the DL and services like searching, browsing, etc. are provided to the users of the DL. The following figure shows an ODL-based [14] NDLTD architecture.

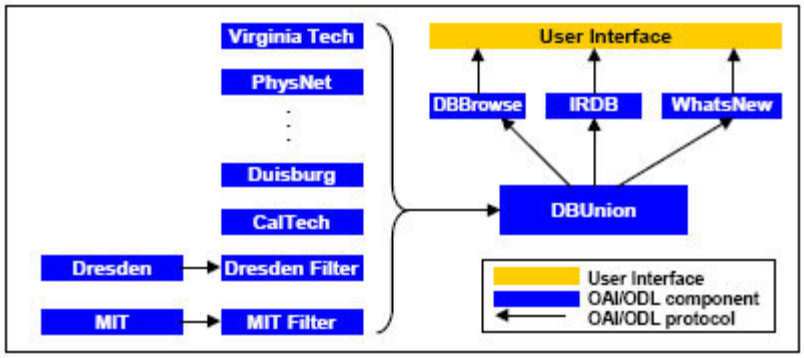


**Figure 2: ODL-based NDLTD Architecture (adapted from [14])**

## 2.2. ETANA-DL

ETANA-DL [15] is a model-based, extensible, componentized DL that manages complex information sources using the client-server paradigm of the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [16]. The DL is built using two archaeological components: DigBase (DB) – a repository and an archive for archaeological data, and DigKit (DK) – a compatible field tool for collecting and recording archaeological data during archaeological surveys and excavations. ETANA-DL is used to enable archaeologists, humanists, and social scientists to gather, preserve, and disseminate real-time and historical archaeological data. It combines data from several excavation projects like Nimrin [17], Lahav [18], Umayri [19], etc., about different artifacts like Bones, Seeds, Figurines, etc., which are represented in many forms, and merges them into a global repository. Based on this information, it provides several services like information access, traditional DL services like annotations, and archaeology specific services like comparing different objects.

### 2.2.1. ETANA-DL Architecture

The architecture of ETANA-DL is as shown in the diagram below [2]:

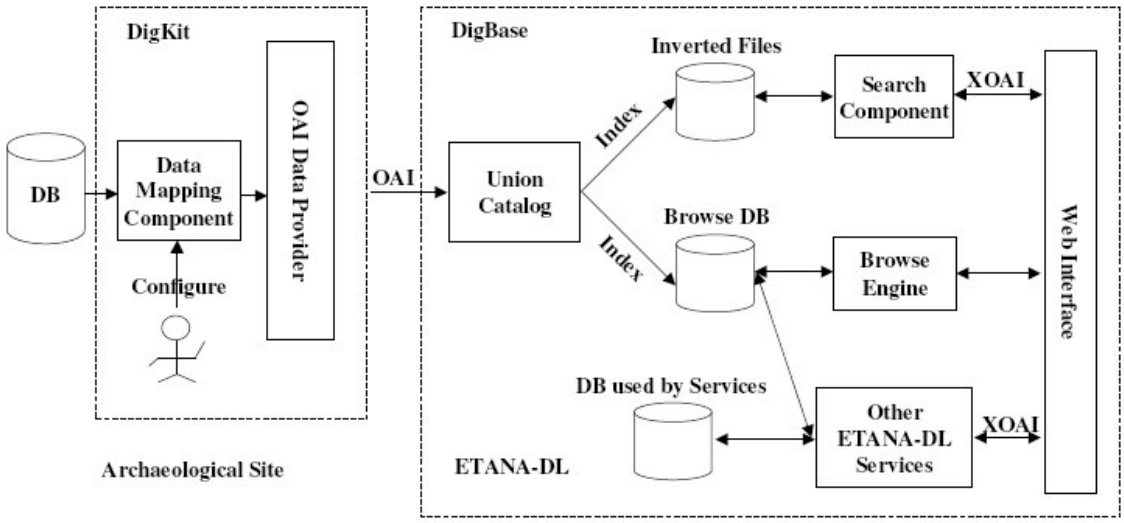


**Figure 3: ETANA-DL Architecture (adapted from [2])**

Each site that acts as a Data Provider needs to expose its data in a standard format. For this to happen there exists a data mapping component that converts the custom local format of the data into the global format. The data (in global format) from all sites is harvested into the Union Catalog at the Service Provider by issuing OAI-PMH requests to the archives provided by the Data Provider. Various services are provided on the local copy of the harvested data. A web interface is then provided for allowing people to utilize the services and access information from this DL.

### 2.2.2. ETANA-DL Services

The Services provided by ETANA-DL can be mainly categorized into 4 parts [2]:

1. Information Satisfaction Services: Searching for specific information, browsing through multiple dimensions and categories, and recommendations for digital objects according to similar interests with other users.
2. Domain-specific services (archaeology domain): Digital object comparison based on parameter specification by the user, marking items as objects of interests to other users.
3. Value-added services: Annotations leading to discussion about various digital objects, allowing users to view their most recent searches and allowing users to create personal collections out of items that interest them.
4. Miscellaneous Services: User Management and Collection Description.

The web interface [20] that allows users to access the services is as follows:

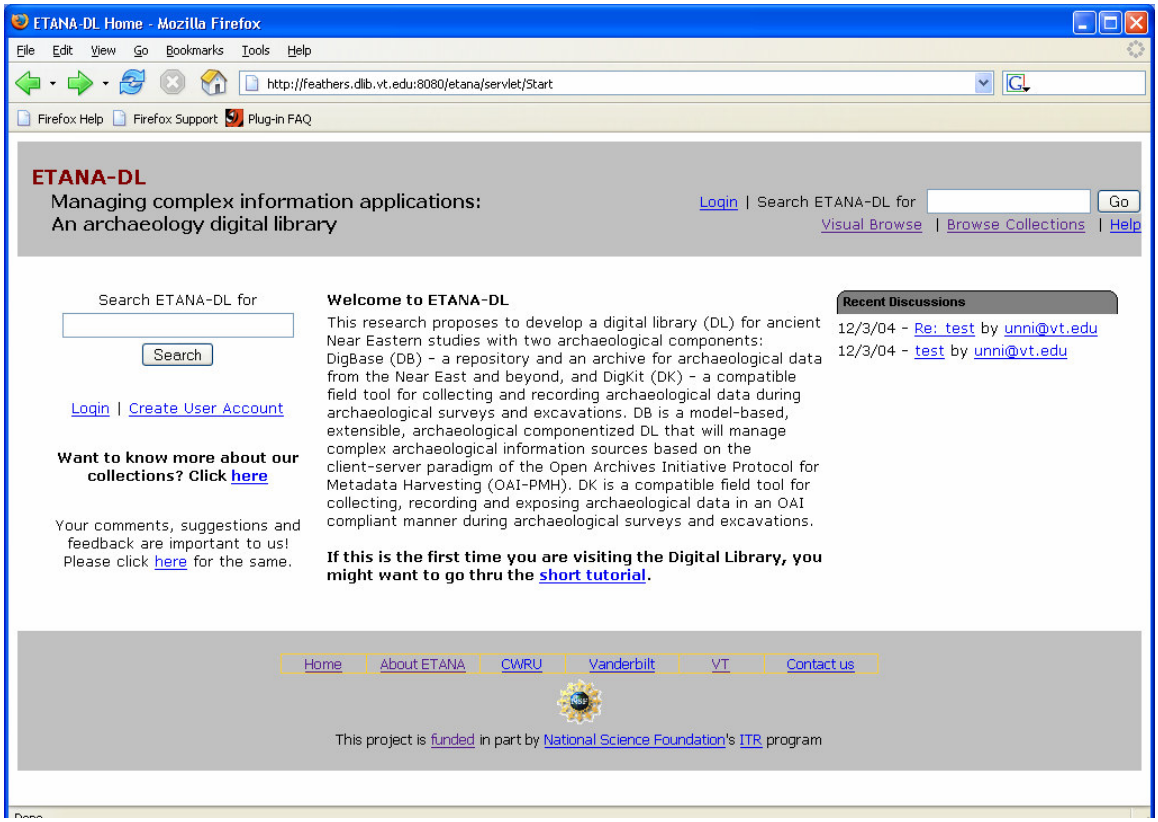


**Figure 4: Web Interface for ETANA-DL**

The present prototype for ETANA-DL supports visual browsing of sites; browsing through collections with multiple dimensions like browsing by site, period, or artifact

type; and searching for a specific object. It also allows user management functions like login.

## 2.3. Schema Mapping

Schema mapping is a challenging problem that has come to the forefront in recent years because it has application in several important domains. While there are older application domains such as database schema integration, there are several newer domains like DLs which involve merging of different kinds of data.

### 2.3.1. Need for Schema Mapping in ETANA-DL

As is discussed in Section 2.2.1 (see Figure 3), there is a need for a data mapping component, that takes in the data in the local format and needs to convert it into the standard ETANA-DL format. Previously this conversion required new and specialized code. Separate proprietary wrappers were written for each new archaeological collection that needed to be integrated. This was a problem not just because proprietary wrapper coding was grossly inefficient but also because these wrappers were not reusable for new collections.

A better alternative to this process would be to have one mapping component that could be used for mapping any local format into the global ETANA-DL format. This would be much more efficient and the resulting mapping component would be re-usable. As this involves mapping from one format to another, the mapping component would be at the schema level rather than at the data level.

### 2.3.2. Overview of Schema Mapping Tools

Schema mapping is an interesting problem which has so far been addressed from either an algorithmic point of view or from a visualization point of view for which commercial tools are available. For trivial database schemas, tools like SpotFire [21] (see Figure 5) can be applied. However, for non-trivial and complex database schemas, usually custom-built solutions have to be developed for each database [22].

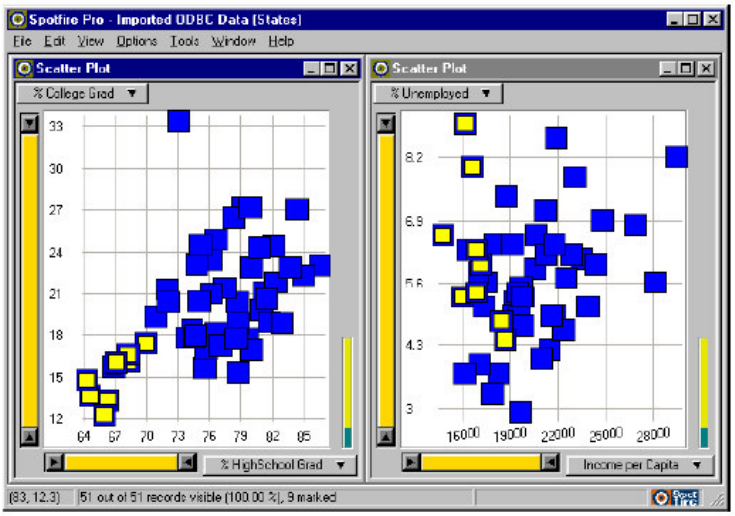


**Figure 5: SpotFire (adapted from [23])**

Snap-Together Visualization [22] enables database owners to rapidly construct multiple-view visualizations for databases without programming. The Snap visualization model extends principles of the relational data model into the realm of visualization. Figure 6 [23] shows an example of the visualization model for a Files and Folders scenario involving multiple views using SpotFire, Hyperbolic trees, data tables, and papers**.**

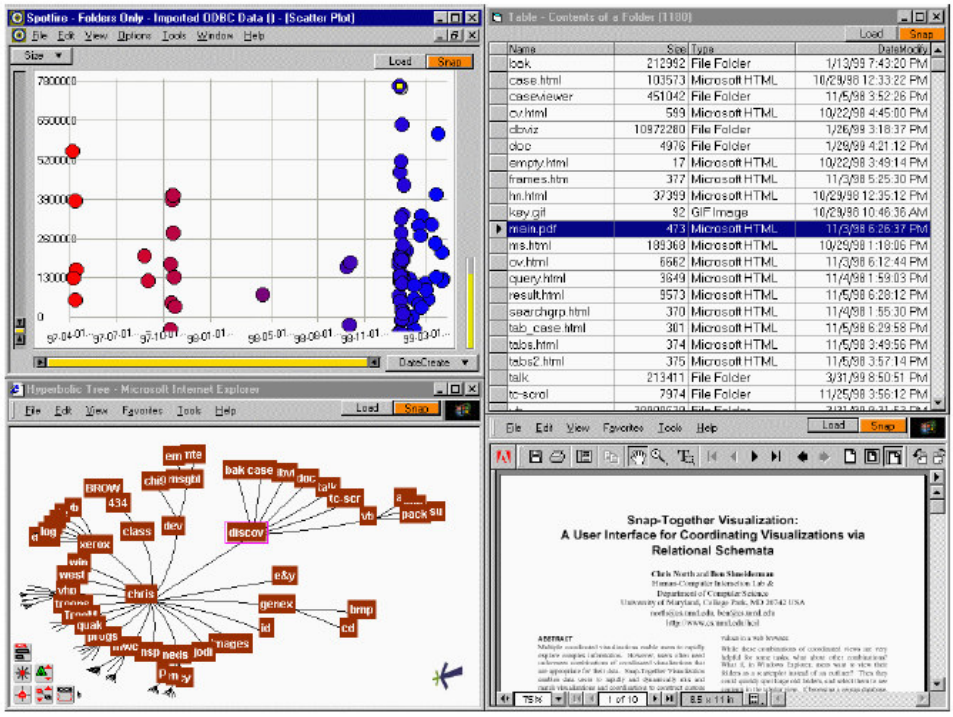


**Figure 6: Files and Folders Scenario Visualized using Snap (adapted from [23])**

OPOSSUM (obtaining presentations of semantic schemas using metaphors) [24] allows for visualizing and editing object oriented database schemas. The notion of a visual metaphor is defined as a mapping from (the elements of) a visual model to (the elements of) a data model. Through this mapping, one is able to assign meaning to visualization (visual schema) in terms of the underlying data schema.

Clio [25] is a schema mapping tool which considers and manages alternative mappings between heterogeneous data sources and a target or integrated schema. Clio uses reasoning about queries to suggest the mappings. However the final choice of mappings is made by the user who understands the semantics of the target schema and the mapping. This interactive mapping creation paradigm is based on value correspondences, which the user needs to specify to show how a value of a target attribute can be created from a set of values of source attributes [26]. Clio is based on the data driven schema mapping paradigm as data examples are the basis of understanding and refining the schema mappings.

There also exist commercial tools like MapForce [27] and BizTalk Mapper [28] that are used for schema mapping purposes. These are for XML-XML mapping, Database-XML mapping, and also for Database-Database mapping. MapForce is a product of Altova while BizTalk Mapper is part of Microsoft Corporation's BizTalk Server and is built to work on the .NET platform and hence is bulkier, when compared with MapForce [29]. Figure 7 represents a mapping between two XML schemas using MapForce.

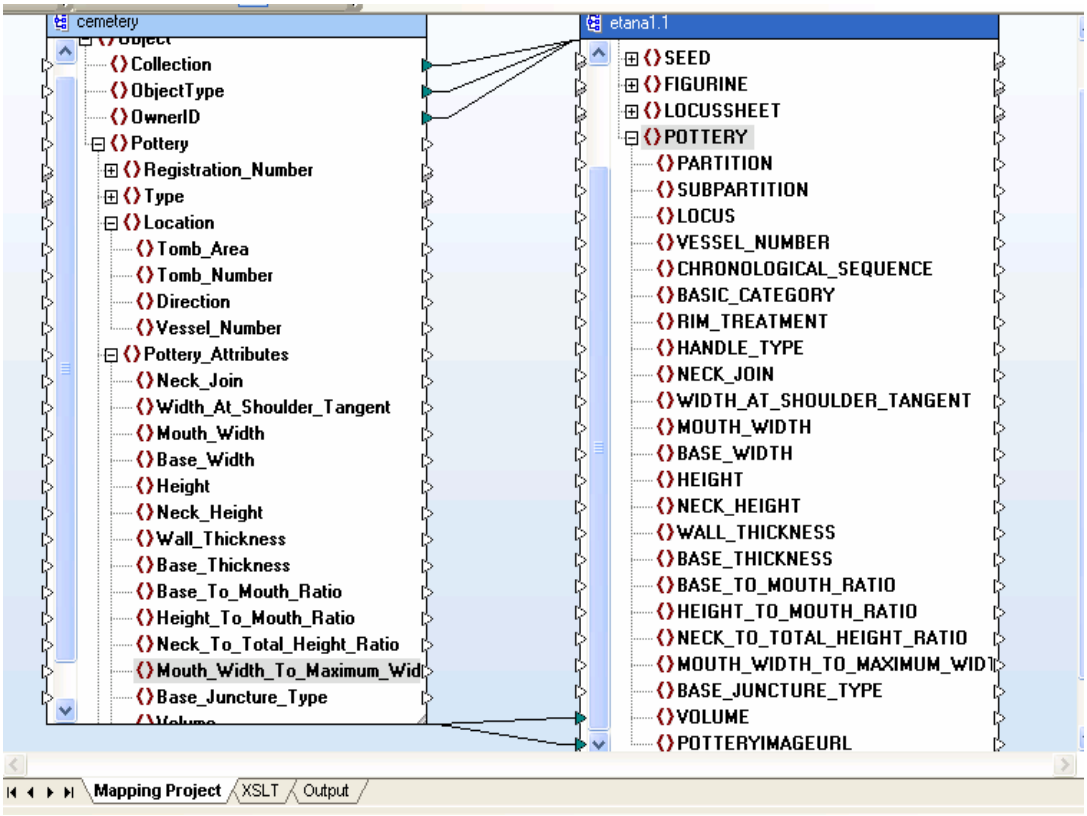


**Figure 7: XML-XML Schema Mapping using MapForce**

### 2.3.3. Schema Visualization Approaches

To quote Card, Mackinlay, and Schneiderman: Information Visualization can be defined as "***The use of computer-supported, interactive, visual representations of abstract data to amplify cognition***" [30]. Visual interfaces to hierarchically structured information apply powerful data analysis and information visualization techniques to gather insight about the information. There exist different visual representations for trees and similar hierarchical information structures.

Traditional representations include the Windows Explorer view (see Figure 8) of the directory structure of a computer wherein typically there is not much of an overview provided, and which requires us to frequently scroll and lose the context with respect to our current focus. However representations like TreeMaps [31], Cone Trees [32], Bubble Trees [33], and Hyperbolic Tree [34, 35] enable us to get more of an overview and hence more of an insight into the data. These alternate techniques are based on the fundamental mantra of Information Visualization of "Overview first, zoom and filter, details on demand" [36].

