79

Towards an Ontology and Metadata Structure for a Distributed Information System for Coastal Zone Management

by

Pubudu C. Wariyapola

B.A. Physics (1994)
Franklin and Marshall College, Lancaster, PA
M.S. Electrical Engineering (1995)
University of Rochester, Rochester, NY

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Ocean Engineering

at the

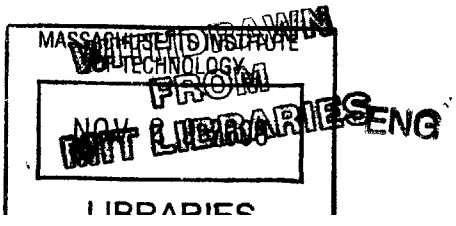
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1999

Author .....
Department of Ocean Engineering
September 7, 1999

Certified by .....
Nicholas M. Patrikalakis, Kawasaki Professor of Engineering
Thesis Supervisor, Department of Ocean Engineering

Accepted by .....
Nicholas M. Patrikalakis, Kawasaki Professor of Engineering
Chairman, Department Committee on Graduate Students



Towards an Ontology and Metadata Structure for a Distributed Information System for Coastal Zone Management

by

Pubudu C. Wariyapola

Submitted to the Department of Ocean Engineering
on September 7, 1999, in partial fulfillment of the
requirements for the degree of
Master of Science in Ocean Engineering

Abstract

The strong new emphasis in understanding and using the oceans, and especially coastal oceans, has created a need for a clearinghouse of information pertaining to the many physical, biological, geological, chemical and other processes that affect the ocean. Therefore we are currently developing such a system, using a distributed software and data architecture to provide access to a wide variety of information resources for Coastal Zone Management. The Poseidon Coastal Zone Management System, as our project is known, will allow distributed users to locate, retrieve, utilize (in simulations and analysis), and visualize information in order to observe ocean processes, conduct scientific research, and develop management and regulatory framework for utilizing and protecting the ocean resources.

The work described in this thesis concentrates on the issues related to efficiently identifying and creating the data needed for a given task. Since scientific data sets often do not contain information about the environment in which the data was obtained, we need a method of distinguishing data based on external information, or metadata. This metadata needs to be standardized in order to facilitate searching. While many metadata standards exist, no one of them is adequate for representing all the information needed for coastal zone management. We have therefore implemented a method that incorporates two existing metadata standards in an expandable object-oriented structure known as the Warwick Framework. Furthermore, since metadata is expensive and tedious to produce, we have developed a web-based software tool that simplifies the process and reduces the storage of redundant information.

While we can search the metadata for the information we need, we still need a common vocabulary, or ontology, to ensure that we can identify data unambiguously. Since no existing vocabulary encapsulates all aspects of the ocean sciences and ocean systems management, we have facilitated the production of such a resource by creating a web-based tool that will allow specialists with the requisite domain knowledge (e.g., oceanographers, acousticians, coastal zone managers, etc.) to populate the ontology independently. The tool handles all the logistical issues of storage, maintenance, and distribution.

Thesis Supervisor: Nicholas M. Patrikalakis, Ph.D.
Title: Kawasaki Professor of Engineering

Acknowledgments

I would like to thank Prof. N. M. Patrikalakis and Dr. W. Cho from the MIT Department of Ocean Engineering, Design Laboratory, Mr. S. L. Abrams from the Harvard University Library, Dr. P. Elisseeff, and Prof. H. Schmidt from the MIT Department of Ocean Engineering, Acoustics Group, Prof. A. R. Robinson from the Harvard University Physical-Interdisciplinary Ocean Science Group, and Dr. K. Streitlien from the MIT Sea Grant College Program, Autonomous Underwater Vehicles Laboratory for contributions to the conceptual and physical development of the Poseidon System and various parts of the work described in this thesis. I also appreciate useful discussions with Prof. C. Houstis, Prof. C. N. Nikolaou, Dr. S. Lalis, Mr. M. Marazakis, and Mr. A. Sidiropoulos of the University of Crete, Department of Computer Science, and ICS-FORTH, Crete, Greece, concerning Poseidon and the related Thetis project at their Institute.

Funding for this work was obtained in part from the U.S. Department of Commerce (NOAA, Sea Grant) under grant NA86RG0074, the U.S. National Ocean Partnership Program via ONR grant N00014-97-1-1018, and the MIT Department of Ocean Engineering. NATO provided travel funds for exchanges between the Poseidon and Thetis groups under grant number CRG971523.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Project Overview

With the advent of new sensors, storage technologies, and widespread access to the Internet, the potential exists for a new era of ocean science investigation where scientists, students, resource managers, and government officials have frictionless access to oceanographic data, simulation results, and software. The objective of the Poseidon project is to develop a network-based architecture to locate, retrieve, analyze and visualize distributed heterogeneous data for scientific exploration and administration of the ocean environment [46, 43, 44]. Poseidon will include distributed analysis, modeling, simulation and visualization software, and the capability to graphically create and execute complex workflows constructed from distributed data and software resources. (Workflows are represented as dependency graphs describing the flow of data between various software processing elements, such as simulation, analysis, visualization, etc.)

The Poseidon architecture assumes that the scientists who collect or create data will remain responsible for that data's storage, maintenance, and accessibility by some on-line access mechanism. Similarly, the available software tools will be accessed in a service-oriented mode [8], running on remote hosts. The organizations and scientists that created these tools retain full ownership and control of the software.

The Poseidon front-end consists of an executive system deployed as a Java applet [27] downloaded to a standard Java-enabled web browser (see Figure 1-1). This executive system supports user authentication, searching for and retrieving distributed data and software resources, creating workflows, and supervising their execution. The Poseidon server interacts

with distributed resources using the CORBA (Common Object Request Broker Architecture) protocol [23]. CORBA was selected because of its wide acceptance and its proven ability to facilitate cross-language, cross-platform integration. All distributed resources in the Poseidon system are encapsulated as CORBA-compliant objects. The creation of CORBA wrappers for legacy systems is performed in collaboration with the owners of those resources.

1.2 Overview of Related Activities

In order to search and locate data relevant to a particular task, we need an accurate description of that data. Many of the resources used in the Poseidon system are scientific data and simulation results that do not contain within themselves adequate information for accurate identification. Thus, we need a method of capturing and storing this information (or metadata, "data about data") *external* to the data itself. We conceptualize metadata as a set of descriptors that uniquely identify a specific entity, possibly using terminology drawn from an ontology (see Chapter 3). Metadata is necessary to capture the semantic content of data sets and software programs. An indexed metadata repository can be used to enable efficient resource discovery in the context of an information system utilizing distributed software and data resources.

Perhaps the most familiar use of metadata is bibliographic information (e.g., author, title, publisher, etc.), which has long been used in library and information retrieval systems [1]. The application of similar bibliographic metadata for web-based resources led to the creation of the Dublin Core metadata standard [3, 17, 56, 57]. The Dublin Core was subsequently extended to include the Nordic Classification system as part of the Nordic Metadata Project [24]. The DIENST system [29] used by the NCSTRL (Networked Computer Science Technical Reference Library) project [14, 34] also uses similar bibliographic metadata based on RFC 1807 [31]. The evolving Resource Description Framework (RDF) [32, 38] makes use of the web-based hyperlink capabilities of XML (Extensible Markup Language) [5] to provide a flexible model for defining the appropriate metadata for resources in a particular problem domain.

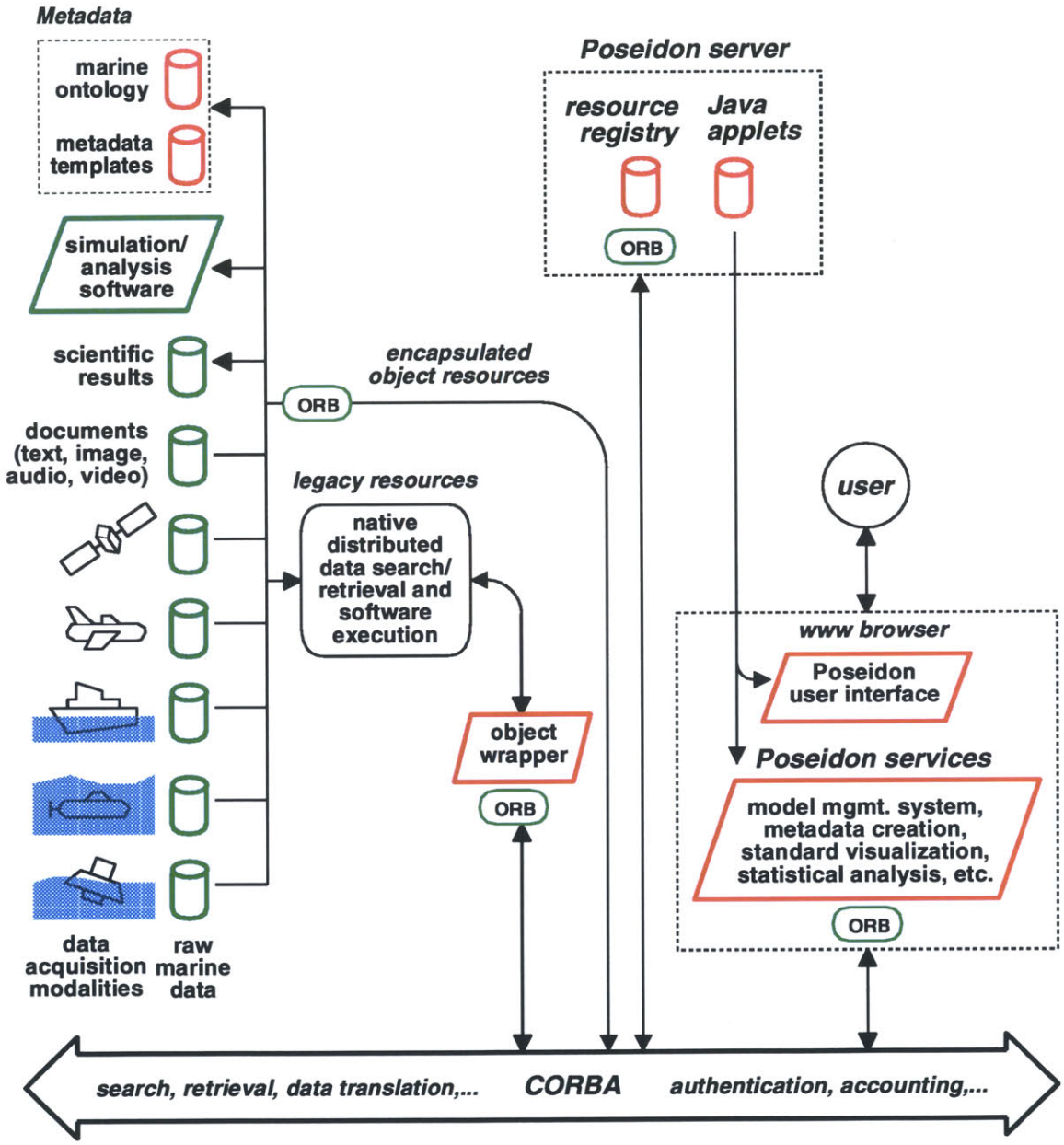


Figure 1-1: Poseidon architecture.

To provide a uniform standard for describing geospatially referenced data, the U.S. government established the Federal Geographic Data Committee (FGDC) under the aegis of the National Spatial Data Infrastructure (NSDI) initiative [11], resulting in the Content Standard for Digital Geospatial Metadata (CSDGM) [19]. CSDGM defines over 250 individual properties organized into seven categories: (1) Identification; (2) Data Quality; (3) Spatial Data Organization; (4) Spatial Reference; (5) Entity and Attribute; (6) Distribution; and (7) Metadata Reference. The CSDGM is used as the basis for interactive web-based search and retrieval for the Defense Modeling and Simulation Office's Master Environmental (MEL) service [35, 36]. The U.S. Geological Survey's Biological Resources Division (BRD) has created a Biological Data Profile of the CSDGM as part of the National Biological Information Infrastructure (NBII) Metadata Standard [4]. The Biological Profile recently completed its public review stage.

Additional geophysical-based metadata efforts are also underway, including NASA's Global Change Master Directory [22], a MEL-like service that uses the CSDGM-compatible Directory Interchange Format (DIF); the NASA-sponsored EOSDIS project [16] for disseminating satellite remote sensing data; and the Unidata project [54] for disseminating meteorological data. Geophysical metadata will be central to the Thetis project [25, 26, 40, 52] underway at ICS-FORTH, Crete, Greece and several other European research institutes (see <http://www.ics.forth.gr/pleiades/THETIS/thetis.html>). The Thetis system is intended to interconnect distributed collections of heterogeneous scientific data repositories, geographic information systems, simulation codes, and visualization tools via the Internet and the WWW. These data repositories are used for the management of coastal zones of the Mediterranean Sea.

There are several ongoing research efforts that have developed metadata creation tools for use in data warehousing and distribution [45]. The XTME metadata editor, MP metadata parser, and CNS metadata pre-processor [51], developed by the U.S. Geological Survey, are stand-alone tools that can be installed on a variety of operating platforms to allow the user to create CSDGM metadata, translate from unformatted to CSDGM format, and verify metadata conformance to the CSDGM standard. The BIG BIC Metadata Form [37], created by the Texas/Mexico Borderlands Information Center (BIC), allows users to create and submit metadata for their resources to the BIC data clearinghouse using an HTML form-based web interface implementing CSDGM. The Nordic Metadata Creator [24] uses a

similar web-based approach for creating Dublin Core metadata.

Several other research projects related to the aims of Poseidon are of interest. The ISAAC Internet Scout project [49] at the University of Wisconsin-Madison has developed tools for the transparent query of, access to, and visualization of information in on-line data repositories using bibliographic metadata. The TSIMMIS project [21, 53] at Stanford University is developing a system to access distributed data sources. Both the ISAAC and TSIMMIS projects, however, are concerned primarily with document-like resources. MEL provides interactive web-based search and retrieval of geospatial data using HTML Form and Java-based query mechanisms. MEL can disseminate complex numerical data, but it only supports relatively simple query and report functions. Poseidon, on the other hand, aims to discover distributed resources with sufficient accuracy to allow automated pipelining through complex workflows.

1.3 Problem Statement and Thesis Outline

In order to create a distributed system of data and software, capable of locating, extracting and pipelining data trough complex workflows, we need metadata to accurately, efficiently and uniquely identify available resources. In order to allow these identifiers to be used correctly, they have to be standardized. While many such standards exist and are widely used, they are not sufficient or optimal for the applications we foresee. Nonetheless, we neither want to, nor have the resources to, duplicate the work that has been done in creating these standards and making them widely accepted. Therefore we need an architecture that will allow us to combine existing standards to create a metadata structure that can be used to describe resources within Poseidon. This structure must be expandable to allow future inclusion of new and/or popular standards. While we can use these metadata standards to describe resources, we still need a common language or ontology to ensure consistency in creating and using metadata.

This thesis describes the issues related to creating a metadata standard for the Poseidon system, and our approach to developing an ontology to be used in conjunction with the metadata. Chapter 2 provides an analysis of the considerations in creating a metadata standard, and takes the reader through the structure, architecture, and implementation of the Poseidon Metadata Standard. It also presents the associated development of the

Poseidon Metadata Creator to allow faster and more efficient creation of metadata. Chapter 3 outlines the approach we have taken to creating an ontology, and describes in detail the architecture and the implementation of a web-based software tool we developed to achieve this end. The last chapter summarizes the research associated with this thesis and presents our conclusions on the status and contributions of this work as well as recommendations for further research. The appendices contain excerpts from the code that has been developed in the work described here and the associated user guides.

Chapter 2

Metadata

2.1 Overview

Metadata, or data about data, provides a mechanism for describing and identifying resources efficiently and accurately. Instead of reading through an entire document, its abstract allows the reader to get an idea of what is contained within. Similar to an abstract, metadata too attempts to summarize the information in the resource, but it goes further by attempting to create the context in which the resource exists (i.e. part of a larger collection of information). While metadata cannot realistically capture all the information contained in a document (this would defeat the purpose of having metadata), it attempts to record all the information that is required to uniquely identify and locate the document.

While the use of metadata improves efficiency and accuracy in locating documents, it is indispensable in describing scientific data sets, many of which contain no information about their contents or the context in which they were created. Metadata therefore must be used in order to give meaning to the mostly numeric data that is stored in these vast sets and allow their retrieval and processing, by any other than the scientist that created them. Since the aim of the Poseidon system is to allow a variety of users (many of whom will not have an in depth knowledge of the data) to access and use the data, metadata becomes an absolutely necessary tool.

2.2 Metadata Standards

While it would be possible to allow each data provider to create the information that he or she feels is relevant to the data, the use of heterogeneous metadata formats decreases the efficiency and effectiveness of searching. To prevent this, we need a well-defined metadata standard that can be followed when creating metadata and can subsequently be used to search for the data accurately and efficiently. On the basis of our analysis of existing metadata standards, no one of them satisfies the needs of the Poseidon system. Poseidon needs to support documents as well as scientific data, while providing sufficient accuracy to allow automated construction of workflows, ensuring compatibility and consistency between sequential operations. Thus, none of the document-oriented standards (Dublin Core, RFC 1807, RDF, etc.) is adequate for our needs. The FGDC standard for geospatial metadata, while being quite effective in classifying information relating to physical phenomena, is not suited to handle the biological aspects of many ocean processes and is too cumbersome for documents. The FGDC Biological Data Profile is only an extension of the CSDGM, and is only relevant in that context.

Even though we did not find any one standard that was sufficient for our needs, we still decided to adopt existing standards to avoid duplicating the extensive amount of work that has already gone into their development. We also wanted a format that will be widely accepted and implemented so that we can use data sets that are available on-line (such as MEL) without requiring any additional work from the producers of the data (to create new sets of metadata). We have therefore selected the FGDC standard for geospatial metadata (along with its biological profile) as part of the overall Poseidon metadata standard. Since this metadata set is quite extensive, we felt that it are not quite appropriate for describing text documents, which require only a small subset of the descriptive information (compared with scientific data). Since the Dublin Core metadata standard was developed expressly for the purpose of describing (text) documents, we have decided to include it in our overall metadata set.

2.3 Metadata Architecture

We have implemented these two distinct standards (Dublin Core and FGDC-CSDGM with its Biological Profile extensions) using the expandable container structure developed in the

Warwick framework [28]. (The Warwick Framework was developed to provide a mechanism to cope with the proliferation of incompatible metadata standards. It defines a hierarchical container model that can encapsulate multiple metadata representations for the same underlying data object.) Using this conceptual format we have developed a metadata container structure for the Poseidon system (see Figure 2-1). Similar to the Warwick framework, each metadata standard is treated as a container (object) within which are stored all of its individual elements (field values). All of these containers (standards) are placed in a larger container defined as the Poseidon Metadata Standard. While the complete metadata set consists of all the elements in all the standards included in the container, only the relevant sections need to be completed (other sections can be linked to a NULL value). This architecture is naturally extensible to allow the inclusion of new standards without having to update existing metadata sets, since any missing values are assumed to be NULL.

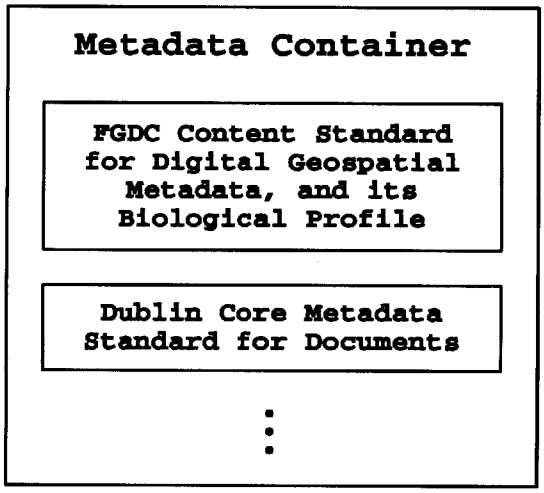


Figure 2-1: Metadata standard architecture.

Within the Poseidon architecture, all resources are treated as objects. Each data or software object in the system has a corresponding metadata object. Conceptually, each metadata object consists of multiple child objects. Using the Warwick framework, currently the top-level children are the two metadata standards described above. This can be recursively extended until each individual metadata element (i.e., a single field in a metadata standard) is treated as an object (see Figure 2-2). Each of these objects will be stored in an object database which can keep track of the semantic and hierarchical links between the objects to allow the entire metadata set associated with a data or software object to be retrieved and visualized as a single unit.

This will also allow the system to store only the unique objects (fields) of any new metadata set, and store the non-unique information as links to existing objects. Since the differences in metadata among various data sets obtained in one experiment–or a set of related experiments–are generally small, this metadata model allows significant reduction in storage space required. It also simplifies searching since only the unique elements need to be indexed and searched.

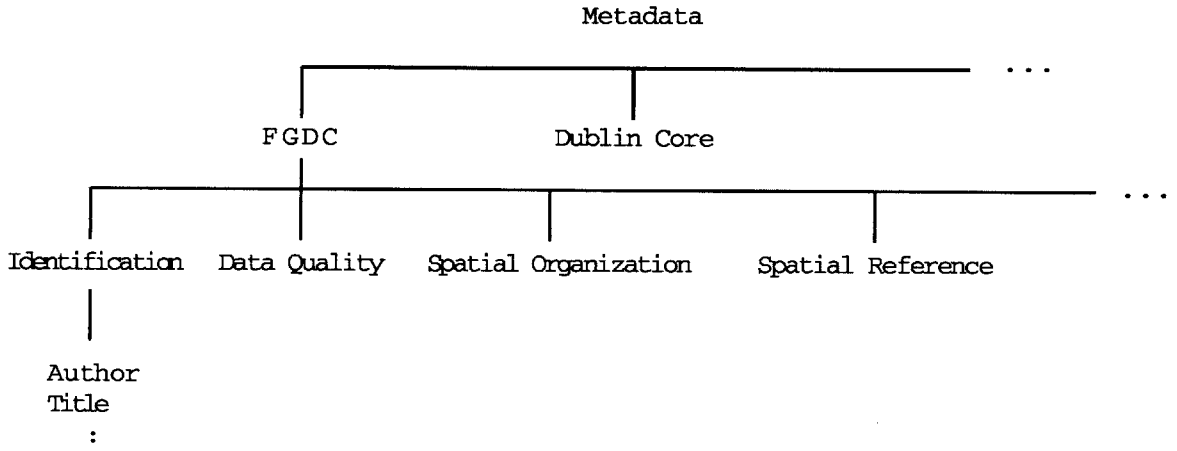


Figure 2-2: Conceptual object-oriented metadata framework.

2.4 Implementation

We have implemented a part of the above schema to verify its feasibility. Our system uses a Unix file system directory structure corresponding to the metadata framework to store the objects. The objects are currently stored as flat files. Since we are not using an object-oriented database, we have only implemented the top three levels of the tree: the top-level abstract metadata object, the second level metadata standard-specific object, and the third level containing all metadata element values. Note that the Dublin Core standard is implemented in two levels. Currently the higher level objects are implemented as HTML documents with links to the associated lower level objects. The user can browse the metadata by following these links. The lowest level objects contain all the information of the metadata set and are implemented as HTML forms with the element tag (metadata field name) and the field data implemented as name-value pairs.