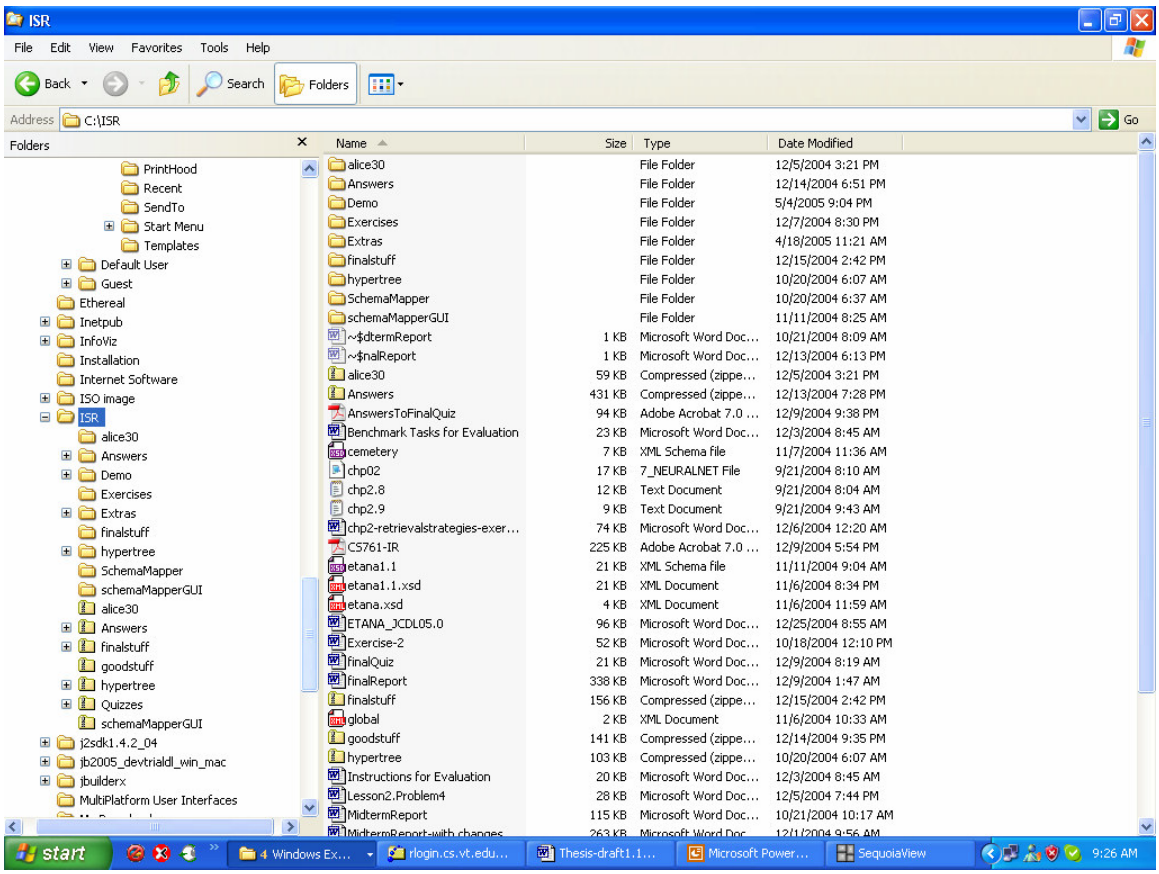


**Figure 8: Windows Explorer View**

TreeMaps (see Figure 9) allow for efficient representation of large trees by representing each node in the tree as a rectangle which is proportional to its size. Further children of this node will all be contained within this rectangle as smaller rectangles proportional to their size.

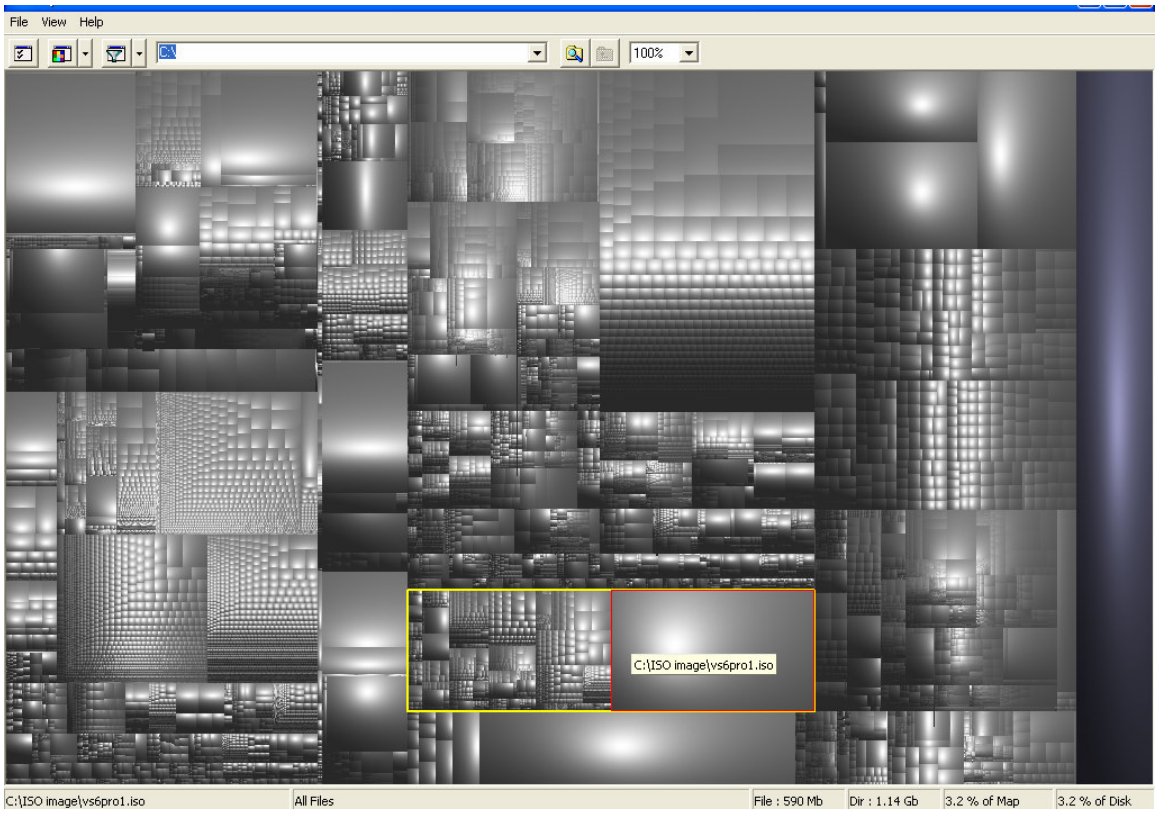


**Figure 9: TreeMaps**

Cone Trees are interactive 3-D visualization techniques aimed to maximize the effective use of screen space and to exploit the perceptual abilities of human beings. However, evaluation studies have indicated that locating data is slower in cone trees as compared to traditional representations [37]. The bubble tree mechanism is proposed, based on the natural property of trees to recursively categorize themselves into sub-trees [33]. Each sub-tree is represented as a bubble enclosing its children and descendants. Bursting the bubble is analogous to focusing on a particular node of the tree.

Hyperbolic trees (see Figure 10) assign more display space to a portion of the hierarchy while still embedding it in context by laying out the hierarchy in a uniform way on a hyperbolic plane and mapping this plane onto a circular display region. These trees represent a focus + context technique called “fish eye”, for visualizing and manipulating large hierarchies [35]. In terms of navigation, studies have shown that hyperbolic trees help people to locate information 62.5% faster as compared to standard navigation methods [38]. It has been shown that such techniques are better by an order of magnitude as compared to the traditional tree representation techniques in terms of number of nodes of the tree that can be displayed at a time.

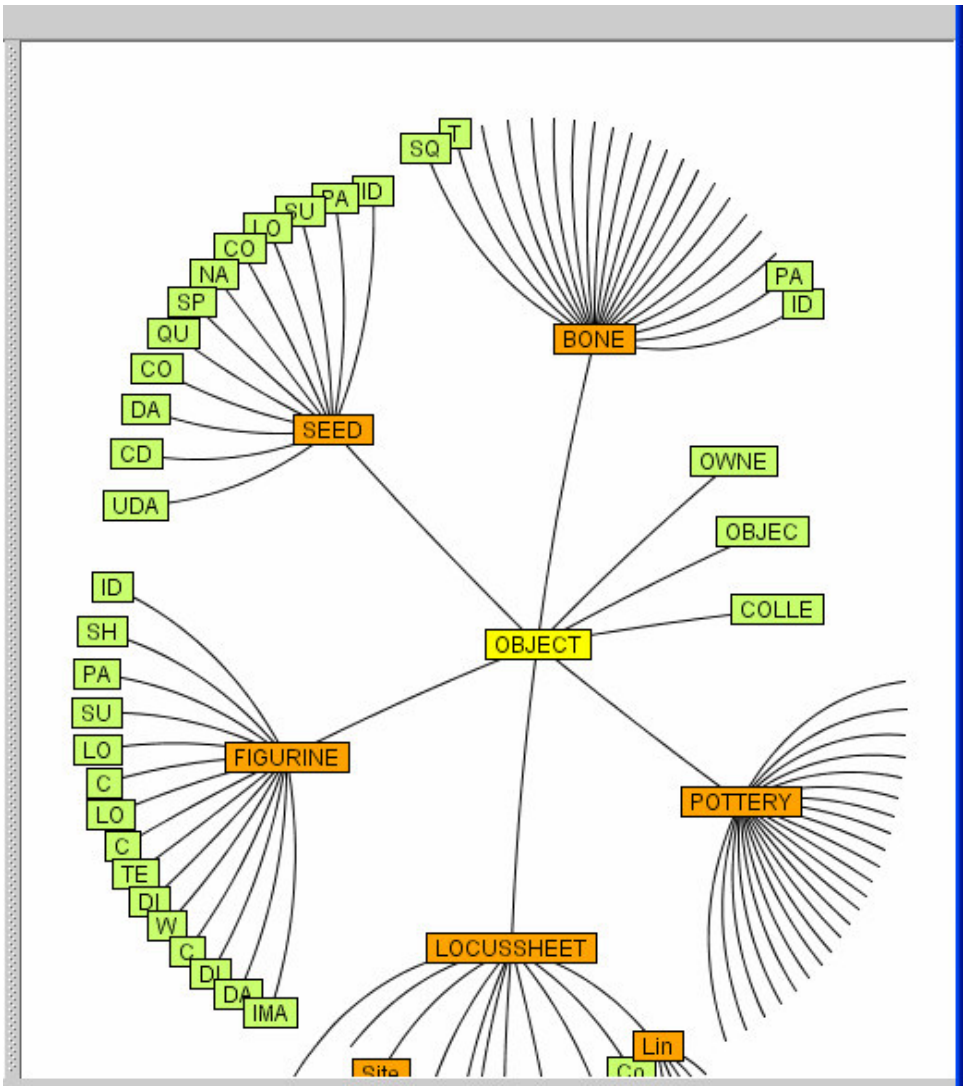


**Figure 10: Hyperbolic Tree Representation**

# 3. SCHEMA MAPPER

In this chapter, we discuss the Schema Mapper system. We start off by describing the requirements for Schema Mapper and also briefly describe the rapid prototyping methodology adopted for requirements elicitation. We proceed to describe the architecture of Schema Mapper and explain the chief components involved. We then describe the implementation details for each of the components involved in the architecture.

## 3.1. System Requirements

### 3.1.1. Core Requirements

Section 2.3.1 described the need for schema mapping in the ETANA-DL and Section 2.3.2 described the existing mapping tools which could achieve schema mapping. However, none of these mapping tools are targeted towards the archaeology domain wherein prediction of the global schema format is impossible simply because of the varied nature of the information representing the artifacts excavated from various sites. For example, an artifact of excavation, "bone", will have as one of its attributes "animal name" (whose bone it is), whereas this attribute would not be present for describing a "pottery" artifact. Similarly, a "pottery" artifact would have an attribute called "Volume" which wouldn't be present to describe a "bone" object. Thus the process of integration of a new artifact, hitherto not present in the ETANA-DL Union Catalog, would mean editing the global schema to include this new artifact type. Hence the new tool needs to have a schema editing capability.

Visualization allows users to gain insight into the data with which they are dealing. This is essential to the speed and accuracy desired for mapping. Hence visualization needs to be provided for the schemas and for the mapping process. Commercial tools like MapForce, which provide visualizing capability, use a linear representation for schemas, which has an inherent limitation in the number of nodes that can be displayed on the screen at a time (about 30 nodes). This meant that excessive scrolling was involved even for moderately sized schemas (50-60 nodes). This also meant that the tool had to adopt a visualization technique which alleviated the problem of scrolling by representing a larger number of nodes on screen.

For the process of mapping, users need to draw lines from the local to the global schema, in tools like MapForce. This results in a large number of lines going across the screen (see Figure 7). This method for representing mappings doesn't scale well and results in confusing the users when they need to identify a particular mapping. Hence the mapping tool required an alternative way of representing mappings.

Thus the basic requirements for the mapping tool can be summarized as follows:

- Local and global schemas need to be visualized on the same screen using a technique that avoids the problem of scrolling while mapping.
- Mappings need to be represented in a manner that avoids lines going across the screen from the local to global schemas.
- Global schema editing capabilities required for global schema enrichment during mapping.

Section 2.3.3 describes alternate schema visualization strategies. Two of these were TreeMaps and Hyperbolic Trees. TreeMaps allow for representing almost 2 orders of magnitude more nodes on the screen than linear representations. However locating a particular node would take time. Also, TreeMaps are good for presenting an overview but are not very good for manipulating data. Hyperbolic trees on the other hand allow for representing an order of magnitude more nodes on screen than their linear counterparts, and locating a node in a hyperbolic tree is faster than in TreeMaps. Hence, we decided to have a hyperbolic tree representation for the local and global schemas. Also, instead of showing lines across the screen for mappings, we decided to display a mapping table that contained the mappings, supplemented by helpful highlighting, e.g., changing new matched pairs to have the same color.

### 3.1.2. Schema Mapper Features

Research has shown that rapid prototyping is an excellent tool for deciding the look and feel of a GUI during software development. Accordingly we went through several iterations (see Appendix A for discussion of the design decisions for each iteration) till the final look and feel of the interface became well-defined. Several key decisions were made regarding the features of Schema Mapper, thanks to this approach. They are detailed as follows:

**Concept of Recommendations**: A significant problem with hyperbolic tree representations is that users tend to lose context within the tree. Hence searching for a particular node at the GUI level becomes extremely difficult as there is no structured traversal that the user can perform for arriving at a particular node. This can be avoided to some extent by "recommending" nodes to the user. Thus, if the user selects a node in the local schema for mapping, Schema Mapper should recommend a node in the global schema and should highlight this node for the user to map to.

**Colors to differentiate between nodes**: In order to provide a sense of context to the user within the hyperbolic tree, we decided to assign colors to differentiate between nodes. Accordingly we assigned different colors to differentiate between root, leaf, and non-leaf nodes. The concept of color was later extended to also differentiate between selected / recommended nodes and mapped nodes. Figure 11 shows a screenshot of Schema Mapper with the local and global schemas being the left and right hyperbolic trees respectively. The coloring key is provided at the right hand bottom corner of the screen.

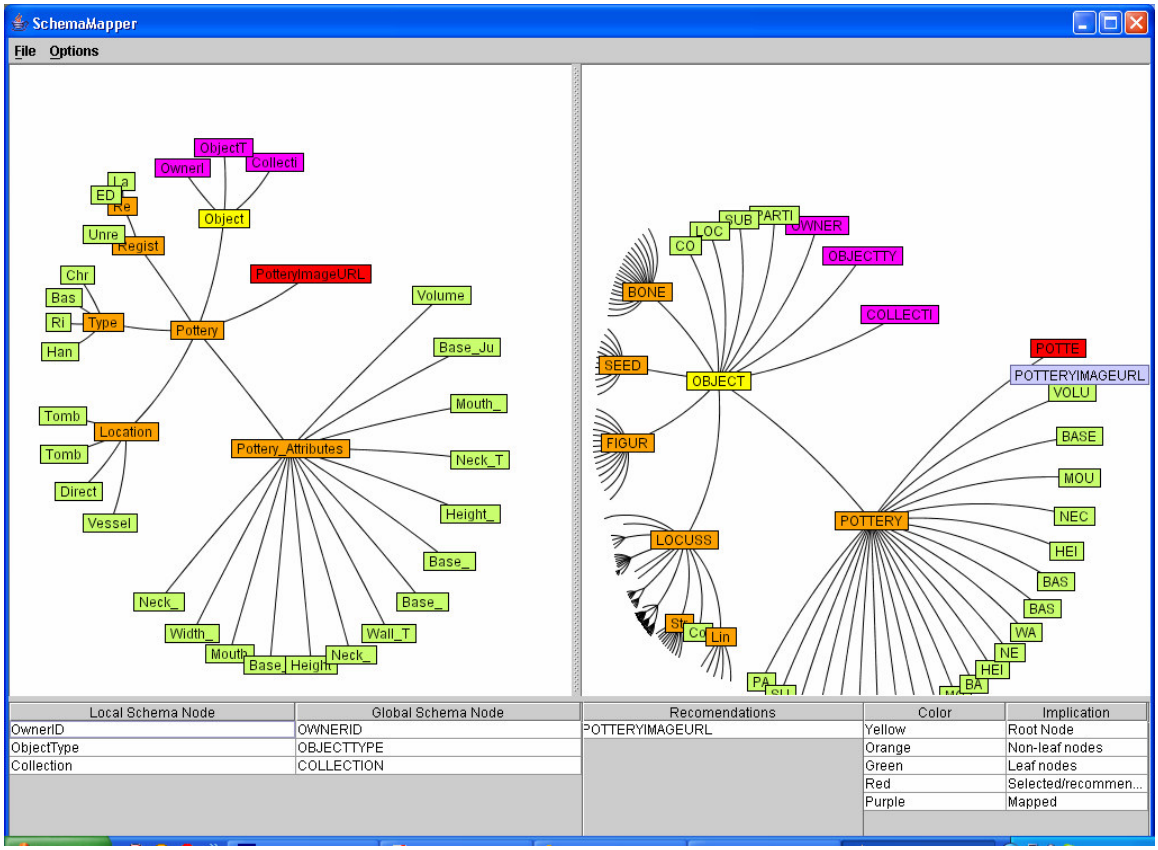


**Figure 11: Screenshot of Schema Mapper Indicating the Different Colors**

**Displaying Mapping and Recommendation Tables:** In order to identify mappings, the mapping table is displayed at the left hand bottom of the screen. The screen also displays a recommendation table (middle bottom). For a selected node in the local schema, the recommendation table shows the list of names of recommended nodes apart from highlighting the recommended global node on screen in red. Each node's name is also available as tool-tip information. This is because the full name of a node may or may not be visible depending on the position of that node on screen.

**Schema Editing:** Schema editing features are provided as context menu (right click) options on each node in the global schema. There are two options provided here: renaming a node and deleting a node. Deleting a parent node also results in deletion of all its children. Schema Mapper also allows adding a local sub-tree as a child of a non-leaf node in the global sub-tree. Thus integration of new collections is facilitated.

**Viewing top-level leaf nodes and viewing only this sub-tree:** As the number of collections keeps on increasing, the global format will keep increasing in size. However, if we are concentrating on mapping a particular sub-branch in a collection, it would be nice if we could choose to view only that sub-branch and make all other sub-branches disappear. The "View Only this Sub-tree" option achieves exactly this. Similarly, if a new collection needs to be added as a child to the global schema root node and then

needs to be mapped, the user may want to avoid viewing other children of the root node except the leaf nodes. Hence an option of "Viewing Top-level Leaf Nodes" is provided.

**Save Edited Global Schema:** If the user edits the global schema, he may choose to save the changes. The "Save Edited Global Schema" option writes the new edited global schema to disk.

**Save Mappings:** Once the user has completed mapping a local schema to a global schema, he may use this option to save the mappings, and generate an XSLT [3] style sheet. This style sheet is the wrapper generated from the mappings. When applied to files conforming to the local schema, it generates files conforming to the global schema.

## 3.2. Architecture

One of the main design considerations for this tool was to make it as componentized as possible for future scalability and maintainability. Figure 12 gives a conceptual overview of the architecture of Schema Mapper. Below is a description of the functionality of each component. Section 3.3 discusses implementation details.

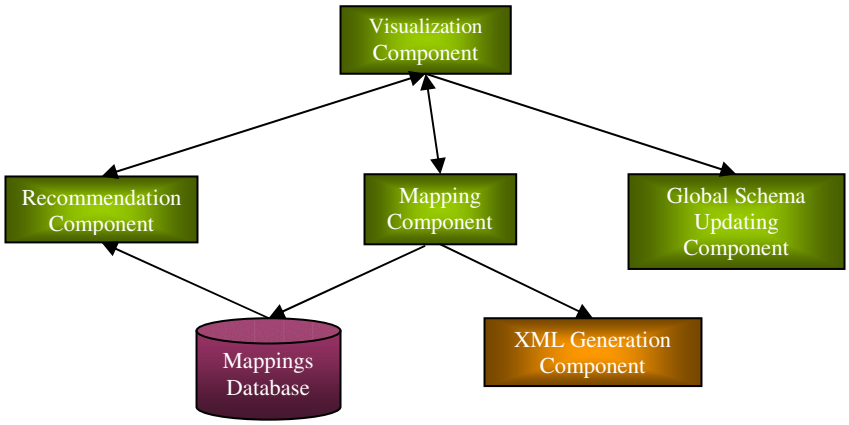


**Figure 12: Architecture of Schema Mapper**

**Visualization Component:**

The Visualization Component contains the logic for generation of the hyperbolic trees and the different color representations to differentiate between the various kinds of nodes (leaf, non-leaf, root, selected/recommended, mapped). It also contains the logic for updating the mapping and recommendation tables and other GUI aspects like displaying context menu options. Presently, this component has logic for representing the schemas as hyperbolic trees. This could easily be replaced in the future by different schema visualizations, without affecting the remaining pieces of the software.

**Recommendation Component:**

The Recommendation Component helps the user in the mapping process by making recommendations of the global schema nodes which are potential matches for a particular local schema node. When the user selects a particular local schema node, the recommendation component uses its recommendation engine to find suitable nodes for mapping and returns this as a list for the Visualization Component to present on screen. The recommendation algorithm used by Schema Mapper currently implements 3 algorithms:

1) **Name-based recommendations:** There are two types of name-based recommendations:

i) **String matching algorithm:** This simply matches the local node name against each of the global node names (ignoring case) and recommends matches.

ii) **Edit Distance Algorithm:** The Edit Distance Algorithm calculates the difference between any 2 strings and returns it as an integer value. By specifying this difference as a parameter, we can employ this algorithm to help select all global nodes whose names exhibit a difference to the local node name that is less than a threshold value.

2) **Recommendations based on Mapping Rules (User–defined rules):** Mapping rules are user-defined rules that the user may wish to specify at the start of a mapping session. These rules can be specified in a separate configuration file. These rules may be specific to the collection that is being mapped into the global schema. For example, the user may specify a rule stating that a local node named "Area" should be mapped to a global node called "PARTITION", while mapping the "Megiddo Collection". The mapping component gets all these rules at the start of the session and makes recommendations based on the given rules.

3) **Recommendations based on Mappings History:** After the user has completed mapping a particular local format to the global format, the mappings for that session are saved into a mappings database as part of the Mappings History. The Recommendation Component loads the mappings history to make recommendations.

**Mapping Component:**

The Mapping Component contains the in-memory representation of all the mappings performed in that session. When the user decides to save mappings and create the style sheet, these mappings are stored in the Mappings History. The mappings also are used to generate the style sheet by the XML Generation Component.

**Global Schema Updating Component:**

During the course of mapping, the user may have modified the global schema. The Global Schema Updating Component keeps the in-memory representation of the schema up-to-date with the changes made by the user. The user may then choose to store the modified global schema to disk by using the "Save Edited Global Schema" or the "Save Edited Global Schema As" feature. This component then serializes the in-memory representation and writes it out to disk.

**XML Generation Component:**

The XML Generation Component takes in the mappings for the current session from the Mapping Component, and also takes in the local and global schema in-memory representations. It generates the XSLT style sheet which forms the wrapper containing the mappings. If a sample local XML file conforming to the local XML schema is

provided, the wrapper is applied to it and the sample output XML file is created and written to disk.

**Mappings History Database:**

The Mappings History Database stores the mappings history. It is updated with the contents of the current mapping session whenever the user decides to save the mappings. This Mappings History Database is accessed by the Recommendation Component to make mapping recommendations for local schema selections made by the user.

## 3.3. Implementation

Schema Mapper has been implemented entirely in Java. We used Java Swing for implementing the GUI. The hyperbolic tree library used for visualizing the schemas is an open source library available for free academic use from sourceforge.net [39]. The database that we used for implementing the Mappings History is a MySQL Database. We proceed to list the important classes and to provide a brief explanation about the functionality of each.

**schemaNode:** This class represents the main data structure that stores schema information. Each schemaNode object contains several data members for representing information about that node. This includes the name of the node, a pointer to the parent of the node, a vector of all the children of this node, the color to be used while visualizing this node, and a few other pieces of information such as whether the node is mapped, highlighted, or recommended. The entire schema is read into an instance of this structure at the very beginning when the user opens a new local schema. The global schema is read into another instance of this structure. These two object instances are used thereafter for the entire mapping process. The actual schema file on disk is not referred to after this first step. This enhances efficiency as the objects are stored in memory. The structure of the schemaNode class is flexible enough so that it can be modified in the future to accommodate other possible pieces of information, as and when required.

**schemaMapperGUI:** This class contains all the visualization logic. It sets up the entire graphical user interface for the tool. An open source library called hypertree has been employed for visualizing the schemas as hyperbolic trees. This class contains the logic for loading schema files into the appropriate data structures for manipulation as well as for visual use. It also acts as the driver for features such as renaming and deleting nodes, making recommendations, mapping and unmapping nodes, and finally saving the mappings. The GUI logic populates the recommendation table each time new recommendations are made and clears out the old ones. It also takes care of updating the mapping and recommendation tables each time a mapping is complete. This class is therefore responsible for making calls to the recommendation component and the mapping component. In essence, it glues together the different components and acts as the driver for the entire mapping process.

**schemaEdit:** This class contains all of the logic for actually editing the global schema. It contains the logic for making modifications to the Document Object Model (DOM) [40], which is also an in-memory representation of the schema. The global DOM is kept up-to-date with the changes that the user makes. Once the user decides to save the edited schema, this class serializes the DOM and writes it out to disk. We prefer having two separate in-memory representations, schemaNode (for visualization) and the DOM (for schema editing), in order to have a loose coupling between the various components.

**MappingComponent:** This class contains the hashtable for storing the mappings in the current session. It also provides accessor and mutator methods to the same, for the GUI to access and modify the mappings depending on the user's actions. Once the user decides

to save the mappings for the current session, the mapping component writes out the current session's mappings to the database. It also instantiates the XSLTgen object for generating the XSLT style sheet.

**RecommendationComponent:** This class contains the logic for recommending global nodes for mapping based on selected local nodes by the user in the GUI. It contains the logic for the string matching algorithm and also accesses the Mappings History database to make recommendations based on the mappings history. It also instantiates the EditDistance object and uses it to calculate the difference between the local and global node names and hence to recommend nodes.

**EditDistance:** This class contains the logic for the Edit Distance Algorithm. When given two strings it calculates the difference between the two strings. For example, if we are trying to find the difference between "Tree" and "Tre", this algorithm will return the difference as 1.

**XSLTgen:** This class contains the logic for generating the XSLT style sheet, given the local and global schemas and the hashtable of mappings for that current session. Once it generates the style sheet, if a local XML file is specified, it instantiates the XSLTStreamTest object for generating the output XML file.

**XSLTStreamTest:** This class contains the logic for transforming the XML file conforming to the local schema to the file conforming to the global schema, by applying the style sheet generated by the XSLTgen object.

**Mappings History Database:** The Mappings History database contains the following data tables that are outlined as follows:

***XPathMappingTable:*** This table contains the XPath [41] expressions for the local and global mapped nodes and a unique ID for each entry. Here the primary key is this ID, which is auto-incremented.

***FileNamesTable:*** This table contains as the foreign key the ID field of XPathMappingTable and contains the local and global schema file names. Here the primary key is formed by all three of the fields.

***LocalFileTable:*** This table contains the local schema file name as the primary key and the corresponding local schema file.

***GlobalFileTable:*** This table contains the global schema file name as the primary key and the corresponding global schema file.

The LocalFileTable and the GlobalFileTable are mainly used as a redundant store for the schema files. This would help create a version of the schemas wherein one can see the progress of the schema over time as more and more collections keep getting added into the ETANA-DL Union Catalog.

# 4. MEGIDDO COLLECTION CASE-STUDY

Having discussed the requirements, architecture, and implementation of Schema Mapper, we now proceed to use the Megiddo [5] Collection as a case-study for using Schema Mapper to integrate new collections into the ETANA-DL Union Catalog. In this chapter, we start off by giving a brief introduction to the Megiddo Collection. We then explain the "incremental approach to schema mapping", thereby justifying the need for schema editing functionality through actual examples. Finally, we describe two sample scenarios of integrating the Megiddo Flint Tool and Vessel artifacts into the ETANA-DL Union catalog, and explain the mapping process step-by-step.

## 4.1. Introduction to Megiddo Collection

Megiddo excavation data integration is used as a case study to demonstrate our approach to archaeological data integration. Megiddo is widely regarded as the most important archaeological site in Israel from Biblical times, and as one of the most significant sites for the study of the ancient Near East.

The excavation data collection we received from Megiddo is stored in more than 10 database tables containing over 30000 records with 7 different types, namely wall, locus, pottery bucket, flint tool, vessel, lab item, and miscellaneous artifact. The Megiddo schema is described in a structure sub-model (see Figure 13) within the 5S framework (Streams, Structures, Space, Scenarios, and Societies) [4].

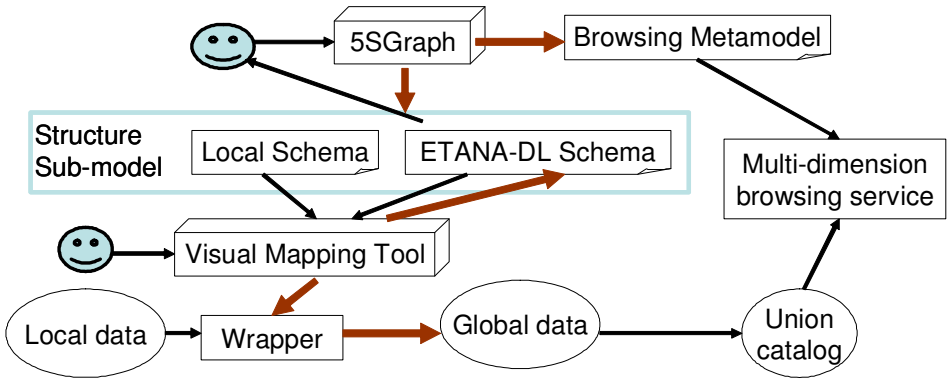


**Figure 13: Process of Integrating Local Archaeological Data into ETANA-DL**

Structures represent the way archaeological information is organized along several dimensions; it is spatially organized, temporally sequenced, and highly variable [42]. Figure 13 also describes the overall process of integrating a new collection into the Union Catalog. In the figure the thin black lines indicate input whereas the thick brown lines indicate output. Consider site organization (see Figure 14). The structures of sites evidence a containment relationship at every level of detail, from the broadest region of interest to the smallest aspect of an individual find, in a simple and consistent manner. Generally, specific regions are subdivided into sites, normally administered and

excavated by different groups. Each site is subdivided into partitions, sub-partitions, and loci, the latter being the nucleus of the excavation. Materials or artifacts found in different loci are organized in containers for further reference and analysis. The locus is the elementary volume unit used for establishing archaeological relationships [43].

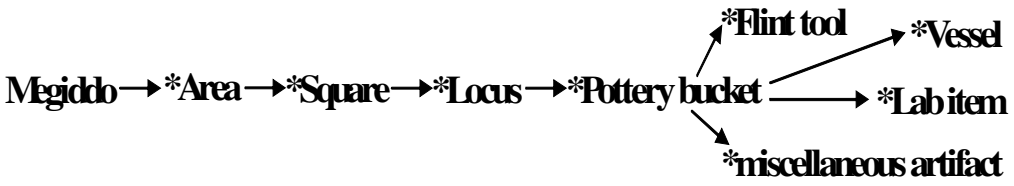


**Figure 14: Megiddo Site Organization**

## 4.2. Incremental Approach to Data Integration

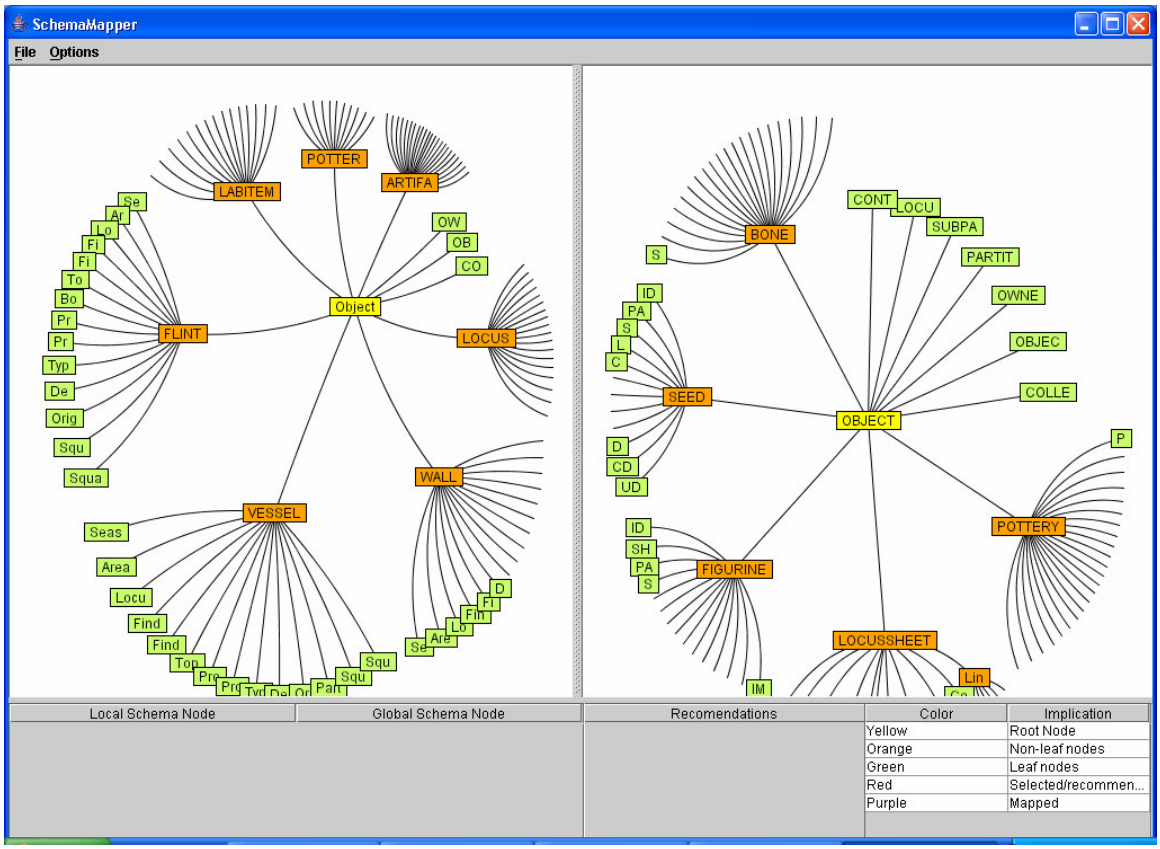


**Figure 15: Screenshot of Schema Mapper Representing the Megiddo Collection Format (left hand side screen) and the ETANA-DL Global Format (right hand screen)**

Figure 15 shows a screenshot of Schema Mapper with the Megiddo local schema (on the left) and the ETANA-DL global schema (on the right). The global schema specifies the attributes for artifact types like Bone, Seed, Figurine, LocusSheet, and Pottery. However, it does not identify the attributes for artifacts like Flint Tool, Vessel, Wall, PotteryBucket, etc. that are contained in the Megiddo Collection. A Vessel artifact is totally different from a Pottery artifact, so we cannot expect to map Megiddo-vessel to the global Pottery.

The only way of resolving this problem is by adding Flint Tool, Vessel, Wall, and all the other artifacts in the Megiddo collection as separate sub-trees into the global schema. Thus with each new collection being integrated into the Union Catalog, the global schema keeps getting incrementally enriched. This reinforces the importance of the schema editing requirement in Schema Mapper. However, this does not mean that each new artifact which needs to be integrated should simply be added as a new sub-tree. There are certain attributes that are common to all objects which are available as leaf nodes of the OBJECT root node in the global schema. We need to map the attributes of

our new artifact to these attributes first and then add the remaining attributes which are specific only to this artifact as a separate sub-tree.

For the Megiddo Collection, we identified the following rules for mapping these attributes.

- OBJECTTYPE - > OBJECTTYPE
- OWNERID - > OWNERID
- COLLECTION - > COLLECTION
- Area - > PARTITION
- Locus - > LOCUS
- OriginalBucket - > CONTAINER
- Square1 - > SUBPARTITION

The rule states the mapping from the local schema node (left hand side of arrow) to the global schema node (right hand side of arrow).

## 4.3. Sample Mappings

We now walk you through the entire process of generating a style sheet through mapping a given local to a global schema. As described earlier, the Megiddo local schema consists of seven different types of artifacts. For integrating items into the union DL, we produce one style sheet of mappings per type. For the purpose of integration of the Megiddo collection into the global schema, we first consider mapping of "flint tool" and then use the knowledge of these mappings to map "vessel".

### 4.3.1. Flint Tool Collection

Figure 15 shows the Megiddo collection as the local schema, along with the ETANA global schema. The objective is to map the flint tool collection into the global schema. Based on the rules previously defined, the following mappings are recommended: OWNERID->OWNERID, OBJECTTYPE->OBJECTTYPE, COLLECTION->COL-LECTION, AREA->PARTITION, Square1->SUBPARTITION, Locus->LOCUS, and OriginalBucket->CONTAINER. We choose to accept these recommendations and map the nodes. Figure 16 shows a screenshot after completion of the initial set of mappings.