The metadata are stored in a directory structure that mirrors the object hierarchy in the Poseidon Server (see Figure 2-3). All the records are stored within the "Metadata"

directory, with the metadata sets listed in the "metadata.html" file (see Appendix A.2.4). This listing provides a link to the files in the "MetadataFiles" directory (see Appendix A.2.5). These files in turn link to the information contained in the "DublinCore" (see Appendix A.2.6) and "FGDC" directories and their respective sub-directories. The files in the "MetadataFiles" directory also contain the JavaScript functions needed to display and copy the information in the current metadata files to the metadata form (see Section 2.8, Figure 2-4 and the left frame in Figure 2-6), in order to allow faster, more efficient creation of new metadata sets.

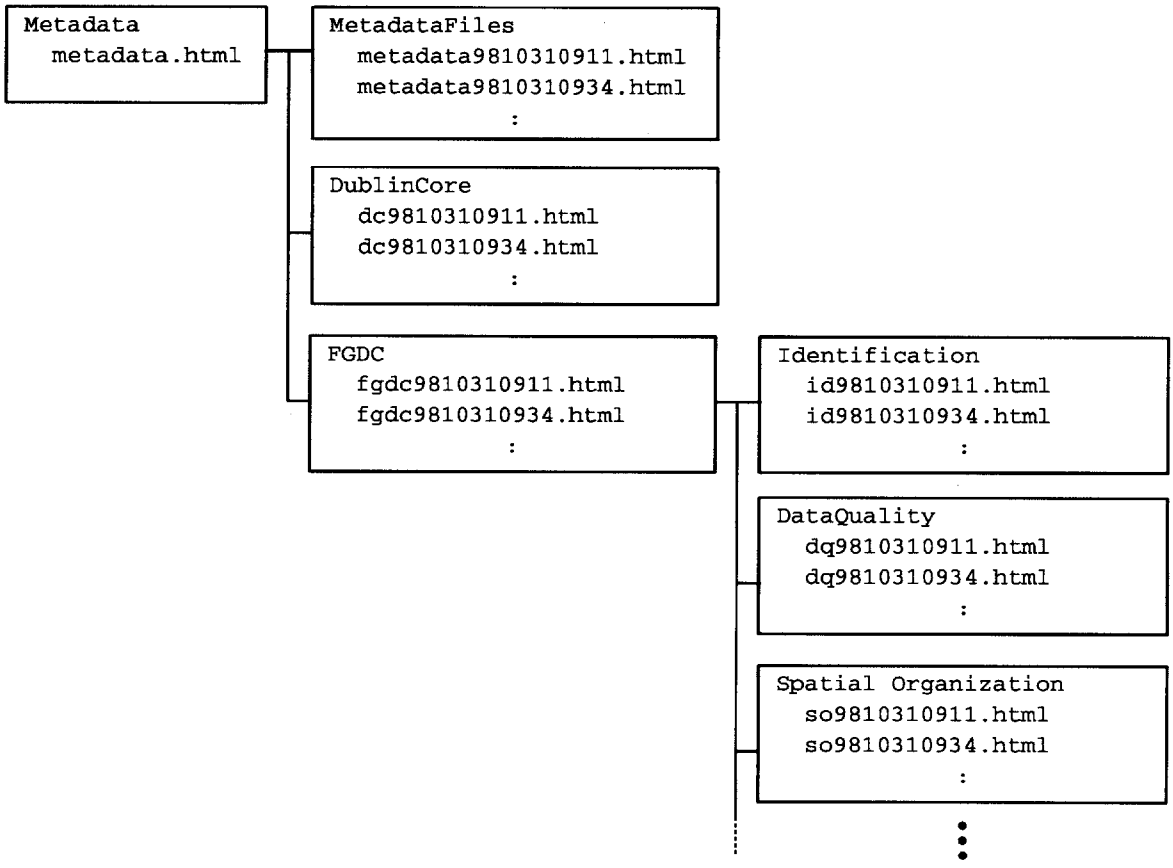


Figure 2-3: Directory based metadata storage structure.

2.5 Metadata Server Architecture

When the system is fully implemented, the metadata will be stored and accessed using a regional client/server architecture. All metadata will be duplicated at each regional server, allowing for scalability [50], fault-tolerance [15, 47, 48], and load-balancing [18, 33]

within the system. Individual metadata objects are indexed at the time they are created; the full metadata collection is indexed daily using an automatic indexer. This allows the system to access all data as metadata is created, and to stay current by eliminating obsolete information.

All searching will be performed on the indices. Once a match has been found, the entire metadata record and its associated data file can be accessed. While the full range of capabilities of the search tool have not yet been defined, we expect that all the metadata fields will be searchable using exact text matching and numerical range matching. We are also investigating the possible use of fuzzy query capabilities. An extension of the DIENST system [29] may be used for this purpose.

2.6 Metadata Creation

In order for metadata to be useful, it needs to be complete and accurate. Since much of the information needed for the metadata is only available at data creation or acquisition time, metadata should ideally be created contemporaneously with the data. The quality of the metadata is enhanced if it is created by an individual with a significant level of knowledge about the data, its acquisition environment, and process; most likely, this will be the scientist responsible for the experiment.

2.7 Metadata Creation Tools

While the level of knowledge needed to create metadata is quite high, so too is the volume of metadata needed to describe a data set accurately. These requirements place a very high time and resource cost on metadata creation. Therefore, we obviously need some tools that can aid in this process. Ideally, we want a software utility that resides within the data acquisition system and automatically creates metadata for the data. In the Poseidon project, we are continuing ongoing collaboration with our partners to implement such a solution on a case by case basis. We are, however, able to create a web-based editor that can be accessed using a standard web browser and that can be used to create metadata efficiently and accurately.

2.8 "Poseidon Metadata Creator"

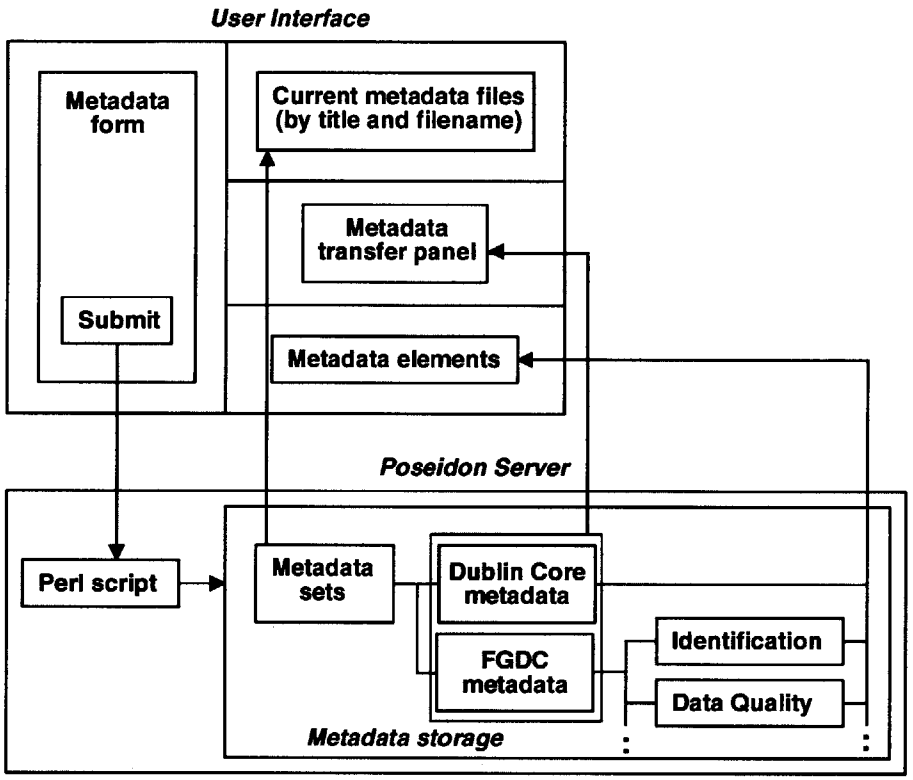


Figure 2-4: Poseidon Metadata Creator design architecture.

Figure 2-4 shows the current architecture of the Poseidon Metadata Creator. The user interface consists of 4 frames. The left half of the page (form) contains a standard HTML form used to accept user-input metadata, which is then submitted to the Poseidon server (see Figure 2-6).

A Perl script residing on the server (see Appendix A.3) parses and stores the metadata as objects on the server. The script creates a unique identifier for the metadata set based on the metadata creation date and the process identification number for the current instantiation of the script (see Figure 2-5). This identifier is written as a link to the master list of metadata sets available in the system (metadata.html). The script subsequently uses this identifier to name the files that it creates to store the metadata field values. As described above the higher level objects are written as HTML links while the lowest level objects are implemented as HTML forms with name-value pairs for the metadata field name and its value.

This information is then returned to the user interface automatically by writing a

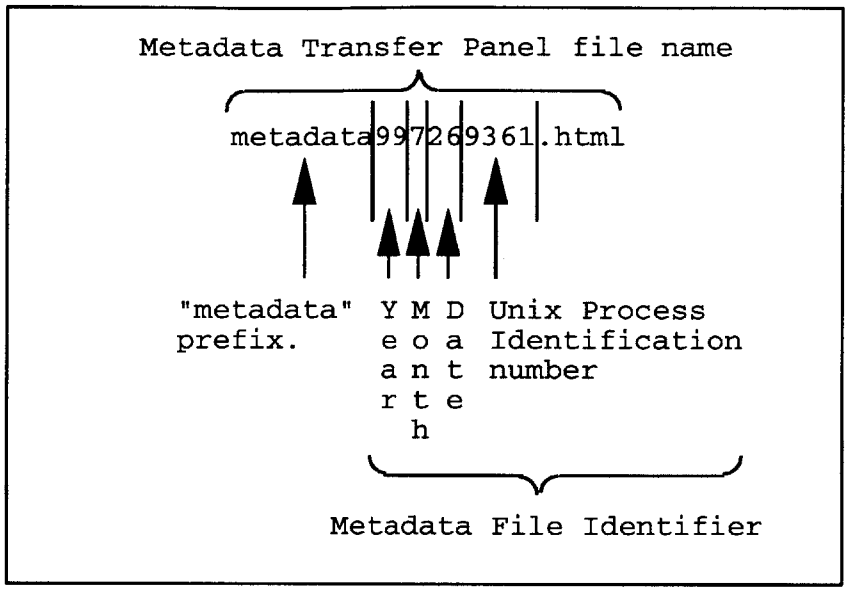


Figure 2-5: Metadata file identifier structure.

JavaScript function to the frame used by the metadata form. The JavaScript function reloads the page (all the frames), which by that point has been updated by the Perl script to include the latest information submitted to the server.

The design of the metadata creator allows the user to browse existing metadata sets and copy any appropriate information to create new metadata. The top right frame (list) shows the list of available metadata sets. These metadata sets can be browsed and displayed in the middle right (transfer panel) and bottom right (display) frames. The transfer panel shows the structure of the metadata set (as shown in Figure 2-4). The display frame contains the name-value pairs that are the metadata elements of the specific sub-set of the standard selected using the transfer panel. The transfer panel also allows the user to automatically copy entire sections of metadata from existing files to the metadata form. Since metadata for a series of data sets produced by a single user, or during a single experiment, differ only slightly, this feature allows significant time savings for creating metadata.

Currently, existing metadata objects are displayed as a scrollable list. As the system continues to grow in size, we propose to change this to a hierarchical list organized on the basis of a user defined field (e.g., owner, location, keywords, etc.). We will also add a search feature to allow users to find metadata sets without browsing the entire list. In the current implementation, the Metadata Creator supports all the elements of the Dublin Core and FGDC-CSDGM, except for recursive calls to itself from within an element description, and

multiple iterations of the same element. While we suggest using the "|" key to delimit multiple entries within a single field, we will need to modify the tool to allow true recursion and multiple instantiation to exactly conform to the standards used. We will also need to add the elements of the Biological Profile of the FGDC-CSDGM once they have been finalized by the National Biological Information Infrastructure (NBII) working group.