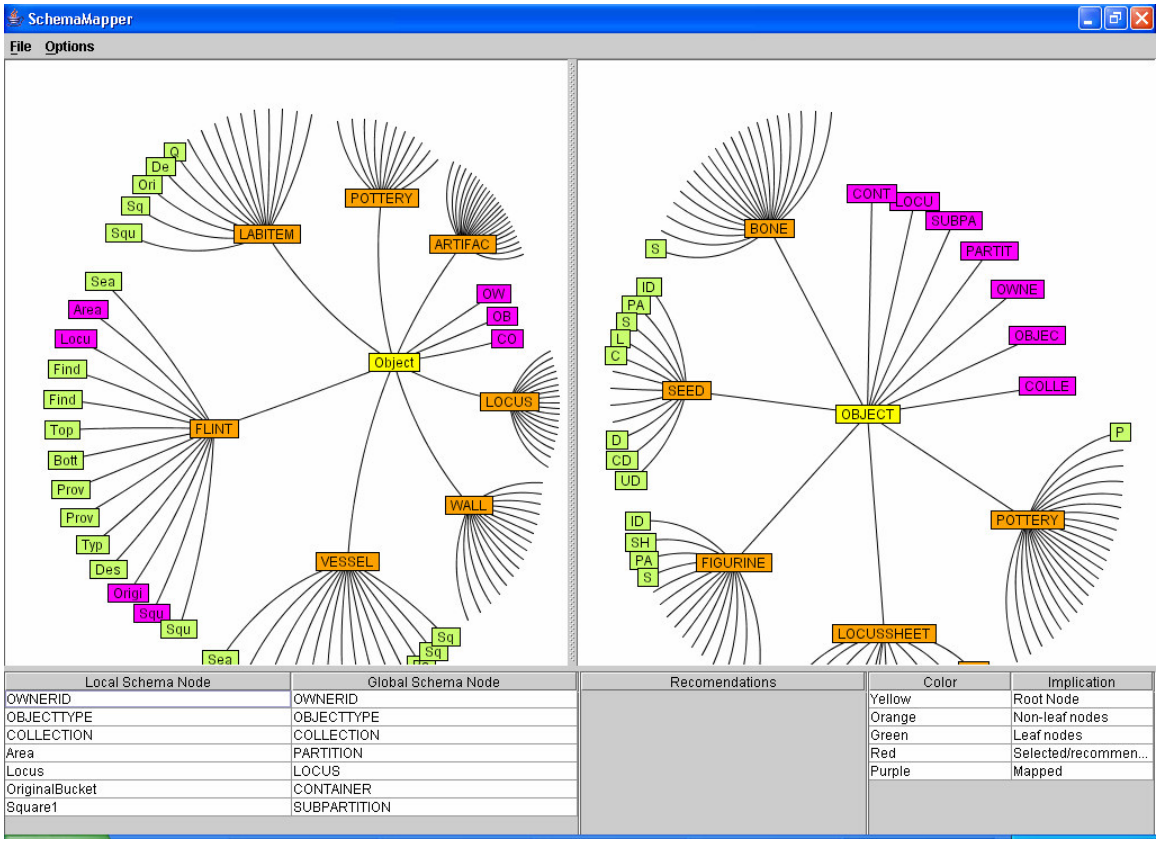


**Figure 16: Initial Set of Mappings Suggested By Recommendations Based on Rules**

As the remaining nodes in the local schema do not have corresponding global schema nodes, we add the flint tool sub-tree as a child of the OBJECT node in the global schema.

This ensures that local schema elements and properties are preserved during the mapping transformation. Schema Mapper determines that some of the nodes (Area, Locus, OriginalBucket, and Square1) are already mapped, deletes these nodes from the global schema sub-tree, and automatically maps the rest with the corresponding elements in the local sub-tree (see Figure 17).

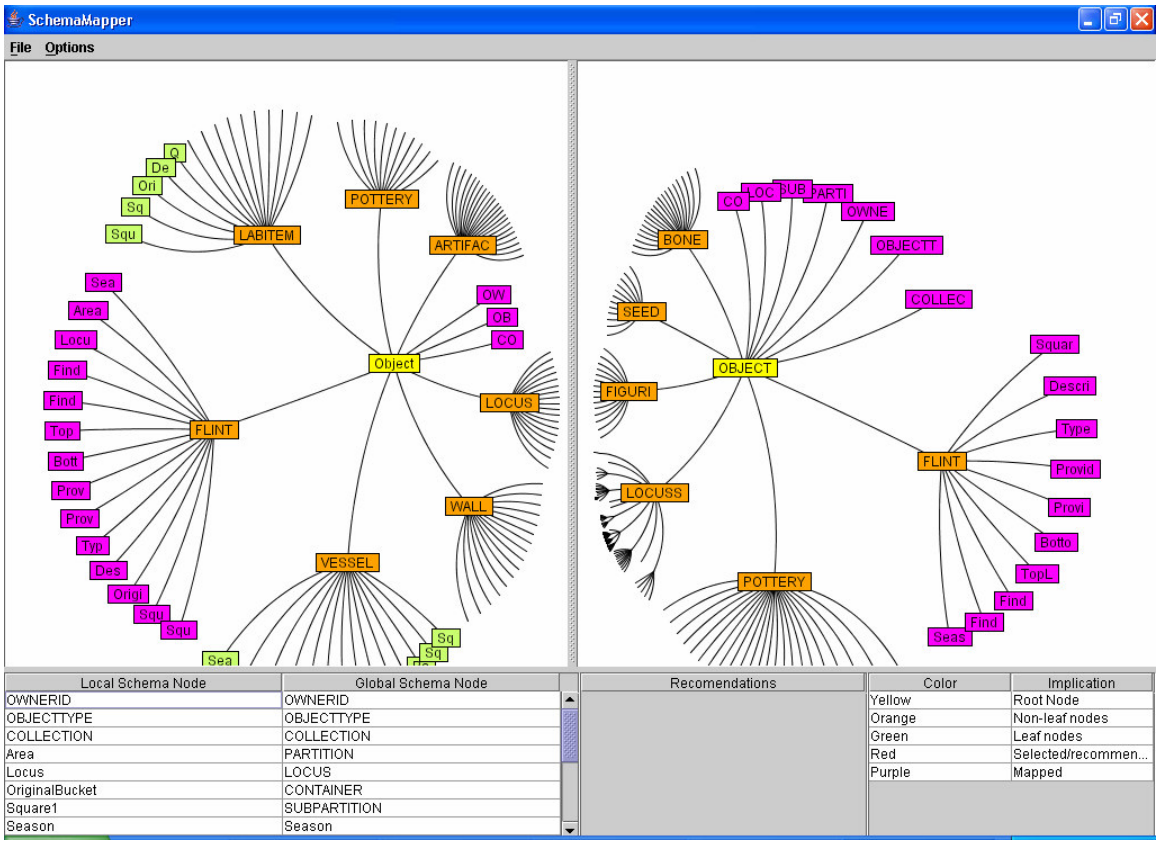


**Figure 17: Adding Flint Tool as a Child of OBJECT in the Global Schema**

We may decide to rename some nodes in the global schema from within this sub-tree to avoid any local connections with the name. Let us assume that we choose to rename global schema node Description to DESCRIPTION. We may then decide to confirm the mappings by selecting the Save Mappings option (see Figure 18). An XSLT style sheet with the mappings is then generated (see Figure 19). The mappings are stored in the Mappings History database. We also may "Save the Edited Global Schema", and the ETANA global schema is updated with the flint tool schema. With this the mapping process is complete.

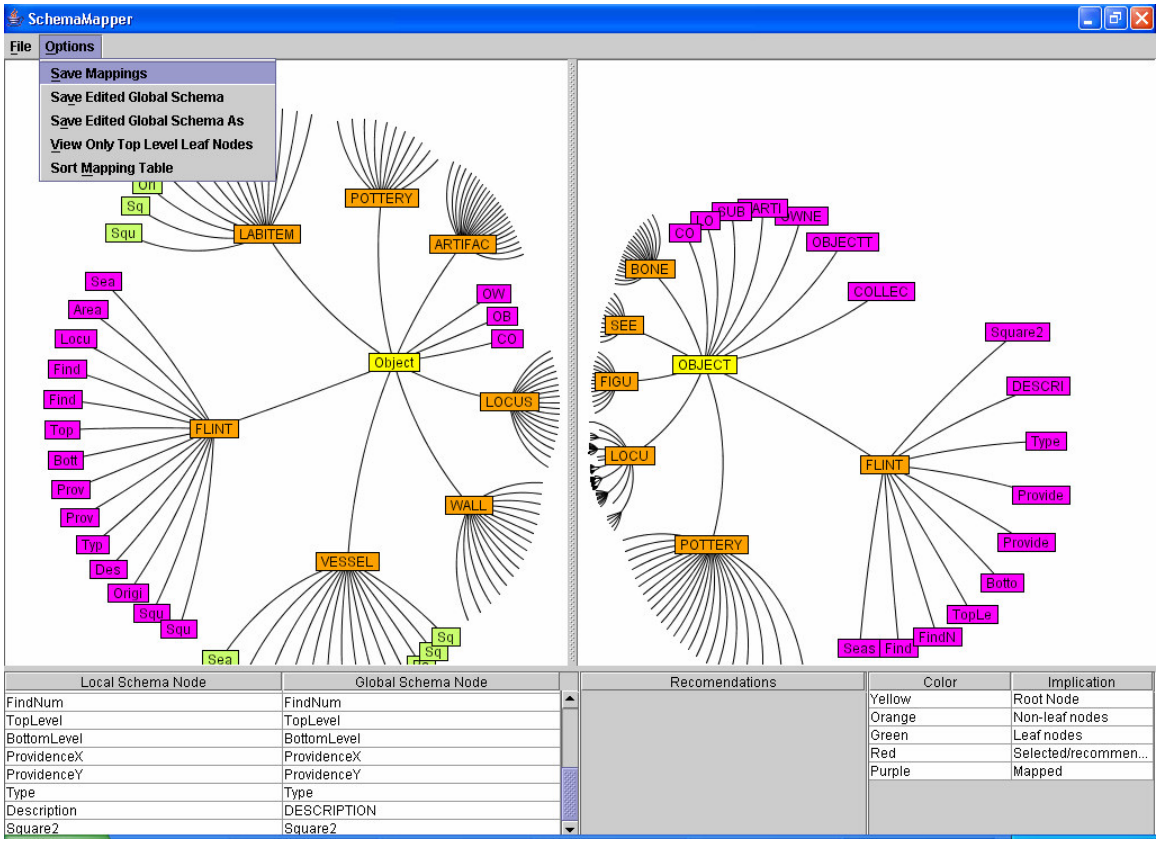


**Figure 18: Description Node in the Global Schema Renamed to DESCRIPTION and the User Choosing to Save Mappings**

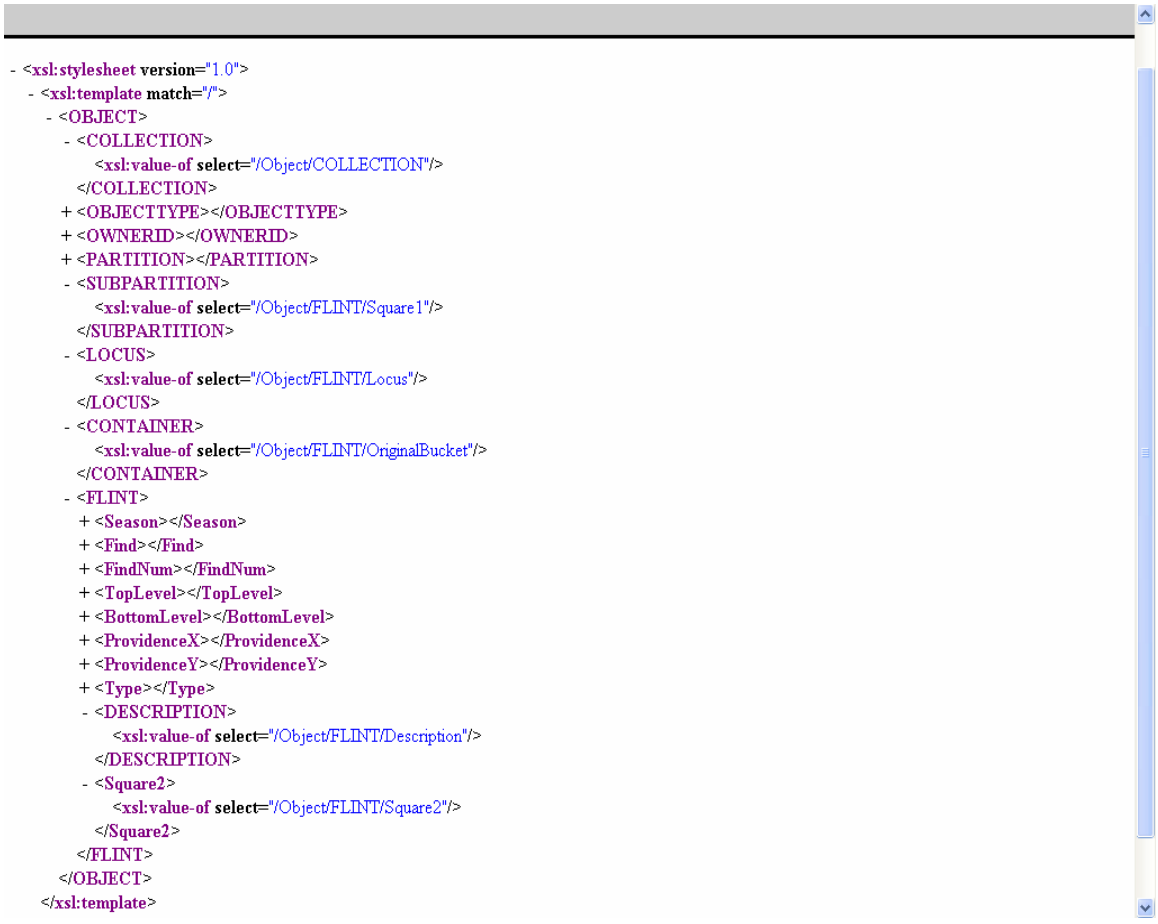


**Figure 19: Flint Tool Style Sheet Generated by Schema Mapper**

### 4.3.2. Vessel Collection

We next integrate the VESSEL artifact of Megiddo into the ETANA global schema. We open the local and global schemas for mapping earlier. We can see that Flint Tool is now a part of the global schema (see Figure 19). As we now want to add the Vessel sub-tree as a new collection, we need not visualize the remaining collections in the global schema. This would avoid erroneous cross mappings from schema nodes of one of the artifacts to similar schema nodes present in other artifacts. This also prevents us from accidentally modifying a node, from say the flint tool sub-tree in the global schema, and rendering the previously generated XML files inconsistent. We may choose the option to view only top level leaf nodes for this purpose (see Figure 20). Also, this avoids confusing us by allowing us to visualize only what we need for mapping purposes. Once again recommendations are made to enable the initial set of seven mappings; after this, we add the VESSEL sub-tree to the global schema.

As before, Schema Mapper finds that the Area, Locus, Square1, and OriginalBucket nodes are already mapped – and deletes them in the global sub-tree and maps the remaining nodes to corresponding local schema nodes automatically. Schema Mapper

also goes through the mappings history and finds that the Description node in the flint tool sub-tree was mapped to the DESCRIPTION node in the global schema. In order to keep naming consistent, Schema Mapper recommends the user to change the name of the Description node in the VESSEL sub-tree to DESCRIPTION (see Figure 21). This is due to the fact that both the DESCRIPTION node in the flint tool sub-branch of global schema and the Description node in the VESSEL sub-branch of the global schema describe the artifact, but as DESCRIPTION has been selected as the global name, all Description elements in the global sub-tree should be renamed as DESCRIPTION. The recommendation, as always, is not mandatory, but if followed will help keep names consistent.

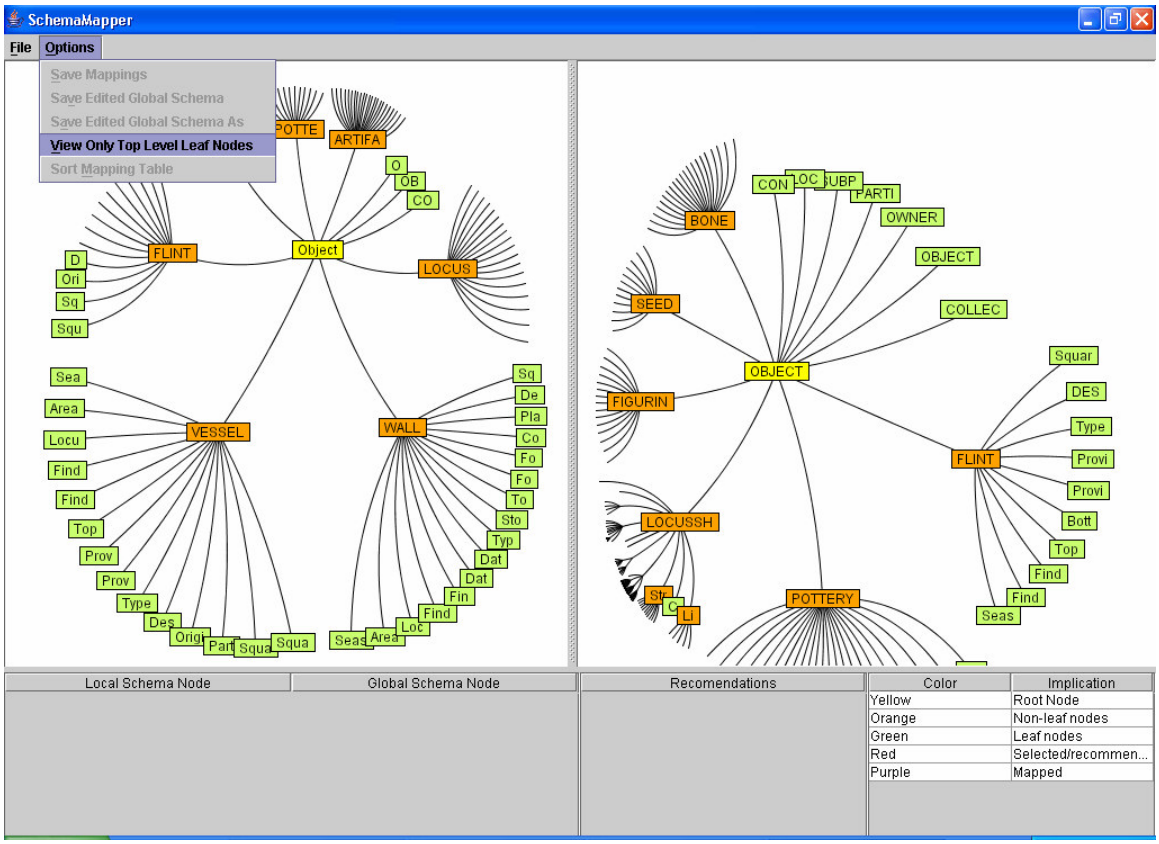


**Figure 20: View Only Top-level Leaf Nodes Option**

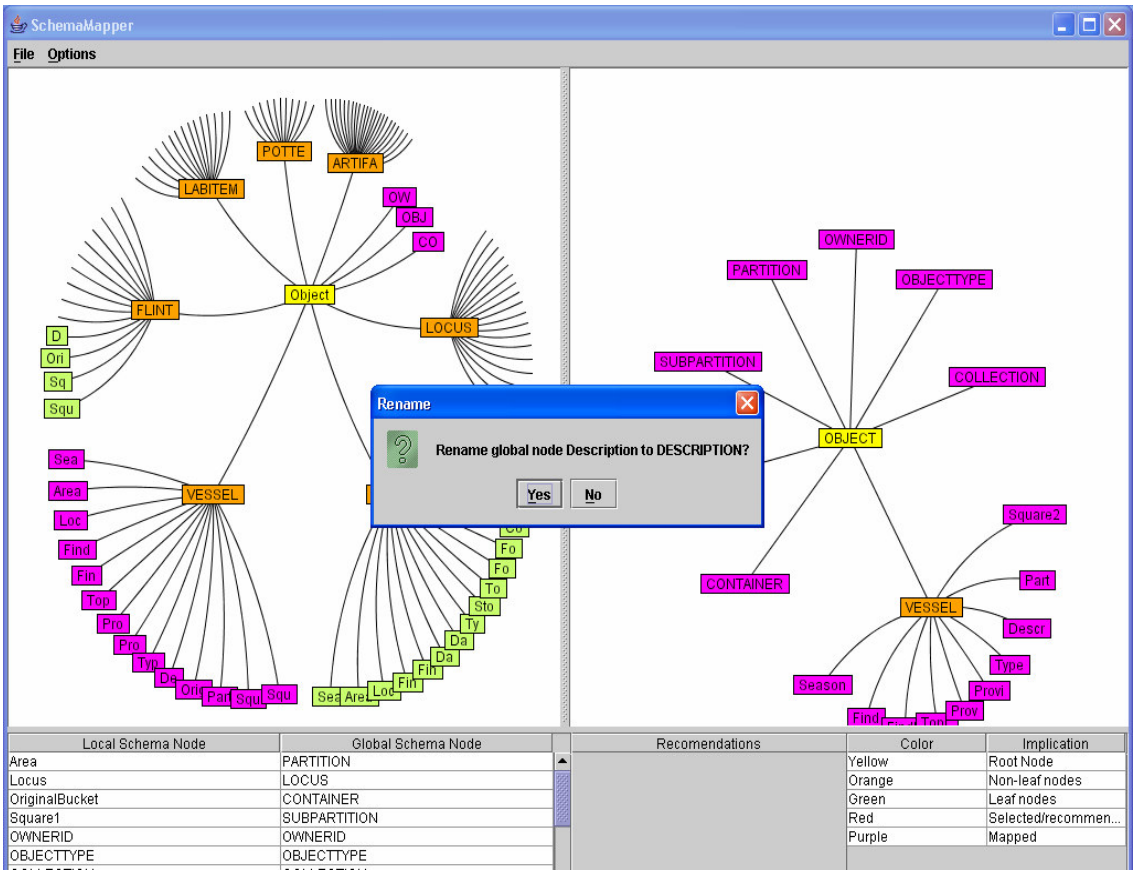


**Figure 21: Name Change Recommendation by Schema Mapper**

When we confirm the mappings, the database is updated and the style sheet generated as with the flint tool collection. We then may choose to save the edited global schema, and the global schema gets updated with the VESSEL schema. It is important to note that the integration of vessel artifacts into the global schema in no way changes the existing flint global entry. This leads us to the observation that modification of the global schema is simply appending a new local artifact into the global schema without changing the existing global artifacts.

In this manner, we may integrate all the 7 artifacts that are a part of the Megiddo Collection. The ETANA-DL global schema, after the integration of all the 7 artifacts, is shown in Figure 22. The style sheets generated are then applied to the XML files corresponding to the respective artifacts, and the global XML files are generated. These are ready for harvesting into the union DL, and available for access by services like Searching and Browsing.

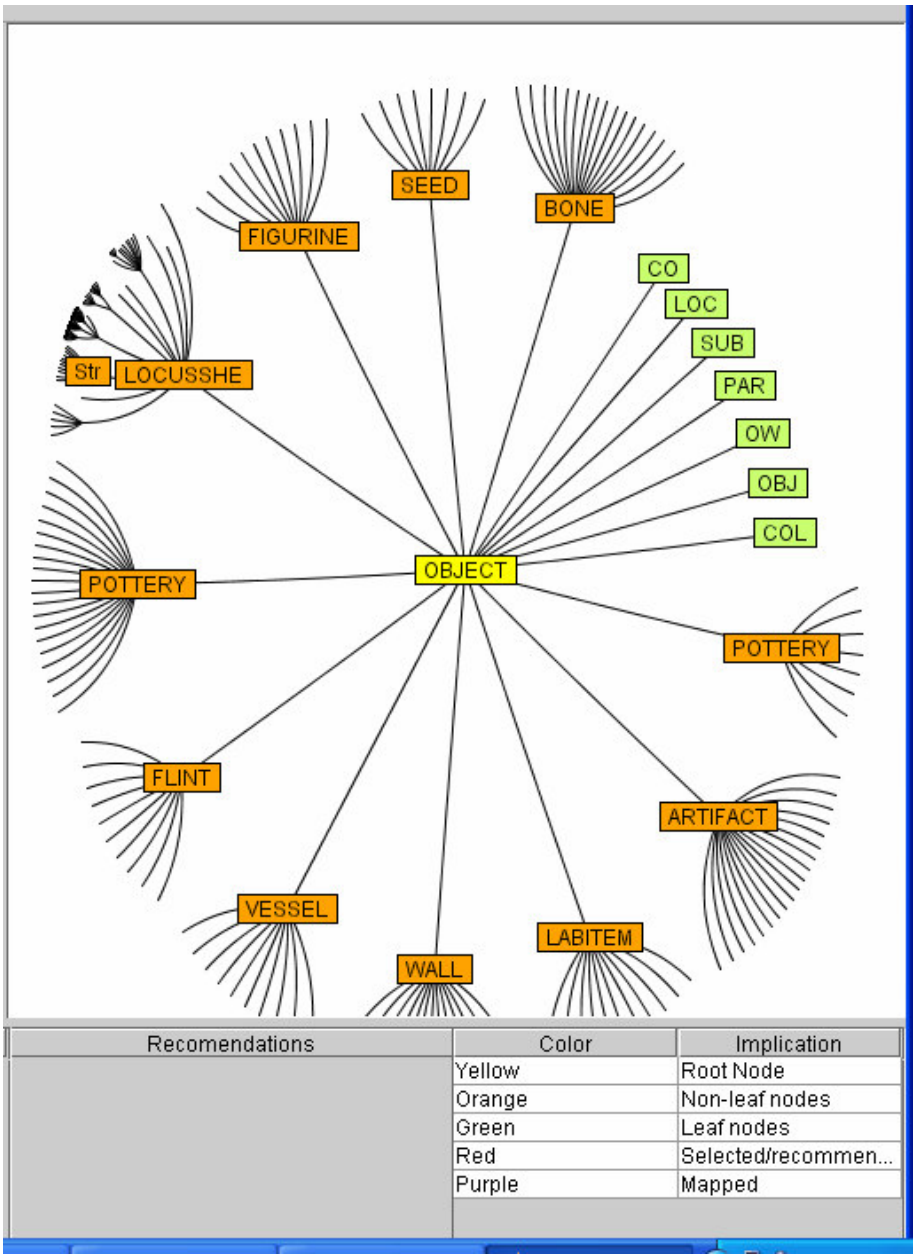


**Figure 22: ETANA-DL Global Schema after the Integration of the Megiddo Collection.**

# 5. DATA-LEVEL MAPPING

In this chapter, we describe one of the applications that Schema Mapper can be put to because of its flexible design. We shall start off by describing the concept of data-level mapping and its use in the ETANA-DL. We proceed to describe the format for the XML files for data-level mapping. We wrap up this chapter by giving sample data-level mappings for the Tell Nimrin collection.

## 5.1. What is Data-Level Mapping

ETANA-DL receives collections from various archaeological sites which need to be integrated into its Union Catalog. These collections are usually received as databases containing the information in data tables. Each of these collections has thousands of records. Some errors occur in the spelling of the data in these tables. Sometimes some entries are missed. Sometimes data is entered as abbreviations. Sometimes certain terminologies are used for representing data values for certain elements in the global XML files. When similar data comes in as part of a new collection, this terminology needs to be extended to these data values too, for consistency. For all such cases, there needs to be a pre-processing step that needs to be performed before the XML files are allowed to be harvested into the Union Catalog.

Schema Mapper could be used to solve this issue of Data Mapping. By specifying rules especially for terminologies, recommendations could be made to extend these terminologies to the newly integrated data. For example, let us say that an element Period may have values “Persian”, “Islamic”, “Modern Bronze”, etc. Let us further assume that a mapping history exists containing the entry MB - > Modern Bronze. When a new collection needs to be integrated into the Union Catalog, if the global XML files contain the entry MB for the element Period, Schema Mapper can recommend that it be mapped to Modern Bronze. This illustrates the concept of Data Mapping.

The output of Data Mapping, unlike schema mapping, is a tab delimited text file which contains the local data value and the corresponding global data value. This text file is then given as the input to the pre-processor which accordingly replaces the local data value for that particular element (tag) in the global XML files with the global data value from the text file. This ensures consistency in the representation of data to the people who access the Union Catalog through the various services exposed by the ETANA-DL.

## 5.2. XML Format for Data-Level Mapping

For schema level mapping, Schema Mapper visualizes XML Schema Descriptions files (.xsd), whereas for data-level mapping, Schema Mapper takes as input a particular format of XML files. Data-level mapping can basically be used for each element (XML tag) that has a limited range of values. We continue to use the hyperbolic tree representation for data-level mapping, however here there is no real hierarchy involved. The root element would be the XML tag element and the leaves would be all the data values that this element could have. Thus there are just two levels that the input format will contain for either the local or the global XML files.

The format is as shown below:

< Element-name >
    <element1 name = “value 1”/>
    <element2 name = “value 2”/>
    <element3 name = “value 3”/>
    …….
    …….
    …….
< / Element-name >

Schema Mapper takes in a local XML file conforming to this format. It represents as its root the element “Element-name”. This root element will have as many children as there are in the local XML file. However these children will have their names as “value 1”, “value 2”, “value 3”, and so on. Basically Schema Mapper looks for the attribute values for each element tag and displays that as the node name for each of the hyperbolic tree nodes except for the root node. Figure 23 shows a sample XML file corresponding to the above format and Figure 24 shows the corresponding hyperbolic tree representation using Schema Mapper.

A few changes needed to be made to Schema Mapper in order to be able to provide a Data Mapping feature. These changes included the logic for hyperbolic tree representation of data files conforming to the above mentioned format. Also the GUI provided additional options to open local and global XML files for data-level mapping. There were little or no changes to the other components, once again highlighting the flexibility of the design of Schema Mapper. For the sake of a clean design we decided to have new data tables analogous to the data tables for schema mapping, the only change being that we were no longer storing the XPath expressions for each of the nodes, instead we were storing the node names themselves. (See Section 3.3 for the Mappings History database.) This was because XPath expressions do not make sense for such a flat representation.

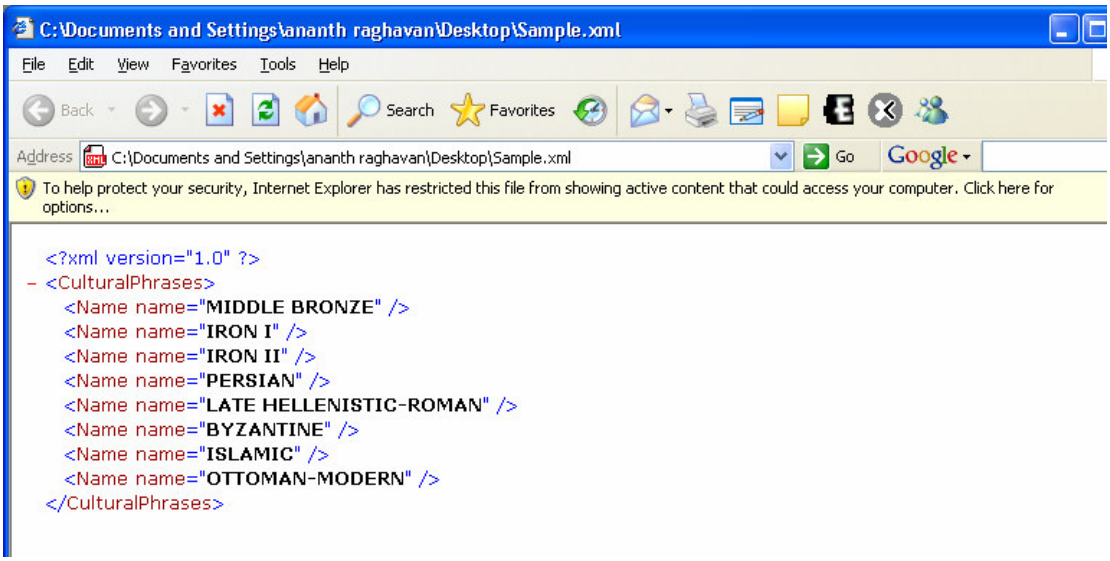


**Figure 23: Sample XML File Conforming to Format Required by Schema Mapper**

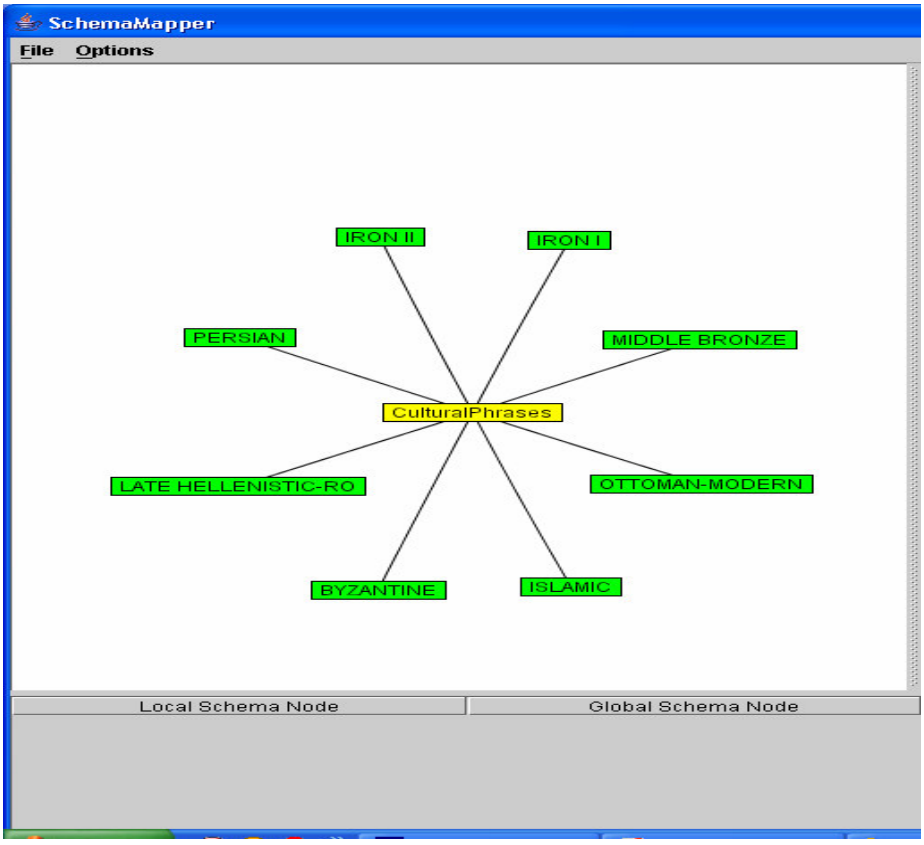


**Figure 24: Schema Mapper Representing Local XML File for Data Mapping**

## 5.3. Sample Mappings

In this section, we give an actual example of data-level mapping using Schema Mapper. We shall take a look at the two XML files to be mapped, the mappings history, and the mapping rules. We will follow the mapping process, and finally have a look at the output text file generated.

The Mappings History basically consists of the following mappings that have been made in the past:

| **Local Value** | **Global Value** |
|---|---|
| Null | UNKNOWN |
| EB | EARLY BRONZE |
| EB3 | EARLY BRONZE III |
| MB2 | MIDDLE BRONZE II |
| MB2C | MIDDLE BRONZE IIC |
| MB2-LB | MIDDLE BRONZE II-LATE BRONZE |
| LB | LATE BRONZE |
| IR | IRON |
| EIR1 | EARLY IRON I |
| EIR2 | EARLY IRON II |
| LIR2 | LATE IRON II |
| LIR2/P | LATE IRON II/PERSIAN (IRON III) |
| PER | PERSIAN |
| HEL | HELLENISTIC |
| CLAS | CLASSICAL |
| CLASS | CLASSICAL |
| ER | EARLY ROMAN |
| CLASS/ISL | CLASSICAL-ISLAMIC |
| CLASS-ISL | CLASSICAL-ISLAMIC |
| BYZ/ISL | BYZANTINE-ISLAMIC |
| ISL | ISLAMIC |
| LISL | LATE ISLAMIC |
| MOD | OTTOMAN-MODERN |
| IRON | IRON |
| Modern | OTTOMAN-MODERN |

**Table 1: Mappings History Table for Tag Element CulturalPhrases**

Figure 25 shows a screenshot of Schema Mapper used to open local and global XML files for the CulturalPhrases element.

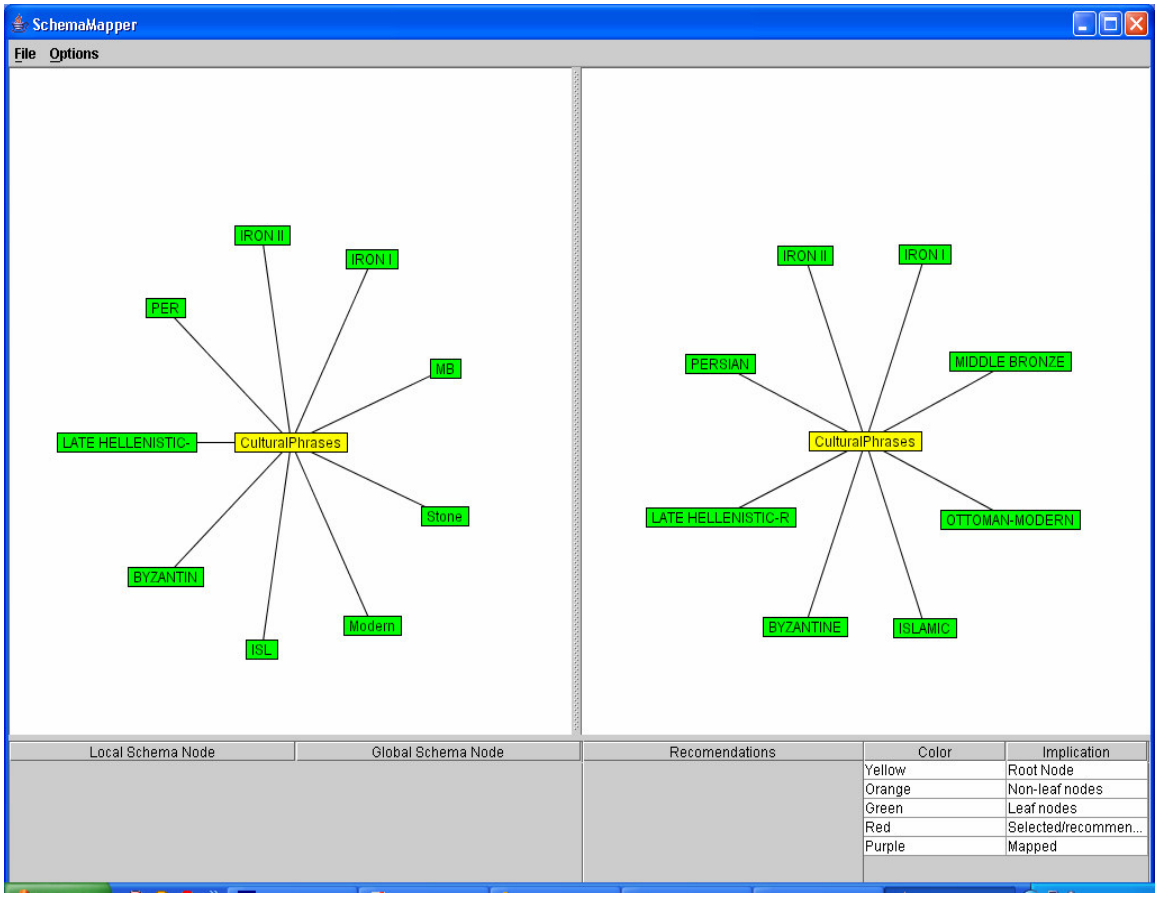


**Figure 25: Local and Global XML Files Containing the Data for CulturalPhrases Element.**

The user then proceeds to map the elements. An example for each category of recommendation is discussed here. When the user selects LATE HELLENISTIC-ROMAN in the local XML file, the LATE HELLENISTIC-ROMAN node is highlighted in the global XML file; this is due to the string matching algorithm. Similar reasoning applies for the mappings IRON I - > IRON I and IRON II - > IRON II.