While a number of tools implement some relatively well known metadata standard, and simplify the creation of metadata by presenting the user with a template that matches that standard (e.g., XTME, BIG BIC, Nordic Metadata Creator, etc.), most of them require that the user individually input all relevant information by hand, although some do have more automated features, such as the ability to extract a small subset of the information from the data itself or the ability of the user to pre-define some fields common for all of metadata, e.g., ESRI's Document.aml [2]. In contrast, the Poseidon metadata creator (which is partially based on the BIG BIC Metadata Form) adds the capability of browsing all metadata files stored in the system as well as the ability to copy any information already available. This significantly reduces the cost of developing metadata for multiple data sets with similar metadata properties.

Two example metadata sets have been created using the metadata creator to illustrate its. These are listed as available metadata files on the Poseidon Metadata Creator at <http://czms.mit.edu/poseidon/software/MetadataCreator>.

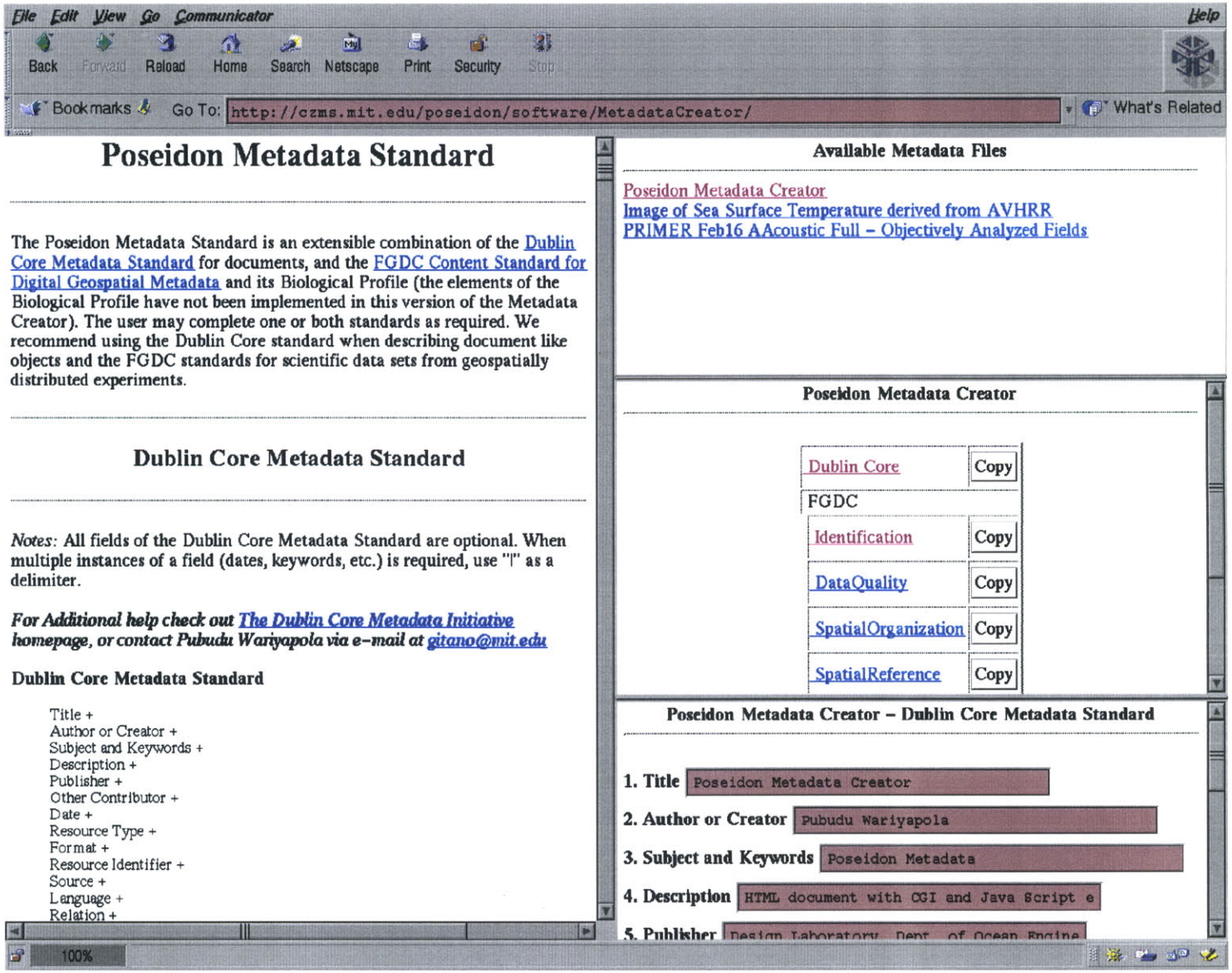


Figure 2-6: Poseidon Metadata Creator user interface.

Chapter 3

Ontology

3.1 Motivation

An ontology is a formal model of the entirety of abstract characteristics and properties that are applicable to all possible specific instances of entities existing within a problem domain [6, 7]. We conceptualize an ontology as a controlled formal vocabulary. While metadata allows us to search for data using auxiliary information, in order for this search to yield accurate results we need to have a consistent language that can be used by both the creators and the users of the metadata. If we can find or create such a language, it can be implemented as a controlled vocabulary within the metadata creator and the metadata search tool to allow consistency in defining and searching. Thus, a common ontology ensures consistency of meaning between information providers and information consumers.

3.2 Marine Ontology

Our research has not provided us with an ontology for the marine sciences and coastal zone management that is broad enough to encapsulate all, or even most, of the information found in the data sets that we envision. In this absence, we are creating just such a resource. Creating an ontology is a time-consuming task that requires a broad range of domain expertise, and significant administrative and editorial overhead. Since our research group does not by itself have the resources necessary to complete the task, we have taken a different approach: instead of directly creating the ontology, we have developed a web-based graphical tool that can be used to populate a pre-defined vocabulary structure. This tool

allows domain specialists to access the ontology from distributed sites and to populate the parts relevant to their specialization. These specialists are not burdened by the need to understand the entire ontology or its administration, and we do not bear the full responsibility for the scientific content or editorial process. The Ontolingua project [20] at Stanford University, uses a similar web based distributed approach to developing and maintaining multiple ontologies, but its user interface is purely text and hypertext based. We are not aware of any other similar research projects in this field.

3.3 Poseidon Ontology Structure

The Poseidon ontology has been implemented as a collection of objects. Each term, or element, in the vocabulary is treated as an object with the properties listed in Table 3.1.

Name:	[*Text*]
Definition:	[*Text*]
Reference:	[*Text*]
Date:	[*Date*]
Parent:	[*Text*]
Creator History:	[*Text Array*]
Definition History:	[*Text Array*]
Reference History:	[*Text Array*]
Date History:	[*Date Array*]

Table 3.1: Ontology element properties

The name and definition fields identify and describe the element, while the reference field is used to provide the source of this information (scientist creating the definition; book, or internet resource from which the definition was extracted, etc.). Since terms may have different meanings in different domains, the concatenation of an element's lineage (the sequence of parents starting at the top level - see Section 3.4) and its name allows the element to be uniquely identified.

The last four properties (creator, definition, reference, and date histories) are used to maintain a history of the element's creation and modification. This is achieved by storing the new definition and reference values along with the name of the user and the date of change. Since the tool allows multiple users to access and change the elements of the ontology, storing the history is necessary to prevent problems due to loss of data and to

resolve conflicts between users (it allows us to save any information that may be lost when a user changes an element definition, or accidentally deletes part of an element). This feature also provides a mechanism with which the domain scientists can see the evolution of the element's definition, and collaborate in expanding it while referring back to older versions for consistency and accountability. Ultimately if two or more scientists disagree on a definition, the differences will be clearly displayed in the element history and will provide a focal point for discussion among them to resolve their differences (the scientists will have to collaborate on resolving these differences since we do not provide editorial mediation). The history mechanism will also afford us a degree of protection and robustness by safeguarding against loss of information due to vandalism or other means.

3.4 Ontology Storage

In our prototype implementation, the objects are stored in flat files in directories corresponding to the vocabulary structure illustrated in Figure 3-1. The depth of this hierarchy is currently limited to three levels to enable a broad structure that can be navigated easily (in future releases, we will increase the depth in order to accommodate growth and enhance visual browsing).

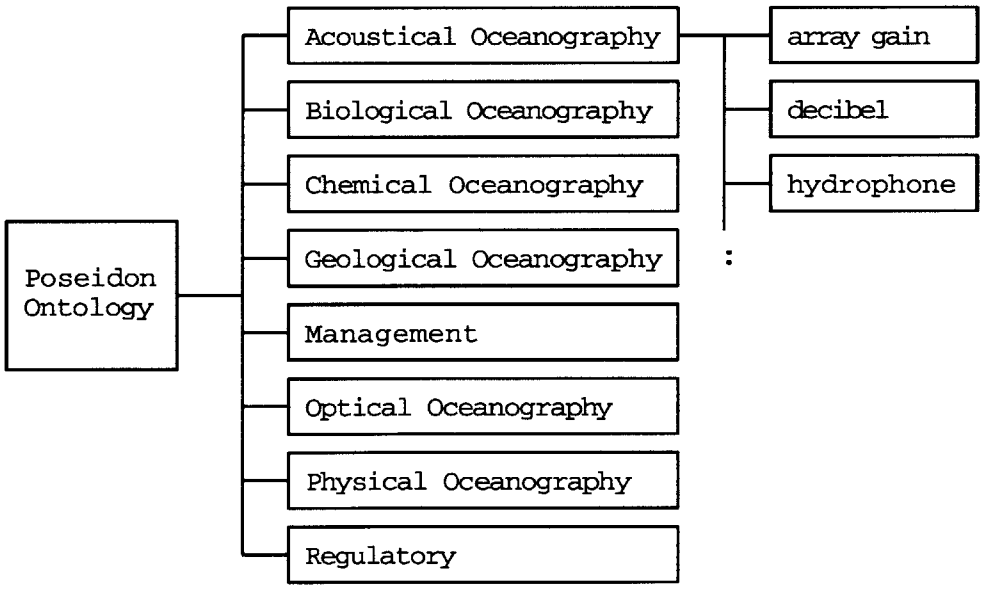


Figure 3-1: Poseidon Ontology structure.