When the user selects PER in the local file, PERSIAN is recommended in the global file because of the Mappings History (see Figure 26). Similar reasoning applies for the mappings ISL - > ISLAMIC and Modern - > OTTOMAN-MODERN. MB is recommended to Middle Bronze because of a user defined rule. BYZANTINE is recommended for local node BYZANTIN because of the edit distance (see Figure 27).

Finally, stone is added as a child to the global schema and renamed as STONE to create a mapping stone - > STONE (see Figure 28). This new mapping will be stored in the mappings history for future recommendations. When the user selects Save Mappings, the tab delimited text file containing the mappings is created (see Figure 29).

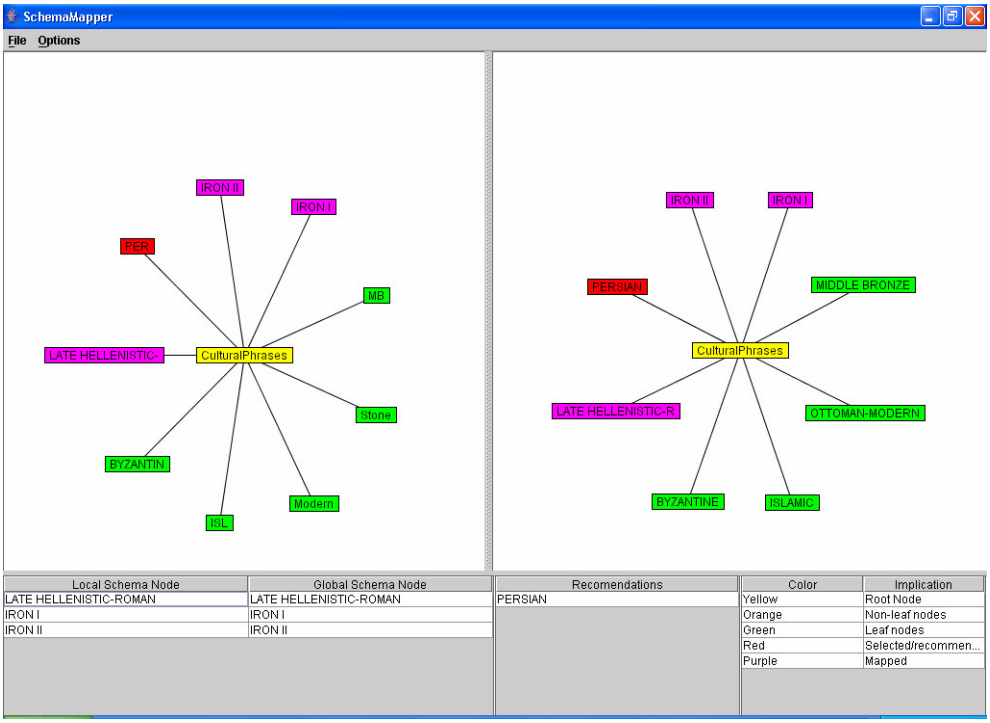


**Figure 26: PERSIAN Recommended (in red color) for Selection PER because of Mappings History**

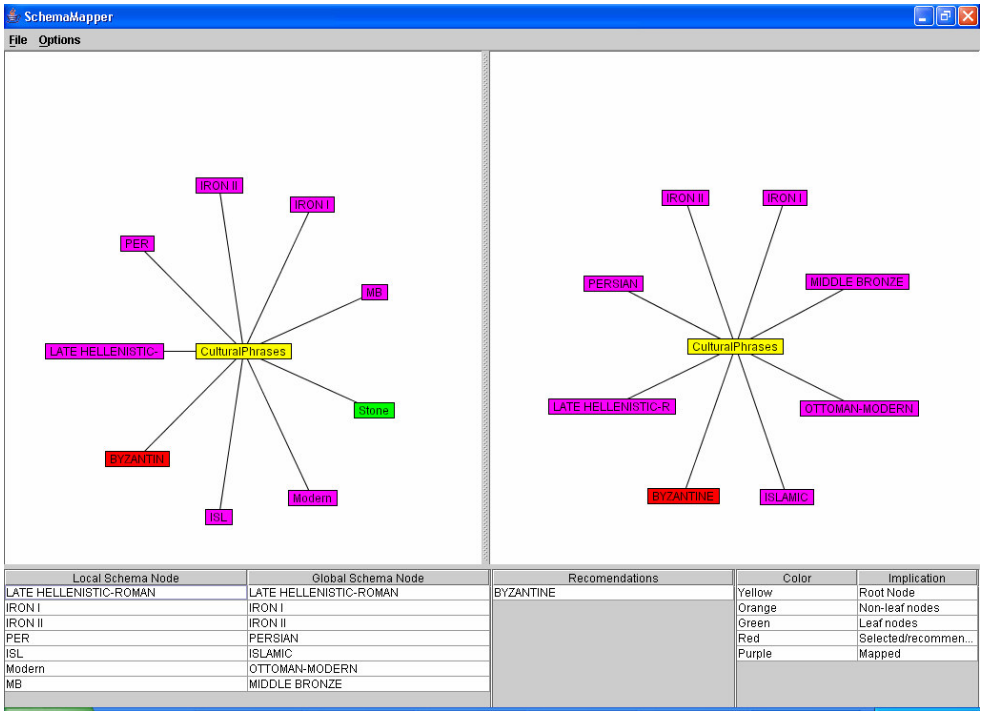


**Figure 27: BYZANTINE Recommended for Selection BYZANTIN because of Edit Distance Algorithm**

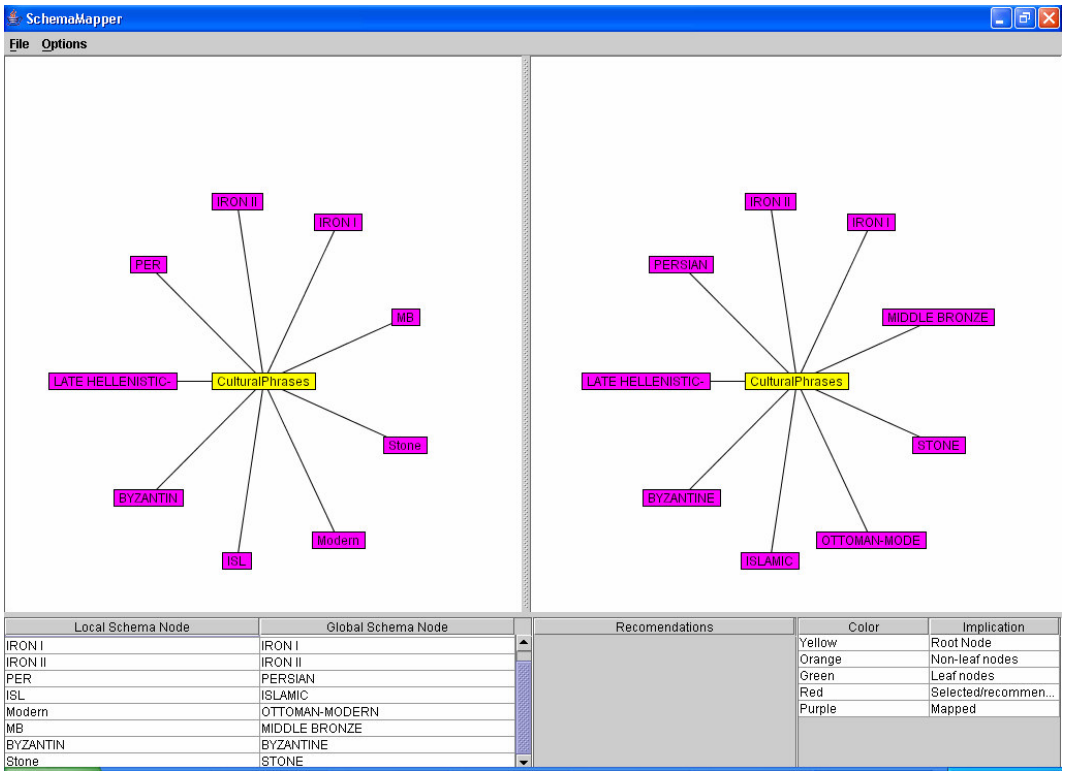


**Figure 28: Stone Added as a Child and Renamed as STONE in the Global XML file**

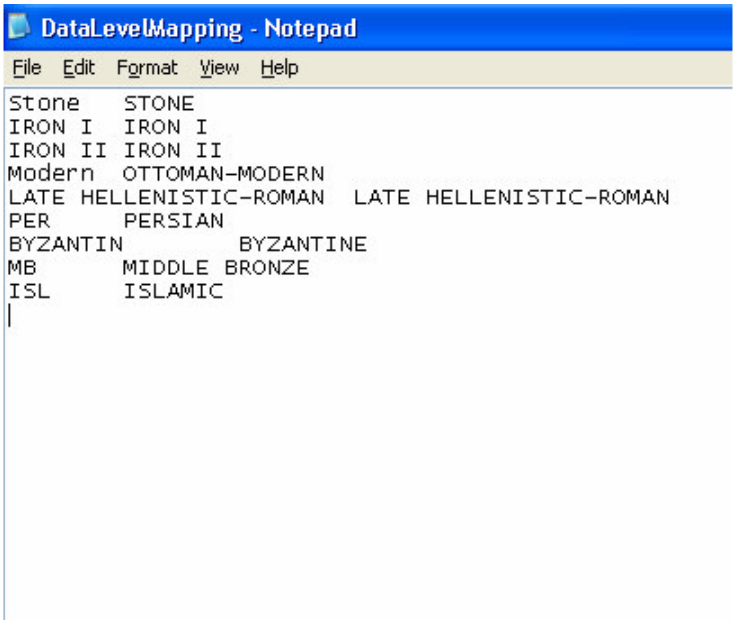


**Figure 29: Output Tab Delimited Text File Containing the Data-Level Mappings**

# 6. USABILITY EVALUATION

Usability Engineering is a branch of Computer Science that is based on the premise that describing the interaction from the user's view should result in more usable software than describing it from the programmer's view. [44]. Usability Evaluation forms a part of Usability Engineering wherein we choose to evaluate the usability of the software developed. There are 3 types of Usability Evaluation (Testing, Inspection, and Enquiry) [45]; we chose the Testing method for performing the evaluation.

In this chapter we discuss the usability evaluation of the GUI of Schema Mapper. We start off by identifying the goals of the usability evaluation through a Claims Analysis. We then describe the experimental set-up and the user demographics. We next discuss each of the benchmark tasks, along with the corresponding qualitative and quantitative results. We mention the Critical Incidents that occurred during the various benchmark tasks and analyze their criticality. We conclude by resolving the top 3 Critical Incidents.

## 6.1. Claims Analysis

There are mainly two types of Usability Testing; Formative and Summative. A Formative Evaluation is used to guide the design process and help re-design features of the User Interface according to feedback received from the users. A Summative Evaluation on the other hand is conducted after the development or at a major checkpoint in order to find out how well we have performed [46].

We conducted a Formative Evaluation in order to make improvements to the GUI. We also wanted to find out show-stopper bugs in the GUI of Schema Mapper and fix the same. For performing a usability evaluation it is important to identify the goals of the evaluation and the means to ascertain whether we have achieved these. Accordingly, we came up with a list of goals (claims) that we had to achieve and performed a Claims Analysis.

Rosson and Carrol state that ***"A claim is a hypothesis about the effect of the features on user activities"*** [47]. Claims Analysis helps us design Benchmark Tasks that test the exact features of the user interface that we want to evaluate through usability testing. In Claims Analysis, we make a claim about a particular feature of the user interface. We go on to mention the pros and cons associated with this feature. Our usability testing thus gets reduced to testing our claim against the drawback and exploring this trade-off.

Table 2 below states the claims, the trade-off that may be involved, a method to explore the trade-off, and how we justify the claim.

| No. | Goal (Claim) | Pros | Cons | Method to explore the trade-off |
|---|---|---|---|---|
| 1 | Schema Mapper takes 15% less time than linear representation with no recommendations. | Hyperbolic tree representation allows for a greater number of nodes to be displayed at a time. Recommendations allow users to locate a node faster for mapping. | Hyperbolic trees may prove confusing to the user and users may not have a sense of context within the hyperbolic tree and hence may not prefer this. | Design a Benchmark Task (BT), which requires users to do mapping using Schema Mapper (hyperbolic trees + recommendations) and compare the performance against MapForce (linear representation + no recommendations). Here time and number of errors are the metrics to compare the performance. Also ask the user through questionnaire as to which of the 2 methods s/he preferred. |
| 2 | Schema Mapper allows for 30% reduction in the number of scrolls required by linear representation. | Hyperbolic tree representation avoids the problem of scrolling. | But it introduces the problem of re-orienting the tree to visualize all the nodes. | During the BT for mapping, count the number of scrolls while using MapForce vs. the number of re-orient actions required with Schema Mapper. |
| 3 | Schema Mapper allows for 10% faster recognition of mappings than a representation that has lines across the screen. | Representing mappings as a table on screen rather than lines connecting nodes across the screen makes the search faster and accurate. | Lines across the screen may be more intuitive than a separate mapping table. | Design a BT that asks the user to identify mappings using MapForce and using Schema Mapper. Measure the time taken and number of errors. Also ask the user as to which of the 2 methods s/he felt was less confusing |
| 4 | Schema Mapper takes 25% lesser time than separate mapping and | Schema Mapper provides both mapping and editing capability | Separate tools may both be more intuitive to use than | Design a BT that explores the time taken by the user to use Schema Mapper to |

| | | | | |
|---|---|---|---|---|
| | editing tools. | that reduces time in switching between tools. | Schema Mapper. | edit the global schema and map a collection against the time taken using MapForce and XML Spy. Also a question may be asked to the user as to which of the two approaches s/he felt was more intuitive. |

**Table 2: Claims, Trade-offs, and Method to Explore These Trade-offs**

## 6.2. Experimental Set-up and User Demographics

The experiment consisted of 4 Benchmark tasks designed to explore the trade-offs discussed in Section 6.1 and justify our claims. A total of 9 users performed the tasks. In order to have a level playing field, 4 people were asked to use MapForce first for each of the tasks and then Schema Mapper, while the remaining 5 were asked to do the tasks using Schema Mapper first and then MapForce. This avoids biasing the results in favor of the tool which is used second time round. The experiments were performed in the DLRL here in VT, on a lab machine where a 30-day evaluation version of MapForce was installed for conducting the experiments. The schemas that were used for this purpose were actual schema examples from the ETANA-DL.

Out of the 9 users there were 5 men and 4 women. 5 of these users had prior experience with XML and XML editing using XML Spy. 3 out of the 9 users had used MapForce before and were familiar with it. 2 out of these 9 users were familiar with Schema Mapper, having contributed their ideas during the prototyping of the tool. For the users unfamiliar with XML editing using XML Spy, a separate instruction sheet was provided with detailed instructions on how to use XML Spy for editing the schema as demanded by the benchmark tasks. The users were given a background regarding each of the mapping tools and were allowed to play with the tools to become comfortable with them before performing the Benchmark Tasks.

Users 3, 6, 8, and 9 used MapForce first and then Schema Mapper for their BTs. The remaining users used Schema Mapper first, followed by MapForce.

## 6.3. Benchmark Tasks, Results and Discussions

### 6.3.1. BT 1

Use one of the tools to do the following:

Open cemetery.xsd as the local schema, and etana1.1.xsd as the global schema. Proceed to map the following to recommended global schema nodes (if using Schema Mapper), or to the nodes with the same name (if using MapForce):

- OwnerID
- Volume
- ObjectType
- PotteryImageURL
- Collection
- Height

Repeat BT 1 using the other tool to map the above 6 nodes to corresponding global schema nodes.

Questions:

1) Was it easier to locate the 6 local schema nodes in Schema Mapper or in MapForce?

2) Was it easier to locate the 6 global schema nodes in Schema Mapper or in MapForce?