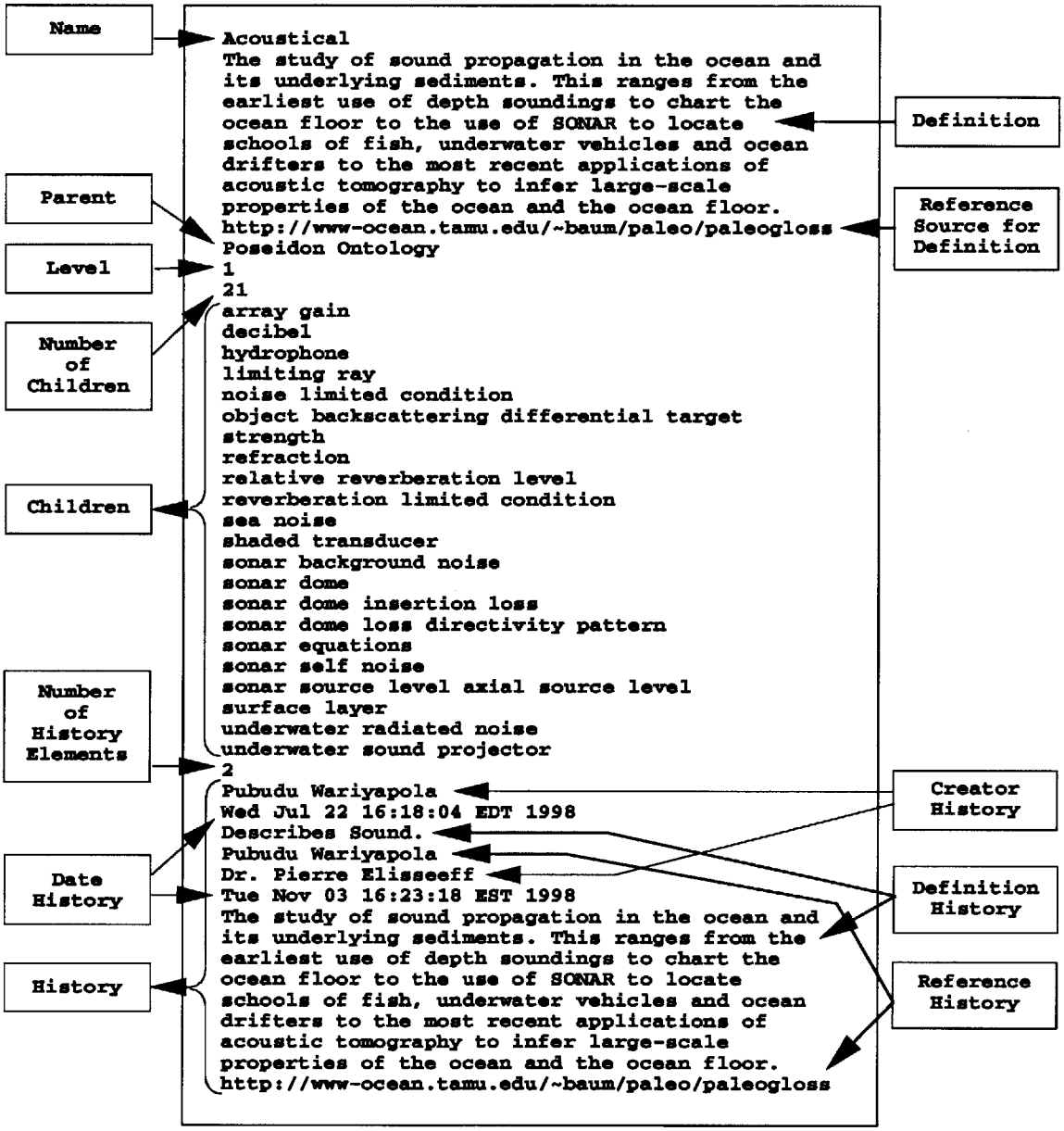


Figure 3-2: Storage format of an ontology element.

Each object (term in the ontology) will be written as a separate text file (see Figure 3-2 and Appendix B.5). The properties of the object (name, definition, etc.) will be recorded as a new line of text. Each file will also contain three numeric values defined as "element level," "number of children," and "number of history elements" that will contain necessary implementation information. The "element level" will define the depth of the element in the ontology (0, 1, or 2 with level 0 referring to the top level element). The "number of children" will contain information on how many terms are in the category defined by the element (this value will be zero for all level 2 elements). The "number of history elements" will show the number of times the element has been modified.

3.5 Ontology Creator

The Poseidon Ontology Creator consists of a Java client applet downloadable to any standard Java-enabled web browser and a Java application daemon running on the Poseidon server. Communication between these two components is handled via a socket connection (see Figure 3-3).

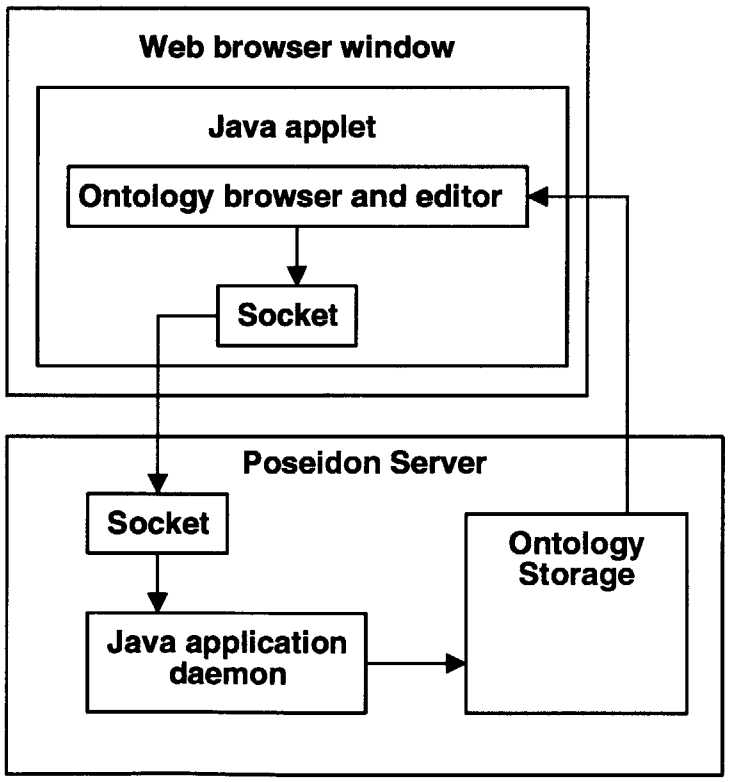


Figure 3-3: Poseidon Ontology Creator architecture.

While the tool allows all users to access and view the existing ontology, editing functionality is reserved for authorized users ("specialists"). Once authenticated, "specialists" can use the Ontology Creator (see Figure 3-4) to add elements to, and edit elements in, the vocabulary. The tool stores the new and modified elements on the Poseidon server and displays them via the Java user interface. Only authenticated users with "editor" privileges can delete an element.

3.6 Ontology Creator Architecture

The two parts of the Ontology Creator (User interface and Server) are written as two independent systems. They communicate through an asynchronous bi-directional Java socket connection [9, 10, 39]. The Ontology Server reads the ontology data stored in the flat file structure described above (see Section 3.4), opens a socket, and waits for a request from a client (see Appendix B.3). All the information must pass through the Ontology Server since the Java language currently does not allow applets to write directly to files. In order for the client (Ontology GUI) to be able to connect to the Server it must know the host name and the port number for the server socket. This information is included in the user interface code.

The user interface consists of the applet containing the Ontology Browser (see Figure 3-4) and the code required to log-in to the system, communicate with the server, and modify the existing ontology (both the local copy and the remote master copy on the server) (see Appendix B.2). Once the interface is opened in a Java enabled browser, it creates the display and requests the user to log-in to the system. Once the user successfully logs in, the interface enables a series of functions reserved for authorized users (add, edit and delete elements). If the user does not log-in (or log-in fails) the system allows browsing of the existing ontology, but does not permit its modification.

The user can browse the ontology by graphically selecting the sections and elements of interest. If the user wants to add an element the system creates a new element template and allows the user to input the name, definition and reference source for the element (the rest of the properties are automatically created based on the parent (category) of the element being created). The user must select the category to which the element must belong before requesting to add an element. Once the user has entered the necessary information,

the system checks its validity (checks for control characters, and ensures that the name and definition fields are filled), creates an element and links it to the existing ontology by modifying the child information contained within its parent. The user interface then submits a request through the socket to the server, requesting the information be updated at the server.

Similarly the user can edit an element by selecting it and requesting the desired function (if authorized). The interface then allows the user to change the definition and reference fields (but not the name), and again verifies the validity of the information before processing the request. The edit function is performed by copying the data in the current element to temporary locations, deleting it, and creating a new element that incorporates the modified definition and reference fields along with the properties of the original element. This process is repeated with the remote (server) copy of the ontology.

Finally the "editors" of the ontology can delete an element by selecting it and requesting its deletion. The system removes the local and remote copies of the element and changes the parent element to remove references to it.

Whenever the GUI requests an operation from the server, it waits for an acknowledgment. Once a request for modification is received at the server, it changes (adds or deletes) the information in its transient memory and writes the changes to disk. Once these operations have been performed successfully, the server returns an acknowledgment with an "Element Deleted" or "Element Added" message.

Part of the Poseidon Ontology has been populated using the Ontology Creator tool. These elements can be browsed by accessing the ontology through the Ontology Creator tool at <http://czms.mit.edu/poseidon/software/OntologyCreator>.

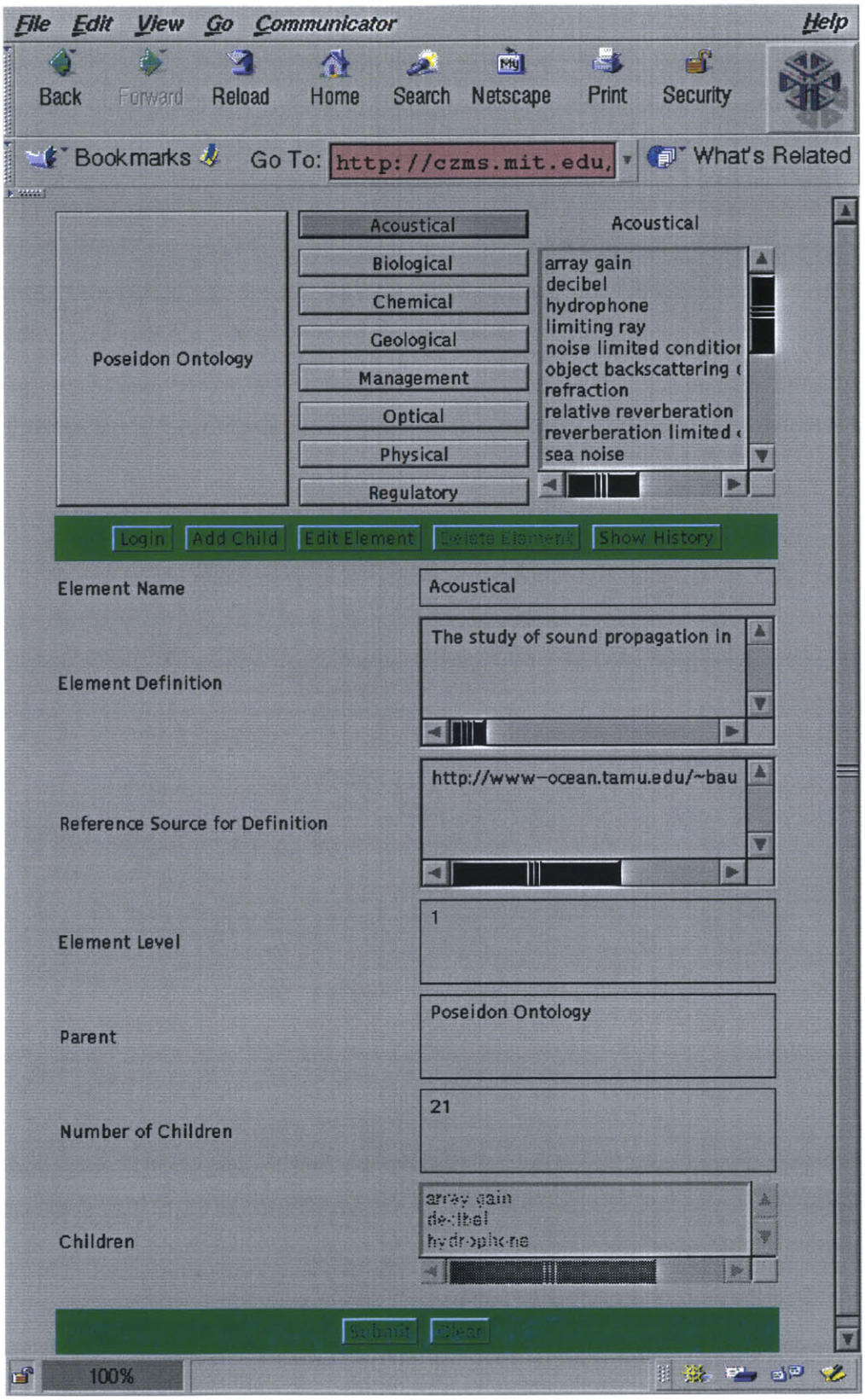


Figure 3-4: Poseidon Ontology Creator user interface.