**Quantitative Results for BT 1:**

| User | Time using Schema Mapper (minutes) | Time using MapForce (minutes) | Number of re-orient actions (Schema Mapper) | Number of scrolls (MapForce) | Number of errors (Schema Mapper) | Number of errors (MapForce) |
|---|---|---|---|---|---|---|
| **1** | 1:10 | 1:40 | 4 | 6 | 0 | 0 |
| **2** | 2:40 | 3:10 | 3 | 9 | 0 | 0 |
| **3** | 1:13 | 1:42 | 4 | 9 | 0 | 0 |
| **4** | 2:15 | 2:45 | 2 | 12 | 0 | 0 |
| **5** | 1:00 | 2:35 | 2 | 14 | 0 | 0 |
| **6** | 0:43 | 1:48 | 2 | 13 | 0 | 0 |
| **7** | 1:16 | 1:55 | 2 | 3 | 1 | 0 |
| **8** | 1:00 | 1:25 | 2 | 10 | 1 | 0 |
| **9** | 0:54 | 1:23 | 5 | 8 | 0 | 0 |

**Table 3: BT 1 Results**

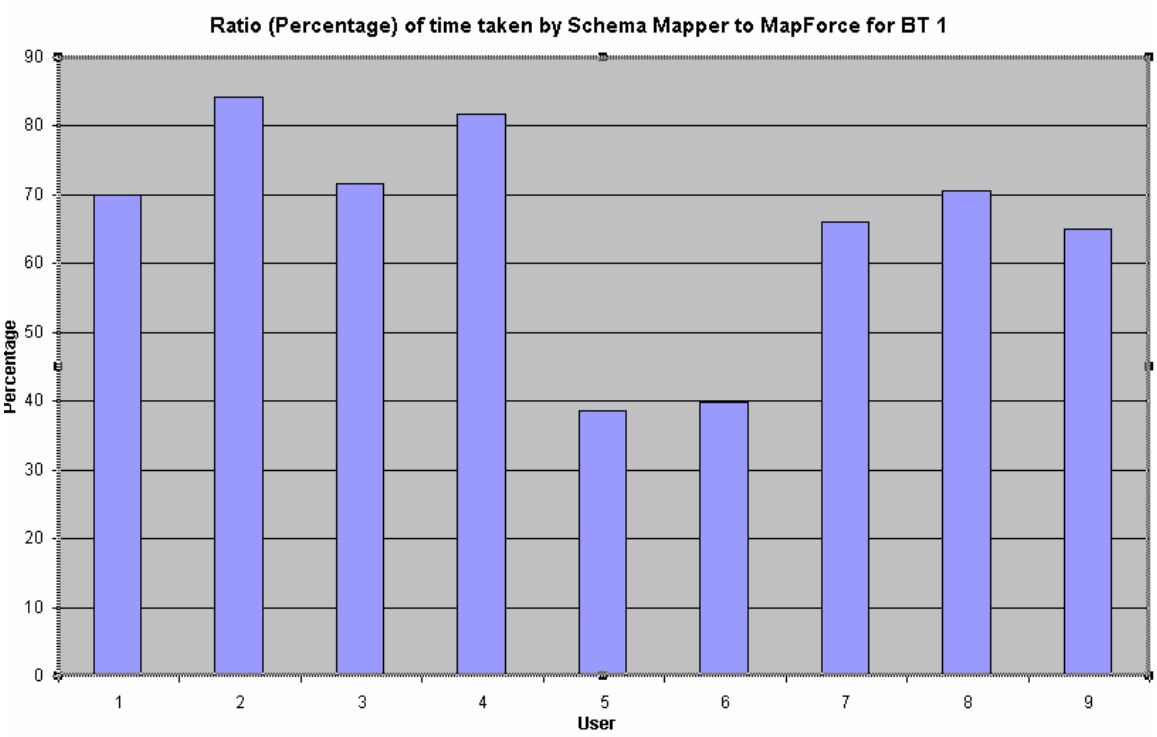


**Figure 30: Ratio (In Percentage) of Time Taken using Schema Mapper to that using MapForce for BT 1**

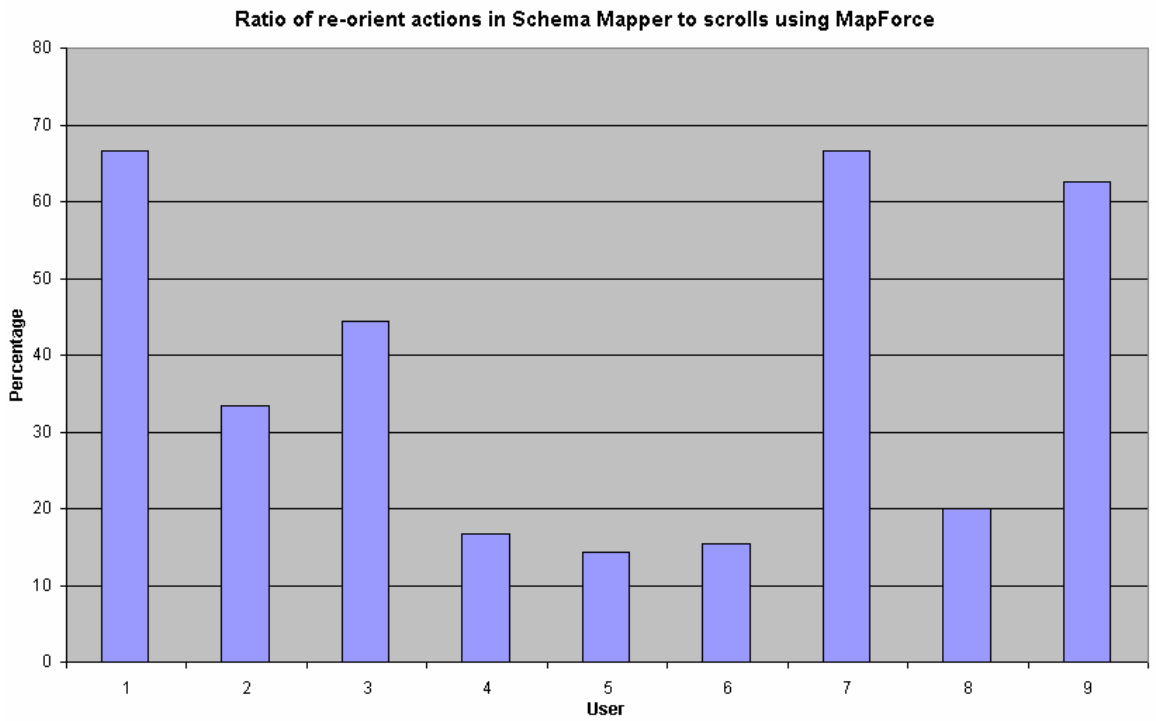


**Figure 31: Ratio (In Percentage) of Re-orient Actions using Schema Mapper to Scrolls using MapForce**

**Discussion of Quantitative Results:**

Table 3 shows the Quantitative Results for BT 1, including the time taken for Schema Mapper and MapForce, the number of re-orient actions for Schema Mapper and the number of scrolls using MapForce, and the number of errors using both the tools. A comparative analysis was performed on this raw data to come up with Figures 30 and 31.

Figure 30 shows the ratio in percentage of the time taken by Schema Mapper to that taken by MapForce for completing BT-1. We can see that in general while using Schema Mapper the users took between 65-85% of the time they took with MapForce. There are however two outliers namely users 5 and 6 who both took much longer while working with MapForce. The reason they offered was that Schema Mapper recommended nodes that made their job a lot easier for detecting nodes for mapping, which was not the case with MapForce.

Figure 31 indicates the ratio of the re-orient actions of Schema Mapper to that of MapForce. As we can see, for all users Schema Mapper took lesser than 70% of the number of scrolls required by MapForce, which once again satisfies our goal. The graph seems to indicate that there is a significant amount of standard deviation in the ratio. However from Table 3, we can observe that users performed 2-5 re-orient actions and the large standard deviation is from the range in the number of scrolls while using MapForce, namely 3-14.

Users 8 and 9 made one error each while using Schema Mapper, which is noted as a critical incident. As the hyperbolic tree representation does not allow the entire node name to be displayed, both these users ended up selecting the wrong local node "Height_to_mouth_ratio" instead of the node "Height" in the local schema. This problem may be fixed by either displaying the selected local schema node name in a separate table at the bottom and highlighting this display, so as to draw attention to the user about his selection, or to have a pop-up with the selected local node name displayed on screen. Both the users, however, realized their errors from the entry into the Mapping table and undid the wrong mappings, which reduces the criticality of the error.

**Discussion of Qualitative Results:**

8 out of the 9 users felt that Schema Mapper was better for locating local schema nodes, whereas one user felt that it was the same for both Schema Mapper and MapForce, their reasoning being that they did not have to scroll as much (and hence not search as much) as in MapForce. 2 out of these 8 users however felt that not being able to see the complete node name was a drawback, as far as locating local schema nodes is concerned.

All 9 users felt that locating global schema nodes was definitely easier using Schema Mapper, mainly because there were recommendations and also as less scrolling was involved. This fact makes us think that probably providing a "find" feature for local schema nodes would help, especially those users who know which node they are looking

for. Overall, the advantages of the hyperbolic tree, along with the concept of recommendations, outweighed the disadvantage of not being able to see the whole node name.

Thus we can see that our claims 1 and 2 (see Table 2 for claims) are justified, namely that mapping is at least 15% faster using a hyperbolic tree with recommendations than with a linear representation with no recommendations, and that a hyperbolic tree definitely reduces the problem of scrolling found in linear representations.

### 6.3.2. BT 2

Use one of the tools to do the following:

Open Megiddo.xsd as the local schema and etana1.1.xsd as the global schema. The objective is to map the flint sub-tree of the local schema to the global schema.

Instructions for Schema Mapper:

Proceed to map the following:

a) OwnerID - > OWNERID
b) ObjectType -> OBJECTTYPE
c) Collection -> COLLECTION
d) Area -> PARTITION
e) Locus -> LOCUS
f) OriginalBucket -> CONTAINER
g) Square1 -> SUBPARTITION

Add flint sub-tree as a child of global schema node OBJECT.
Rename all global nodes belonging to the flint sub-tree with ALL CAPS (all letters in capitals).

Instructions for MapForce:

Proceed to map the following:

a) OwnerID - > OWNERID
b) ObjectType -> OBJECTTYPE
c) Collection -> COLLECTION
d) Area -> PARTITION
e) Locus -> LOCUS
f) OriginalBucket -> CONTAINER
g) Square1 -> SUBPARTITION

1) Use XML Spy to edit the global schema and add flint tool as a sub-tree. While adding, note that the flint tool sub-tree in the global schema should not contain the

nodes Area, Locus, OriginalBucket, and Square1. (Please refer to the separate sheet given to you for instructions on how to do the same if you are not familiar with XML editing.)

2) Rename all the global nodes in the flint tool sub-tree of the global schema with ALL CAPS.
3) Using MapForce map the remaining local schema nodes in the flint tool sub-tree to corresponding nodes in the global schema sub-tree.

Questions:

Which of the 2 approaches did you find more intuitive and easy to use (Schema editing by MapForce / Schema editing by Schema Mapper)? Please indicate in particular if schema editing in Schema Mapper was confusing?

**Quantitative Results for BT 2**

| User | Time taken by Schema Mapper (minutes) | Time taken by MapForce (minutes) | Number of errors in Schema Mapper | Number of errors in MapForce |
|---|---|---|---|---|
| 1 | 4:30 | 8:02 | 1 (rename) | 0 |
| 2 | 6:15 | 7:13 | 0 | 2 |
| 3 | 4:10 | 9:25 | 1 (rename) | 5 |
| 4 | 8:50 | 8:55 | 5 (rename) | 2 |
| 5 | 4:30 | 7:30 | 0 | 0 |
| 6 | 2:59 | 6:48 | 0 | User actually lost all his earlier mappings |
| 7 | 4:50 | 6:30 | 2 (rename) | 1 |
| 8 | 2:59 | 6:19 | 1 (rename) | 0 |
| 9 | 3:25 | 6:58 | 2 (rename) | 1 |

**Table 4: BT 2 Results**

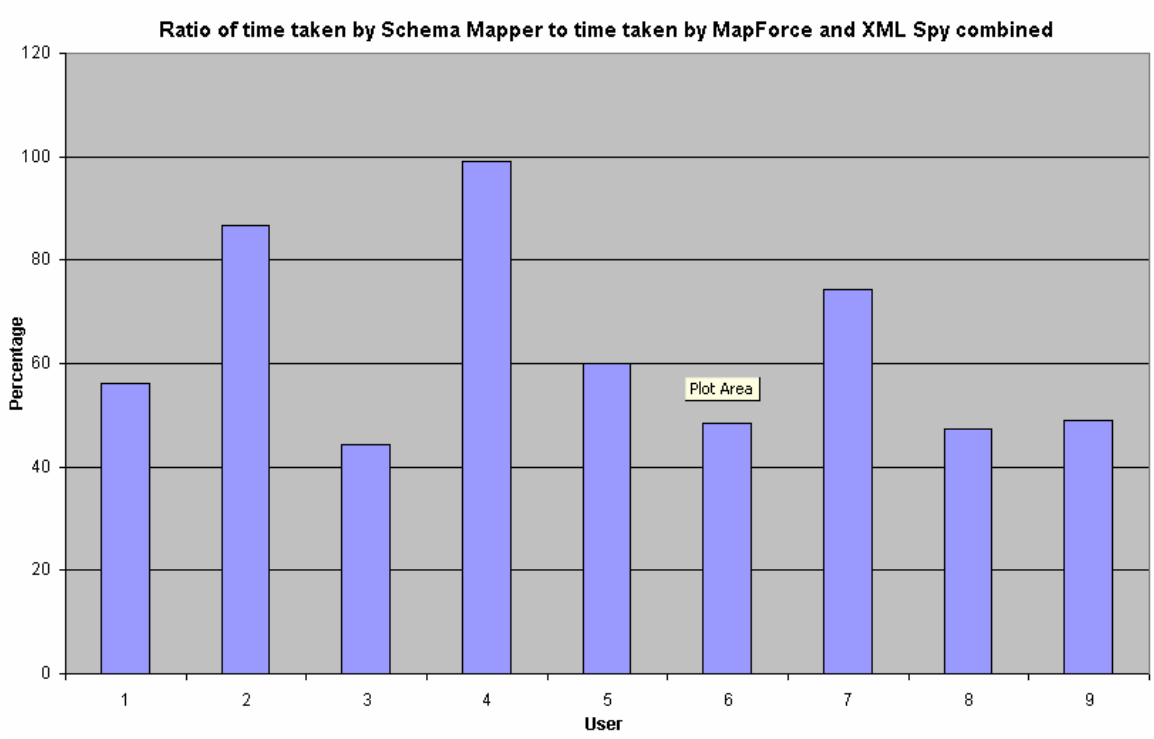


**Figure 32: Ratio (In Percentage) of Time Taken using Schema Mapper to Time Taken using MapForce and XML Spy Combined for BT 2**

**Discussion of Quantitative Results:**

Table 4 shows us the quantitative results for BT 2. We can see that Schema Mapper seems to have a clear advantage in terms of the time required to map when editing the schema is involved. A comparative analysis of the data in Table 4 gives us the overview of Figure 32.

We can see from the figure that all the users except for users 2 and 4 completed the task using Schema Mapper well within 75% of the time they took using MapForce and XML Spy combined. User 4 took a much higher time using Schema Mapper and this was mainly because of the 5 errors committed during the renaming process. Thus this was a critical incident or a show stopper bug that prevented the user from performing BT 2 fast enough and hence resulted in us not achieving our goal for this user.

Also, all the errors encountered in Schema Mapper were because of one reason. During Rename, we do not show the old name, as the Benchmark task required the users to change the name to the same name with uppercase letters, and also as the node name does not appear in full, many users renamed the nodes erroneously (including user 4). This error can be easily rectified by specifying the node name in the Rename box. This way the user will know what the old name was and not commit the error. This is a high priority fix as many users made errors because of the absence of the old node name.

**Discussion of Qualitative Results:**

All users unanimously agreed that schema editing using Schema Mapper was easier as compared to using XML Spy and MapForce. However there were a few suggestions which are worth mentioning:

- Allowing rename in the mapping table for the global schema node, the corresponding change would then be reflected on the hyperbolic node. This would however be a costly feature to implement as one column of the table should be made editable whereas the local schema node name column should not be editable.
- Having a group rename option at the parent level in the global schema, which would pull up a separate window containing an editable list of all the children node names. On changing the list the node names in the tree also will be changed.

Hence from the Qualitative and Quantitative Results we can conclude that claim 4 is justified for 7 out of the 9 users. We also found the reason why it was not justified for the remaining 2 users, namely the problem with the Rename box, and provided a fix for the same.

### 6.3.3. BT 3

Use one of the tools to do the following:

Locate the following mappings (for each, point the mapping out to the evaluator)

1) Mapping for Area
2) Mapping for Square 2
3) Mapping for Original Bucket

Use the other tool and repeat BT3 (again point out the mapping to the evaluator).

**Questions:**
Which of the two approaches did you like better? (MapForce indicating with lines from local to global schema / Schema Mapper representing mappings as a table)

**Quantitative Results for BT 3:**

| User | Time taken by Schema Mapper (seconds) | Time taken by MapForce (seconds) |
|---|---|---|
| 1 | 34 | 10 |
| 2 | 10 | 20 |
| 3 | 14 | 17 |
| 4 | 25 | 20 |
| 5 | 12 | 23 |
| 6 | 22 | 23 |
| 7 | 17 | 35 |
| 8 | 13 | 16 |
| 9 | 12 | 38 |

**Table 5: BT 3 Results**

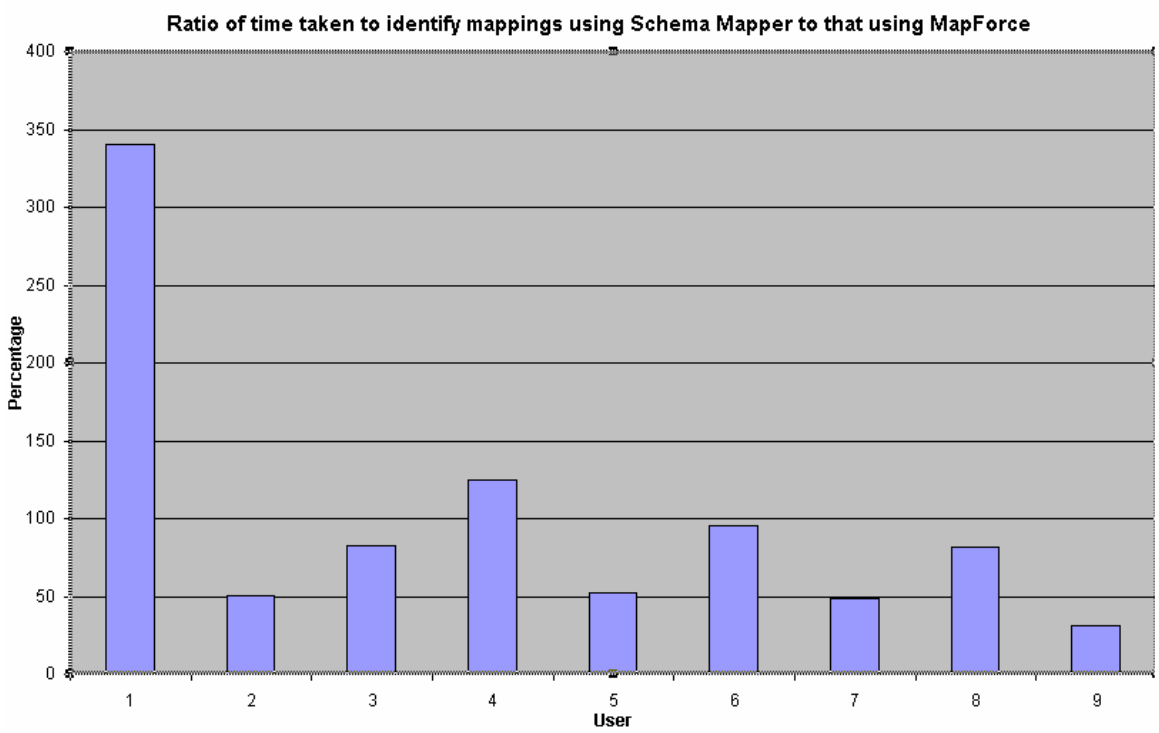


**Figure 33: Ratio (In Percentage) of Time Taken to Identify Mappings using Schema Mapper to that using MapForce**

**Discussion of Quantitative Results:**

Table 5 shows results of BT 3. From the table we can see that 7 out of 9 people performed the task faster using Schema Mapper than using MapForce. A comparative

analysis of the individual times of Schema Mapper and MapForce was performed to come up with the results shown in Figure 33. There were a couple of anomalies where users were slower using Schema Mapper than using MapForce (Users 1 and 4). Also user 6 took almost the same time using both tools.

All these users had a problem in that they could not locate a node in the mapping table fast enough, as it was not a sorted list. Providing a sort feature would definitely help in achieving our goal of having users identify mappings faster using Schema Mapper and hence this feature was subsequently implemented.

**Discussion of Qualitative Results:**

All 9 users agreed that identifying mappings in Schema Mapper was easier than using MapForce. The lines across the screen proved to be confusing. 2 out of 9 users reported the wrong mapping initially, then were confused with the lines, and finally pointed to the right mapping.

Thus our claim 3 was justified for 6 out of the 9 users with all 9 indicating that they preferred a mapping table representation over lines across the screen. We were able to identify the reason for the claim not being justified for the remaining 3, and provided a fix for the same.