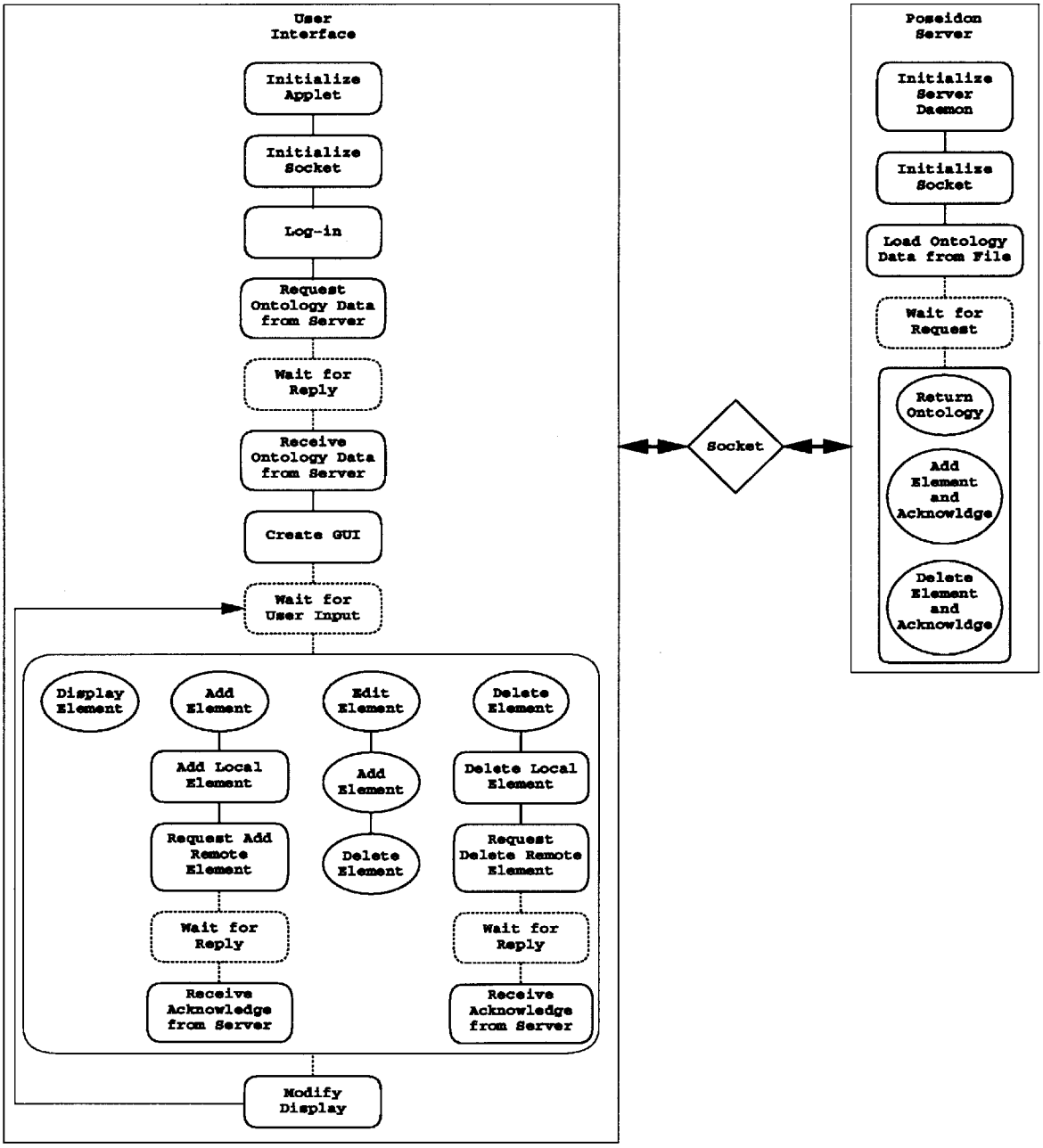


Figure 3-5: Poseidon Ontology Creator program flowchart.

Chapter 4

Conclusions and Recommendations

We have implemented a web-based software tool to facilitate the creation of an ontology for the marine sciences and coastal zone management by appropriate domain specialists. The ontology has a pre-defined, but expandable multi-level hierarchical structure. It has been implemented for three levels and partially populated using the creation tool. We have also developed a conceptual framework for the metadata used by the Poseidon distributed coastal zone management system. This includes provision for two distinct metadata standards combined using the Warwick Framework, their implementation using object-oriented concepts, and a scalable distributed architecture for metadata creation, storage, and searching. We have developed a metadata creation tool that allows distributed users to produce a metadata set conforming to the Poseidon Metadata Structure using a web front-end. We plan to improve the tool in the future by adding user-specified hierarchical browsing and a search mechanism to allow users to find metadata directly on the basis of keyword or value matching, without the necessity of manual browsing.

We have not presented the structure of the Poseidon Metadata Server in much detail here. While we anticipate it to be implemented within a regional client/server architecture, much work needs to be done before we can complete the system design. There are many issues that need to be resolved in this context. Will the metadata be replicated at each site? How will we guarantee the accuracy of the metadata during transfer and storage? How will the metadata be indexed? How will it be searched? These are only a small subset of the work that needs to be done before the metadata server can be implemented.

While we have attempted to build tools that will be able to perform the expected

functions, we have not attempted to guarantee their robustness in high traffic situations. We have identified some possible failure conditions and have presented them in the appendices (see Appendices A.1.3 and B.1.3). In order for the tools developed as a part of this thesis to be used in commercial or high traffic applications, these possible problems have to be rectified. Further work will be needed to this end.

The metadata tool is most relevant to the creation of metadata for small sets of experimental data or for legacy data sets. It may not be appropriate for metadata for data sets produced by real-time automated sensors. Rather than use an interactive system, the metadata for such data should be produced in as automated a fashion as is possible by the data acquisition system itself.

To date, we have focused our efforts on metadata for data. Construction of complex oceanographic workflows will require resource discovery of both data *and* software. The problem of appropriate metadata for programs remains an open research topic [41, 42]. One promising approach is the use of semantic networks and a rule-based inference engine proposed by Thetis project researchers [12, 30].

Appendix A

Metadata Creator

A.1 Metadata Creator User Manual

A.1.1 Installation

The Poseidon Metadata Creator consists of a series of programs (files) that create the user interface and one Perl [13] script that operates as a Common Gateway Interface (CGI) script to receive, store and return the data submitted from the user interface. The user interface is primarily HTML code with a few Javascript enhancements, running on a standard Javascript enabled browser.

The Poseidon Metadata Creator has to be installed in a machine with a web server installed and operational. The web server has to be configured to allow data to be submitted to a CGI script (normally stored in /var/www/cgi-bin folder).

Initially setup a folder name "MetadataCreator" and copy "index.html," "display.html," and "metadataform.html" files to it. Make sure that all three files allow read access to the world. Create a subdirectory name "Metadata" to store all the metadata files. Copy the "metadata.html" to the "Metadata" directory. Within the "Metadata" directory Create sub directories "MetaDataFiles," "DbCore," and "FGDC." Within the FGDC directory create directories "Identification," "DataQuality," "SpatialOrganization," "SpatialReference," "Entity," "Distribution," and "MetadataReference." Make sure that all these directories allow read, write and execute access to the world.

Edit the <base href> tag in "index.html" to show the path to the "MetadataCreator" directory. Edit "display.html" file to show the correct paths to the "metadata.html" file

and the first available metadata file (if not available, leave blank and include them once the first metadata file has been created using the MetadataCreator).

Copy "parse.pl" to the cgi-bin directory of the web server. Change the $Form, $MDATA, $FGDC, and $DBCORE variables to the correct paths for the "MetadataCreator," "Metadata," "FGDC," and DbCore" directories. Make sure the $Form variable is in URL form and the others are listed as absolute file paths from root.

The Metadata Creator is now ready for operation. It can be used with a standard browser by pointing to the URL for the "MetadataCreator" directory. The browser will be split vertically across the middle. The left half will contain the metadataform while the right half will be split into three horizontal strips containing (from top to bottom) the "Available Metadata Files," Metadata Transfer Panel," and the "Metadata Browser" windows.

The Poseidon Metadata Creator has been installed at the Department of Ocean Engineering at MIT and is accessible via the Poseidon web page at <http://czms.mit.edu/poseidon> to authorized users.

A.1.2 Metadata Creation

The new metadata can be manually entered using the metadata form. The form contains all fifteen element of the Dublin Core standard as well as all the non-recursive elements of the FGDC content standard for digital geospatial metadata. We suggest using the Dublin Core standard to describe mostly document- like resources while the FGDC standard is more appropriate for scientific data generated from geospatially distributed experiments (the FGDC standard can be used to create metadata for documents, but is not optimally suited to this purpose). The user may provide metadata for either or both standards, but is responsible for the consistency of the information so furnished. All elements of the Dublin Core standard are optional. The elements of the FGDC standard are listed and color coded as either mandatory (red), mandatory if applicable (green), or optional (brown). The form is equipped with internal hyper-links to allow the user to easily navigate around the fields, bypassing inapplicable sections. Sections where not all fields are required for completeness, or where only a certain subset of the fields should be used for compatibility with the requirements of the standard are clearly described in the metadata form.

In order to improve the efficiency in creating metadata sets for a series of resources that have been created in the same experiment, by the same scientist, or the same geographical

region, the Metadata Creator allows the user to browse existing metadata files by using the top right window, to select the appropriate file, the middle right panel (transfer panel) to specify the section of metadata and the bottom right window to view it. The interface also allows the user to copy entire sections of metadata by selecting the appropriate metadata file and using the "COPY" buttons on the transfer panel. The copy function will write any information in the selected section of the selected file to the metadata form. It will overwrite any fields for which the metadata file contains data but will not overwrite elements that have not been filled in the selected file.

Once the metadata has been entered, it can be submitted to the server by using the "Submit" button at the bottom of the form. Once submitted the data is extracted, transformed to HTML code and stored in the server and returned to the browser window using the Perl script to write a Javascript function to the "form" frame.

The Perl script uses the "Title" fields in either FGDC or Dublin Core sections to create a title for the metadata. If both fields are filled, the FGDC "Title" will be used. If both fields are empty the file will be listed with a "No Title" heading. The file name will not contain this name and therefore duplicate titles will not affect the system (the file name is generated using a combination of the date and the process ID for the instantiation of the Perl script - see Figure 2-5).

A.1.3 Known Limitations

The Metadata Creator user interface has been tested successfully on Unix (IRIX running on SGI workstations), HP, and NT (both Netscape and Explorer) environments. It failed during data submission on a Linux platform (running on a PC). Further analysis is needed to identify and correct the causes of this incompatibility.

The Metadata form does not contain the entire FGDC standard since the standard contains recursive elements that could conceivably make the form infinitely long. Since the form is pre-coded, it cannot be expanded during run time to accommodate multiple iterations of a single field. Much of this problem can be alleviated by using the "|" key as a delimiter between multiple entries in a single field. The form also supports pre-coded multiple instantiations of entity and attribute values since these are likely to be necessary for scientific data.

The data entry fields of the form may shift if the scroll bar is dragged with a mouse

(without using the up/down scroll buttons or page up/down or scrolling arrows). The only known solution to this is to submit the incomplete form and complete it by copying the data to a new blank form (using the transfer panel) and filling in the new fields.

A.2 Metadata Creator User-Interface Code

A.2.1 Index File (index.html)

```
<html>

<title>Poseidon Metadata Creator</title>

<base href="http://czms.mit.edu/poseidon/software/MetadataCreator/">

<FRAMESET COLS="50%,*" border=0>
   <FRAME SRC="metadataform.html" name="form">
   <FRAME SRC="display.html" name="display">
</FRAMESET>

</html>
```

A.2.2 Display Control File (display.html)

```
<html>

<title>Poseidon Metadata Display</title>

<FRAMESET ROWS="30%,40%,*" border=3>
   <FRAME SRC="MetaData/metadata.html" name="level1">
   <FRAME SRC="MetaData/MetaDataFiles/metadata997269361.html"
name="level2">
   <FRAME SRC="MetaData/FGDC/Identification/id997269361.html"
name="level3">
</FRAMESET>

</html>
```

A.2.3 Metadata Form (metadataform.html)

The Poseidon Metadata Standard is a combination of the Dublin Core and Federal Geographic Data Committee Content Standard for Digital Geospatial Metadata. The Poseidon Metadata Form implements this standard as an HTML Form using over 12000 lines of code. The first part of this code is attached here.

```
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=iso-8859-1">
   <TITLE>METADATA FORM</TITLE>
</HEAD>

<BODY BGCOLOR=WHITE>

<FORM action="http://czms.mit.edu/cgi-bin/parse.pl" method="POST"
name="MetadataForm">

<INPUT type=hidden name="recipient" value="poseidon@mit.edu">

<INPUT type=hidden name="sort" value="order:dctitle,dccreator,
dcsubject,dcdescription,dcpublisher,dccontibutor,dcdate,dctype,
dcformat,dcidentifier,dcsource,dclanguage,dcrelation,dccoverage,
dcrights,Originator,pubdate,pubtime,Title,edition,geoform,SerieName,
Isson,pubplace,publish,othercit,onlink,... (line truncated)>

<INPUT type=hidden name="subject" value="Poseidon Metadata"><BR>


<CENTER><H1>Poseidon Metadata Standard</H1></CENTER><P>

<HR><P>
```

```
The Poseidon Metadata Standard is an extensible combination of the
<A HREF=#DublinCore>Dublin Core Metadata Standard</A> for documents,
and the <A HREF=#FGDC>FGDC Content Standard for Digital Geospatial
Metadata</A> and its Biological Profile (the elements of the
Biological Profile have not been implemented in this version of the
Metadata Creator). The user may complete one or both standards as
required. We recommend using the Dublin Core standard when describing
document like objects and the FGDC standards for scientific data sets
from geospatially distributed experiments. <P>

<A Name="DublinCore"></A><HR><P>

<CENTER><H2>Dublin Core Metadata Standard</H2></CENTER><P>

<HR><P>

<I>Notes:</I>  All fields of the Dublin Core Metadata Standard are
optional. When multiple instances of a field (dates, keywords, etc.)
are required, use "|" as a delimiter.

<P>
<B><I>For Additional help check out <A HREF="http://purl.oclc.org/dc/"
target=new>The Dublin Core Metadata Initiative</A> homepage, or
contact us via e-mail at <A HREF="mailto:poseidon@mit.
edu">poseidon@mit.edu</A></I>
</B>
</P>


<B>Dublin Core Metadata Standard</B><BR><P>
```

```
<BLOCKQUOTE>
  <FONT SIZE=-1>
  Title + <BR>
  Author or Creator + <BR>
  Subject and Keywords + <BR>
  Description + <BR>
  Publisher + <BR>
  Other Contributor + <BR>
  Date + <BR>
  Resource Type + <BR>
  Format + <BR>
  Resource Identifier + <BR>
  Source + <BR>
  Language + <BR>
  Relation + <BR>
  Coverage + <BR>
  Rights Management
  </FONT>
</BLOCKQUOTE>


<B>1. Title </B>
<INPUT type=text size=40 maxlength=256 name="dctitle">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  The name given to the resource by the "creator" or "publisher".
  </FONT>
</BLOCKQUOTE>


<B>2. Author or Creator </B>
```

```
<INPUT type=text size=40 maxlength=256 name="dccreator">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  The person or organization primarily responsible for creating the
intellectual content of the resource. For example, authors in the case
of written documents, artists, photographers, or illustrators in the
case of visual resources.
  </FONT>
</BLOCKQUOTE>


<B>3. Subject and Keywords </B>
<INPUT type=text size=40 maxlength=256 name="dcsubject">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  The topic of the resource. Typically, subject will be expressed as
keywords or phrases that describe the subject or content of the
resource. The use of controlled vocabularies and formal classification
schemas is encouraged.
  </FONT>
</BLOCKQUOTE>


<B>4. Description </B>
<INPUT type=text size=40 maxlength=256 name="dcdescription">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  A textual description of the content of the resource, including
abstracts in the case of document-like objects or content descriptions
```

```
in the case of visual resources.
  </FONT>
</BLOCKQUOTE>


<B>5. Publisher </B>
<INPUT type=text size=40 maxlength=256 name="dcpublisher">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  The entity responsible for making the resource available in its
present form, such as a publishing house, a university department, or
a corporate entity.
  </FONT>
</BLOCKQUOTE>


<B>6. Other Contributor </B>
<INPUT type=text size=40 maxlength=256 name="dccontibutor">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  A person or organization not specified in a CREATOR element who has
made significant intellectual contributions to the resource but whose
contribution is secondary to any person or organization specified in a
CREATOR element (for example, editor, transcriber, and illustrator).
  </FONT>
</BLOCKQUOTE>


<B>7. Date </B>
<INPUT type=text size=40 maxlength=256 name="dcdate">
```

```
<BLOCKQUOTE>
  <FONT SIZE=-1>
  The date the resource was made available in its present form.
Recommended best practice is an 8 digit number in the form YYYY-MM-DD
as defined in <A HREF="http://www.w3.org/TR/NOTE-datetime" target=new>
<B>http://www.w3.org/TR/NOTE-datetime</B></A>, a profile of ISO 8601.
In this scheme, the date element 1994-11-05 corresponds to November 5,
1994. Many other schema are possible, but if used, they should be
identified in an unambiguous manner.
  </FONT>
</BLOCKQUOTE>


<B>8. Resource Type </B>
<INPUT type=text size=40 maxlength=256 name="dctype">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  The category of the resource, such as home page, novel, poem,
working paper, technical report, essay, dictionary. For the sake of
interoperability, TYPE should be selected from an enumerated list that
is under development in the workshop series at the time of publication
of this document. See <A HREF="http://sunsite.berkeley.edu/Metadata/
types.html" target=new><B>http://sunsite.berkeley.edu/Metadata/types.
html</B></A> for current thinking on the application of this element.
  </FONT>
</BLOCKQUOTE>


<B>9. Format </B>
<INPUT type=text size=40 maxlength=256 name="dcformat">
```

```
<BLOCKQUOTE>
  <FONT SIZE=-1>
  The data format of the resource, used to identify the software and
possibly hardware that might be needed to display or operate the
resource. For the sake of interoperability, FORMAT should be selected
from an enumerated list that is under development in the workshop
series at the time of publication of this document.
  </FONT>
</BLOCKQUOTE>


<B>10. Resource Identifier </B>
<INPUT type=text size=40 maxlength=256 name="dcidentifier">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  String or number used to uniquely identify the resource. Examples
for networked resources include URLs and URNs (when implemented).
Other globally-unique identifiers,such as International Standard Book
Numbers (ISBN) or other formal names would also be candidates for this
element in the case of off-line resources.
  </FONT>
</BLOCKQUOTE>


<B>11. Source </B>
<INPUT type=text size=40 maxlength=256 name="dcsource">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  A string or number used to uniquely identify the work from which
```

this resource was derived, if applicable. For example, a PDF version of a novel might have a SOURCE element containing an ISBN number for the physical book from which the PDF version was derived.

```
  </FONT>
</BLOCKQUOTE>


<B>12. Language </B>
<INPUT type=text size=40 maxlength=256 name="dclanguage">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  Language(s) of the intellectual content of the resource. Where
practical, the content of this field should coincide with RFC 1766.
<BR>See: <B><A HREF="http://ftp.funet.fi/pub/netinfo/rfc/rfc1766.txt"
target=new>http://ftp.funet.fi/pub/netinfo/rfc/rfc1766.txt</A></B>
  </FONT>
</BLOCKQUOTE>


<B>13. Relation </B>
<INPUT type=text size=40 maxlength=256 name="dcrelation">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  The relationship of this resource to other resources. The intent of
this element is to provide a means to express relationships among
resources that have formal relationships to others, but exist as
discrete resources themselves. For example, images in a document,
chapters in a book, or items in a collection. Formal specification of
RELATION is currently under development. Users and developers should
understand that use of this element is currently considered to be
```

```
experimental.
  </FONT>
</BLOCKQUOTE>


<B>14. Coverage </B>
<INPUT type=text size=40 maxlength=256 name="dccoverage">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  The spatial and/or temporal characteristics of the resource. Formal
specification of COVERAGE is currently under development. Users and
developers should understand that use of this element is currently
considered to be experimental.<
  </FONT>
</BLOCKQUOTE>


<B>15. Rights Management </B>
<INPUT type=text size=40 maxlength=256 name="dcrights">

<BLOCKQUOTE>
  <FONT SIZE=-1>
  A link to a copyright notice, to a rights-management statement, or
to a service that would provide information about terms of access to
the resource. Formal specification of RIGHTS is currently under
development. Users and developers should understand that use of this
element is currently considered to be experimental.
  </FONT>
</BLOCKQUOTE>

:
```

```
:
:
(File Truncated)
```

A.2.4 Available Metadata Files (metadata.html)

```
<HTML>

<BODY BGCOLOR="White">

<CENTER><B>Available Metadata Files</B></CENTER><HR>

<A HREF="MetaDataFiles/metadata997269361.html" TARGET="level2">
Poseidon Metadata Creator</A><BR>
<A HREF="MetaDataFiles/metadata997269435.html" TARGET="level2">
Image of Sea Surface Temperature derived from AVHRR</A><BR>
<A HREF="MetaDataFiles/metadata997269623.html" TARGET="level2">
PRIMER  Feb16 Acoustic Full - Objectively Analyzed Fields</A><BR>
```