### 6.3.4. BT 4

(Only Schema Mapper)

You might appreciate the fact that adding newer collections may increase the size of a global schema to such an extent that there might be too many sub-branches and hence many nodes on screen. The evaluator will demo a work-around feature (viewing only a sub-branch / viewing only top-level nodes).

**Questions:**

1. Do you think this feature is confusing as it does not represent the actual global schema structure? Do you see yourself using this feature? If so please indicate that and proceed to answer question 2.

2. Presently there is no one-click way to view all the nodes again. The evaluator will demo an alternate way of getting the original view back (re-opening the local and global schemas). Would this alternate way suffice or is an undo option essential?

**Discussion of answers for above questions:**

All the people unanimously agreed that:

1) They were not confused by viewing only one sub-branch and viewing only top-level leaf nodes. All the people also agreed that they would use the feature

2) only if an Undo option is present to get back the original view. This makes implementation of the Undo option critical. Hence the Undo feature was subsequently implemented.

## 6.4. SUMMARY OF USABILITY EVALUATION

- **Feature needs to be added for getting the original view (or Undo View only top-level leaf nodes or Undo View only this sub-tree).**

- **Rename box should contain the old name of the node to be renamed.**

- **Feature needs to be added to sort the mapping table.**

Subsequently all these features were implemented.

# 7. CONCLUSIONS AND FUTURE WORK

This chapter discusses the conclusions of the thesis work. It discusses how the hypothesis is demonstrated / proven. It also discusses the main contributions of this thesis and gives pointers to the particular sections that talk about these contributions. The Conclusions are followed by the Future Work section, which details the potential extensions to this thesis.

## 7.1. Conclusions

We re-state the hypothesis that was mentioned in the beginning of the thesis:

We hypothesize that:

a) The problem (described in Section 1.1) of integrating heterogeneous data into a Union Catalog of a DL

b) can be partially automated through our approach (described in Section 1.2) of building a flexible GUI-based mapping tool which, through human interaction during the mapping process, generates a wrapper that could then be applied to the data to achieve the necessary transformation and be ready for harvesting into the Union Catalog.

We prove this hypothesis by describing the tool "Schema Mapper" – which is:

- **Scalable:** Demonstrated by integrating the Megiddo Collection containing 7 artifacts and over 30000 records into the ETANA-DL Union Catalog. (See Chapter 4.)

- **Flexible:** Demonstrated by adding data-level mapping feature (see Chapter 5)

- **Manageable:** Demonstrated through the componentized architecture of Schema Mapper (see Sections 3.2, 3.3)

- **Combines visualization and algorithmic components**: Demonstrated through architecture and implementation of Schema Mapper (see Sections 3.2 and 3.3)

- **Provides an easy approach to schema editing:** Demonstrated through examples in the Megiddo Collection case study (see Section 4.3) and results of the Usability Evaluation (see Section 6.3.2)

## 7.2. Future Work

The problem of Schema Mapping is a hard one to solve completely and there are many potential extensions that are worth exploring.

**Extending GUI-Capabilities:**

Schema Mapper is a GUI-based tool. Hence a number of extensions are possible to the GUI which would make the tool more usable and hence the software more effective. These can be stated as follows:

- One of the main problems with the hyperbolic tree is that the node names are not completely visible. This is because the hyperbolic tree minimizes the size of the nodes in order to provide more of an overview of the schema structure itself. An interesting visualization problem would be to be able to retain the larger overview that hyperbolic trees provide and at the same time ensure that essential information like the entire node name not be hidden away from the user.

- Schema Mapper performs better than MapForce for simple 1-1 schema mapping due to its hyperbolic tree representation and the concept of recommendations. An interesting question to explore is: **What is the contribution of each of these factors?** Replacing the hyperbolic tree representation with a linear representation which has recommendation features and turning off recommendations with the hyperbolic tree representation and hence conducting usability studies could provide an insight into what actually works for the mapping problem. Providing options at the UI level to switch on/off these features would definitely help for such studies.

- Another interesting issue to explore is the usage of colors. We are using colors extensively to distinguish between various types of nodes. However, no study has been conducted as to how identifiable these colors are for color blind individuals. Usage of appropriate colors, replacing colors by other highlighting schemes (blinking / change in intensity) are issues that could be explored. Also, the legend in the GUI could contain the actual colors instead of the color name.

**Recommendation Component:**

The Recommendation Component forms the other important part of Schema Mapper. Presently, there are three algorithms being implemented for the recommendation engine. There are several extensions here:

- Implement more recommendation algorithms (matching algorithms), and hence reduce the possibility of there being no recommended node, and thereby improve the mapping efficiency.

- Another possibility is to be able to assign coefficients of effectiveness to each of these algorithms. These may be refined by using a training set (for both schema and data-level mapping). These coefficients could be applied to a test collection, and the recommended nodes could then be ranked according to the measure of similarity assigned to them, due to these coefficients.

**Issue of Complex Mappings:**

- Presently Schema Mapper supports only 1-1, simple schema mapping, and this suffices for the collections of the ETANA-DL. A future extension is to allow complex mappings to be represented. Complex mappings are one-many and many-one mappings. These are made possible through the presence of operators like concatenation for strings, mathematical operators for numbers, etc. Building this functionality both at the GUI and the algorithm level, as well as testing it against collections that would yield to complex mappings, would be a challenging task that is worth exploring.

- This idea could be extended to provide a constraint based system. Here the users can indicate the kind of constraints that they want Schema Mapper to work under. For example, users could specify many-one as the kind of mapping; the representation, say, hyperbolic representation; the set of operators, say concatenation, addition, multiplication etc.; and the recommendation algorithms, say, rule-based, name-based, and mapping history based. This would dictate the mode in which Schema Mapper would operate.

**Summative Evaluation:**

An important future extension to this work could be a Summative Evaluation which compares Schema Mapper with other tools in terms of performance, taking into account time to complete benchmark tasks as well as comparison of errors and learnability issues. Statistical claims could be made about the performance of Schema Mapper with a much larger user base and more rigorous usability studies.

# REFERENCES


[1] Ananth Raghavan, Divya Rangarajan, Rao Shen, Marcos André Gonçalves, Naga Srinivas Vemuri, Weiguo Fan, Edward A. Fox. Schema Mapper: A Visualization Tool for DL Integration. To be presented at Joint Conference of Digital Libraries (JCDL 2005), June 7-11, Denver, Colorado, 2005

[2] U. Ravindranathan. Prototyping Digital Libraries, Handling Heterogeneous Data Sources - An ETANA-DL Case Study. Masters Thesis. Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, April, 2004. http://scholar.lib.vt.edu/theses/available/etd-04262004-153555/

[3] XSLT, 1999 http://www.w3.org/TR/xslt (as of April 2005)

[4] 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, 2004.

[5] Megiddo, 2005 http://www.tau.ac.il/humanities/archaeology/megiddo/index.html (as of April 2005)

[6] Bush, V. *As We May Think*, in The Atlantic Monthly, vol. 176, pp. 101-108, July, 1945.

[7] Digital Library Federation, 2005. *www.diglib.org* (as of April 2005)

[8] Digital Libraries definitions, 1999 http://feathers.dlib.vt.edu/~etana/Publications/ETANAArticleSHAJ.pdf, (as of April 2005)

[9] Drabenstott, K. M. "Analytical review of the library of the future", Council on Library Resources, Washington, DC, 1994.

[10] MARIAN DL Information System, 2005 http://www.dlib.vt.edu/products/marian.html (as of April 2005).

[11] Marcos André Gonçalves, Robert K. France, and Edward A. Fox. "MARIAN: Flexible Interoperability for Federated Digital Libraries." *Research and Advanced Technology for Digital Libraries: [Proc. of] 5th European Conference, ECDL-01 (Darmstadt, Germany: 4-9 Sept. 2001)* Springer, 2001, pp. 173-186.

[12] 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*. Presented at 9th String Processing and Information Retrieval Symposium (SPIRE 2002), Lisbon, Portugal, 2002.

[13] Networked Digital Library of Theses and Dissertations, 2005 http://www.ndltd.org/ (as of April 2005).

[14] Suleman, H. *Open Digital Libraries*. Ph.D. Dissertation. Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, 2002. *http://scholar.lib.vt.edu/theses/available/etd-11222002-155624/*.

[15] Ravindranathan, U., Shen, R., Goncalves, M. A., Fan, W., Fox, E. A., and Flanagan, J. W. *ETANA-DL: A Digital Library for Integrated Handling Of Heterogeneous Archaeological Data*. Presented at Joint Conference on Digital Libraries (JCDL 2004), Tucson, AZ, June 7-11, 2004.

[16] Open Archives Initiative, 2005. *www.openarchives.org* (as of April 2005).

[17] Tell Nimrin - A virtual archaeological site, 2005.

*http://www.cwru.edu/affil/nimrin/* (as of April 2005).
[18] Lahav Research Project, 1999. http://www.cobb.msstate.edu/dig/LRP-1999-01/ (as of April 2005)
[19] Madaba Plains Project – Tall al-'Umayri, 2005. *http://www.wwc.edu/academics/departments/theology/mpp* (as of April, 2005).
[20] ETANA-DL Demo, 2005. http://feathers.dlib.vt.edu:8080/etana/servlet/Start (as of April 2005)
[21] SpotFire, 2005. http://www.spotfire.com/ (as of April 2005)
[22] Chris North, Nathan Conklin, Varun Saini. *Visualization Schemas for Flexible Information Visualization*. In Proceedings of the IEEE Symposium on Information Visualization (InfoVis '02), October 28 - 29, 2002.
[23] North, C., "A User Interface for Coordinating Visualizations based on Relational Schemata: Snap-Together Visualization", University of Maryland Computer Science Dept. Doctoral Dissertation, May 2000.
[24] Haber, Eben M., Ioannidis, Yannis E., Livny, Meron. *Opossum: A flexible schema visualization and Editing Tool.* Conference on Human Factors in Computing Systems. Boston, Massachusetts, United States 1994, pages 321-322.
[25] L. L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. In SIGMOD Conference, 2001.
[26] R. J. Miller, L. M. Haas, and M. Hernandez. *Schema Mapping as Query Discovery*. In Proc. of the Int'l Conf. on Very Large Data Bases (VLDB), pages 77-88, Cairo, Egypt, September 2000.
[27] Altova. *Mapforce, 2005*. http://www.altova.com/products_mapforce.html (as of April 2005)
[28] Microsoft. *BizTalk Mapper. 2005.* http://www.samspublishing.com/articles/article.asp?p=26551&seqNum=5 (as of April 2005)
[29] MapForce vs. BizTalk Mapper, 2004. http://www.osnews.com/story.php?news_id=6809 (as of April 2005)
[30] Card, S. K., Mackinlay, J., and Shneiderman, B., 1999, *Readings in Information Visualization Using Vision to Think*, San Francisco, CA, Morgan Kaufmann.
[31] Johnson, B. and Shneiderman, B., Treemaps: A Space-filling Approach to the Visualization of Hierarchical Information Structures. In Proc. of the 2nd International IEEE Visualization Conference (San Diego, USA, 1991), 284-291.
[32] Robertson, G., Mackinlay J. and Card, S.: Cone Trees: animated 3D visualizations of hierarchical information. Human factors in computing systems conference proceedings on reaching through technology. ACM Press, New Orleans, Louisiana, United States (1991), 189-194.
[33] Boardman, Richard. *Bubble Trees: Visualization of Hierarchical Information Structures*. Conference on Human Factors in Computing Systems, The Hague, Netherlands (2000), pages 315-316.
[34] Lamping, J. and Rao, R., Laying Out and Visualizing Large Trees Using a Hyperbolic Space. In Proceedings of the ACM Symposium on User Interface Software and Technology, 1994, 13-14.

[35] John Lamping, Ramana Rao, and Peter Pirolli. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. Xerox Palo Alto Research Center, 1995.
[36] Shneiderman, B., The Eyes have it: A Task by Data Type Taxonomy. In Proceedings of IEEE Symposium. Visual Languages 96, 1996, 336-343
[37] Cockburn, A. and McKenzie, B.: *An Evaluation of Cone Trees*. In Proceedings of the 2000 British Computer Society Conference on Human Computer Interaction, University of Sunderland, September 2000.
[38] Zylab. *Visualization Module, 2005.* http://www.zylab.com/products_technology/productsheets/Visualization.pdf (as of April 2005)
[39] SourceForge, 2005 http://sourceforge.net/ (as of April 2005)
[40] Document Object Model (DOM), 2005 http://www.w3.org/DOM/ (as of April 2005)
[41] XPath, 1999 http://www.w3.org/TR/xpath (as of April 2005)
[42] Schloen, J.D. Archaeological Data Models and Web Publication Using XML. Computers and the Humanities, 35(2). 123-152, 2001
[43] Locus Definition. http://www.deltasinai.com/records00.htm#(1)%20Trench/Locus (as of April 2005)
[44] CS 5714 Course Notes, spring 2004. http://courses.cs.vt.edu/~cs5714/spring2004/Class%20notes/01-Intro.pdf (as of April 2005)
[45] Usability Evaluation, 2002. http://www.pages.drexel.edu/~zwz22/UsabilityHome.html (as of April 2005)
[46] CS 3724 Course Notes, fall 2003. http://courses.cs.vt.edu/~cs3724/fall2003-mccrickard/lecture-notes/evaluation.pdf (as of April 2005)
[47] Rosson M B, Carroll J M, (2002) Usability engineering: scenario based development of human-computer interaction. Academic Press, San Francisco, CA.

# Appendix A

Requirements for the mapping tool from the client:

a) Local and global schemas need to be visualized in the same screen for mapping.
b) Mappings need to be indicated in a better fashion than lines going across the screen from local to global schema.
c) Global schema editing capabilities required for incremental global schema enrichment while mapping.

Based on the above requirements, the following design possibilities were considered for representing the two schemas:

a) Schemas could be represented as linear trees similar to the representation followed by commercial tools like MapForce and BizTalk Mapper. However a linear representation has drawbacks like excessive scrolling for even moderately sized schemas (30-50 nodes) as the number of nodes that can be visualized at one time is somewhat less (20-30 nodes).
b) Schemas could be represented as TreeMaps, which allow for representing almost 2 orders of magnitude more nodes on the screen. However, locating a particular node would take time. TreeMaps are good for presenting an overview but are not very good for manipulating data.
c) Schemas could be represented as hyperbolic trees, thus allowing for representing more nodes on the screen as compared to a linear representation. Also, searching for a node in the hyperbolic tree is faster than in tree maps.

Hence, the following design iterations were performed while prototyping:

a) Visualizing the local and global schemas as hyperbolic trees and showing the mappings in a separate mapping table

Positives:

1. No scrolling is required unlike commercially available tools.

2. Mapping table avoids the problem of criss-cross lines across the screen.

Negatives:

1. Hyperbolic tree will need to be re-oriented to visualize certain portions of the tree. This is analogous to scrolling in linear representation.

2. User may be lost in the tree as there is no indication as to which node is the root node and which nodes are sub-branch roots and so on.

3. As a result, user may also need more time to locate a node in hyperbolic trees (as the sense of context is lost).

4. Representing more than one local schema may prove to be problematic as number of nodes displayed on the screen in each hyperbolic tree would be reduced to a great extent.

b) Concept of recommendations: recommending nodes for matching (recommendations based on previous mapping history, mapping rules, and name based matching algorithms)

Positives:

1. Locating a node for mapping is no longer a problem as the recommendation engine locates the nodes for you. (Solves negative 3 in design iteration (a).)

2. Builds mapping history, can also be used later as a mappings database for future purposes.

Negatives:

1. Recommendations need not be accurate, so user may end up making a wrong mapping.

2. Sometimes no recommendation may be made by recommendation engine.

c) Coloring nodes to differentiate between recommended, mapped, root, leaf, and non-leaf node.

Positives:

1. Solves negative 2 of design iteration (a) of the user being lost within the tree (as provides local context)

2. Provides another way to visualize mapped nodes apart from the mapping table. (Satisfies requirement (b) for the mapping tool.)

Negatives:

1. 5 colors are optimal; more colors (in the future) may cause confusion.

d) Editing global schema feature:

Positives:

1. Renaming, deleting, and adding sub-tree options provided, so user does not need to use a separate editing tool for editing the schema by hand. (Satisfies requirement (c) mentioned above.)

2. User requires very little knowledge of XML for editing the schema.

Negatives:

1. Adding too many sub-children may result in global schema becoming too large. In spite of the colors and recommendations, the user may not be able to locate some nodes quickly enough.

e) Viewing top-level children only feature: Allows the user to view only the top-level children clearing away the remaining nodes in the hyperbolic tree.

Positives:

1. Even if the global schema is very large, the number of nodes represented on screen will be only top level leaf nodes, making it ideal for adding a new leaf node or a new sub-tree root node. (Resolves negative (1) mentioned in iteration (d) above.)

Negatives:

1. No undo option is present so if the user chooses to visualize using this option, the only way to get the original representation is to re-open the global schema again.

2. As the global schema and the representation are now different the user may get confused as to whether the changes that he incorporates would be done to the actual global schema or to the XML version of the visualized hyperbolic tree with top-level leaf nodes.

f) Viewing only a sub-tree: Allows the user to view only a particular sub-tree and the top-level leaf nodes.

Positives:

1. Even if the global schema is very large, the number of nodes represented on screen will be only top level leaf nodes and the sub-tree, making it ideal for mapping the local schema to this sub-tree.

Negatives:

1. No undo option is present so if the user chooses to visualize using this option; the only way to get the original representation is to re-open the global schema again.

2. As the global schema and the representation are now different the user may get confused as to whether the changes that he incorporates would be done to the actual global schema, or to the XML version of the visualized hyperbolic tree with top-level leaf nodes and the sub-tree.

Resolution of unresolved negatives:

1) ETANA-DL has collections wherein only simple 1-1 schema mapping is involved. So there is no need for having more than one local or global schema on screen at a time. So representing schemas as hyperbolic trees is all right. (resolving negative (4) of design iteration (a)).
2) Number of re-orient actions needs to be measured to see if representing as hyperbolic trees provides any improvement as compared to representing as linear trees with scrolling (resolving negative (1) of design iteration (a)).
3) For design iterations (e) and (f), we need to test if these features are being used at all. If they are, then providing an undo option may be considered. However if users are confused with the representation of the global schema without all the nodes (just one sub-branch / only top-level nodes), then probably these features themselves should not be provided.

# VITA

Ananth Raghavan was born in Mumbai, India, in January 1982. He earned his Bachelor of Engineering in Computer Science and Engineering in August 2003 from Vivekanand Education Society's Institute of Technology (VESIT), Mumbai University, with distinction. He came to the United States in pursuit of a Master of Science degree from Virginia Tech with concentration in Computer Science in the fall of 2003.

During his graduate studies, he worked as a Graduate Teaching Assistant in the Department of Computer Science for 2 semesters. He spent the summer of 2004 as an intern with Microsoft Corp. in Redmond, WA. In addition to Information Retrieval and Digital Libraries, his interests in Computer Science include Graph Algorithms and User Interface Design.

The submission of this work completes his Master of Science requirements. Upon graduation, he will be moving to Redmond, WA to take up a position as Software Design Engineer in Test with the Office-Live Meeting team in Microsoft Corp.