A.2.5 Metadata Transfer Panel (metadata997269361.html)

```
<HTML>
<HEAD>

<script language="JavaScript">
  <!-- hide from old browsers

    function copydc(){
      document.forms[0].CopyFlag.value = "true";
      top.display.level3.location.href= "../DbCore/dbcore"
+ document.forms[0].MetaDataID.value + ".html";
    }

    function copy(directory, object){
```

```
      document.forms[0].CopyFlag.value = "true";
      top.display.level3.location.href= "../FGDC/" + directory
+ "/" + object + document.forms[0].MetaDataID.value + ".html";
    }

    function loadID(){
      top.display.level3.location.href= "../FGDC/Identification/id"
+ document.forms[0].MetaDataID.value + ".html";
    }

  // -->
</script>

</HEAD>

<BODY onLoad=loadID() BGCOLOR="White">

<CENTER><B>Poseidon Metadata Creator</B><CENTER><HR>

<FORM>

<INPUT type=hidden name="MetaDataID" value="997269361">
<INPUT type=hidden name="CopyFlag" value="false">

<TABLE BORDER=2>

  <TR>
    <TD COLSPAN=2><A HREF="../DbCore/dbcore997269361.html"
TARGET="level3"> Dublin Core</A></TD>
    <TD><input type="button" name="buttondc" value="Copy"
onClick="copydc()"></TD>
  </TR>
```

```
  <TR>
    <TD COLSPAN=3> FGDC</TD>
  </TR>

  <TR>
    <TD></TD>
    <TD><A HREF="../FGDC/Identification/id997269361.html"
TARGET="level3"> Identification</A></TD>
    <TD><input type="button" name="buttonid" value="Copy"
onClick="copy('Identification', 'id')"></TD>
  </TR>

  <TR>
    <TD></TD>
    <TD><A HREF="../FGDC/DataQuality/dq997269361.html"
TARGET="level3"> DataQuality</A></TD>
    <TD><input type="button" name="buttondq" value="Copy"
onClick="copy('DataQuality', 'dq')"></TD>
  </TR>

  <TR>
    <TD></TD>
    <TD><A HREF="../FGDC/SpatialOrganization/so997269361.html"
TARGET="level3"> SpatialOrganization</A></TD>
    <TD><input type="button" name="buttonso" value="Copy"
onClick="copy('SpatialOrganization', 'so')"></TD>
  </TR>

  <TR>
    <TD></TD>
    <TD><A HREF="../FGDC/SpatialReference/sr997269361.html"
```

```
TARGET="level3"> SpatialReference</A></TD>
    <TD><input type="button" name="buttonsr" value="Copy"
onClick="copy('SpatialReference', 'sr')"></TD>
  </TR>

  <TR>
    <TD></TD>
    <TD><A HREF="../FGDC/Entity/en997269361.html"
TARGET="level3"> Entity</A></TD>
    <TD><input type="button" name="buttonen" value="Copy"
onClick="copy('Entity', 'en')"></TD>
  </TR>

  <TR>
    <TD></TD>
    <TD><A HREF="../FGDC/Distribution/db997269361.html"
TARGET="level3"> Distribution</A></TD>
    <TD><input type="button" name="buttondb" value="Copy"
onClick="copy('Distribution', 'db')"></TD>
  </TR>

  <TR>
    <TD></TD>
    <TD><A HREF="../FGDC/MetadataReference/mr997269361.html"
TARGET="level3"> MetadataReference</A></TD>
    <TD><input type="button" name="buttonmr" value="Copy"
onClick="copy('MetadataReference', 'mr')"></TD>
  </TR>

</TABLE>
</FORM>
</BODY>
```

```
</HTML>
```

A.2.6 Dublin Core Metadata File (dbcore997269361.html)

Only the Dublin Core Metadata file is shown here since all the other files are similar except for the specific information described (which is based on the specific standard).

```
<HTML>


<HEAD>
<script language="JavaScript">
  <!-- hide from old browsers


    function display(){
        top.display.level3.document.forms[0].elements["dctitle"]
.value = "Poseidon Metadata Creator"
        top.display.level3.document.forms[0].elements["dccreator"]
.value = "Pubudu Wariyapola"
        top.display.level3.document.forms[0].elements["dcsubject"]
.value = "Poseidon Metadata"
        top.display.level3.document.forms[0].elements["dcdescription"]
.value = "HTML document with CGI and Java Script enhancements for
creating metadata for use in the Poseidon System."
        top.display.level3.document.forms[0].elements["dcpublisher"]
.value = "Design Laboratory, Dept. of Ocean Engineering, Massachusetts
Institute of Technology."
        top.display.level3.document.forms[0].elements["dcdate"]
.value = "8/26/99"
        top.display.level3.document.forms[0].elements["dctype"]
.value = "HTML document"
        top.display.level3.document.forms[0].elements["dcformat"]
.value = "Software"
        top.display.level3.document.forms[0].elements["dcidentifier"]
```

```
.value = "http://czms.mit.edu/poseidon/software/MetadataCreator/"
        top.display.level3.document.forms[0].elements["dclanguage"]
.value = "English"

        if (top.display.level2.document.forms[0].CopyFlag.value ==
"true"){
            copy();
    top.display.level2.document.forms[0].CopyFlag.value =
"false";
        }
    }

    function copy(){
        top.form.document.forms[0].elements["dctitle"].value =
"Poseidon Metadata Creator"
        top.form.document.forms[0].elements["dccreator"].value =
"Pubudu Wariyapola"
        top.form.document.forms[0].elements["dcsubject"].value =
"Poseidon Metadata"
        top.form.document.forms[0].elements["dcdescription"].value =
"HTML document with CGI and Java Script enhancements for creating
metadata for use in the Poseidon System."
        top.form.document.forms[0].elements["dcpublisher"].value =
"Design Laboratory, Dept. of Ocean Engineering, Massachusetts
Institute of Technology."
        top.form.document.forms[0].elements["dcdate"].value =
"8/26/99"
        top.form.document.forms[0].elements["dctype"].value = "HTML
document"
        top.form.document.forms[0].elements["dcformat"].value =
"Software"
        top.form.document.forms[0].elements["dcidentifier"].value =
```

```
"http://czms.mit.edu/poseidon/software/MetadataCreator/"
        top.form.document.forms[0].elements["dclanguage"].value =
"English"

    }
  // -->
</script>
</HEAD>

<BODY onLoad=display() BGCOLOR="White">

<CENTER><B>Poseidon Metadata Creator - Dublin Core Metadata Standard
</B>
</CENTER><HR>

<FORM>

<B>1. Title </B>
<INPUT type=text size=40 maxlength=256 name="dctitle"><BR>

<B>2. Author or Creator </B>
<INPUT type=text size=40 maxlength=256 name="dccreator"><BR>

<B>3. Subject and Keywords </B>
<INPUT type=text size=40 maxlength=256 name="dcsubject"><BR>

<B>4. Description </B>
<INPUT type=text size=40 maxlength=256 name="dcdescription"><BR>

<B>5. Publisher </B>
<INPUT type=text size=40 maxlength=256 name="dcpublisher"><BR>
```

```
<B>6. Other Contributor </B>
<INPUT type=text size=40 maxlength=256 name="dccontibutor"><BR>

<B>7. Date </B>
<INPUT type=text size=40 maxlength=256 name="dcdate"><BR>

<B>8. Resource Type </B>
<INPUT type=text size=40 maxlength=256 name="dctype"><BR>

<B>9. Format </B>
<INPUT type=text size=40 maxlength=256 name="dcformat"><BR>

<B>10. Resource Identifier </B>
<INPUT type=text size=40 maxlength=256 name="dcidentifier"><BR>

<B>11. Source </B>
<INPUT type=text size=40 maxlength=256 name="dcsource"><BR>

<B>12. Language </B>
<INPUT type=text size=40 maxlength=256 name="dclanguage"><BR>

<B>13. Relation </B>
<INPUT type=text size=40 maxlength=256 name="dcrelation"><BR>

<B>14. Coverage </B>
<INPUT type=text size=40 maxlength=256 name="dccoverage"><BR>

<B>15. Rights Management </B>
<INPUT type=text size=40 maxlength=256 name="dcrights">

</FORM>
</BODY>
```

```
</HTML>
```

A.3 Metadata Creator Common Gateway Interface (CGI) Code

The Poseidon Metadata Creator User-Interface submits data to a Perl script running on the Poseidon Server. This script (parse.pl) parses and stores the incoming data as HTML documents with Javascript enhancements in the Poseidon Server and reloads the User-Interface page using a Javascript function. Much of this code consists of the HTML sections that are needed to format the User-Interface and have been truncated for readablity.

```
#!/usr/bin/perl

##########################################################################
#                                                                        #
#        Initialize Variables                                            #
#                                                                        #
##########################################################################

$Form = "http://czms.mit.edu/poseidon/software/MetadataCreator";
$MDATA = "/var/www/htdocs/poseidon/software/MetadataCreator/MetaData";
$FGDC = "/var/www/htdocs/poseidon/software/MetadataCreator/MetaData/
FGDC";
$DBCORE = "/var/www/htdocs/poseidon/software/MetadataCreator/MetaData/
DbCore";

@MetadataObjects = (id, dq, so, sr, en, db, mr);
%MetadataFiles = (id, IDFILE, dq, DQFILE, so, SOFILE, sr, SRFILE,
en, ENFILE, db, DBFILE,  mr, MRFILE);
%MetadataNames = (id, Identification, dq, DataQuality, so,
SpatialOrganization, sr, SpatialReference, en, Entity, db,
Distribution, mr, MetadataReference);

@Dbcore = (dctitle,dccreator,dcsubject,... (list truncated));
```

```
@Identification = (Originator,pubdate,pubtime,... (list truncated));

@DataQuality = (AttAccR,AttrAccV,AttrAccE,... (list truncated));

@SpatialOrganization = (IndSpRef,Direct,... (list truncated));

@SpatialReference = (LatRes,LongRes,GeoCU,... (list truncated));

@Entity = (EntityTypeL_1,ETDef_1,ETDefSource_1,... (list truncated));

@Distribution = (DistCntPerP,DistCntOrg,... (list truncated));

@MetadataReference = (MetaDate,MetaRevD,... (list truncated));


##########################################################################
#                                                                        #
#        Extract ENV data and read input from form                       #
#                                                                        #
##########################################################################

$method = $ENV{'REQUEST_METHOD'};
#extract method used by form (not used - assumed "POST")
$n = $ENV{'CONTENT_LENGTH'};       #length of input
read (STDIN, $buffer, $n);         #read input

($sec,$min,$hour,$mday,$mon,$year,$wday,$ydat,$isdst) = localtime();
#read time
$MetaDataFileId = "$year$mon$mday$$";
#define fileID using time and process ID

@pairs = split(/&/, $buffer);      #separate input into name/value
```

```
pairs

foreach $pair (@pairs)
#store name/value pairs in reference array
{
  ($name, $value) = split(/=/, $pair);
  $value =~ tr/+/ /;
  $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
  $contents{$name} = $value;
}



##########################################################################
#                                                                        #
#        Set data set title using FGDC (first choice) or DC title        #
#        field.                                                          #
#                                                                        #
##########################################################################

if (!$contents{Title} || $contents{Title} =~ /^\s+$/){
    if (!$contents{dctitle} || $contents{dctitle} =~ /^\s+$/){
$Title = "No Title";
    }
    else {$Title = $contents{dctitle};}
}
else {$Title = $contents{Title};}



##########################################################################
#                                                                        #
```

```
#        Define text for each data object file                          #
#                                                                       #
#########################################################################

$filetextblock1 = "
<HTML>

<HEAD>
<script language=\"JavaScript\">
  <!-- hide from old browsers

   function display(){
";

$filetextblock2 = "
       if (top.display.level2.document.forms[0].CopyFlag.value ==
\"true\"){
           copy();
   top.display.level2.document.forms[0].CopyFlag.value =
\"false\";
       }
   }

   function copy(){
";

$filetextblock3 = "
   }
  // -->
</script>
</HEAD>
```

```
<BODY onLoad=display() BGCOLOR=\"White\">
";



#########################################################################
#                                                                       #
#       Print JavaScript function to screen to reload updated files #
#       (the loading delay allows the script to complete writing    #
#       files before they are loaded).                                  #
#                                                                       #
#########################################################################

print "Content-type: text/html

<HTML>
<BODY onLoad=\"reload()\">

<script language=\"JavaScript\">
    <!-- hide
        function reload() {
    top.display.level1.location.href= \"", $Form, "/MetaData/
metadata.html\"
    top.display.level2.location.href= \"", $Form, "/MetaData/
MetaDataFiles/metadata", $MetaDataFileId, ".html\"
    top.form.location.href= \"", $Form, "/metadataform.html\"
}
// -->
</script>

</BODY>
</HTML>
```

```
";



#########################################################################
#                                                                       #
#        Print ENV variables for debugging (disabled)                   #
#                                                                       #
#########################################################################

#print "
#Time = $sec,$min,$hour,$mday,$mon,$year,$wday,$ydat,$isdst <BR>
#Getway-interface: ", $ENV{'GATEWAY_INTERFACE'}, "<BR>
#Server-name: ", $ENV{'SERVER_NAME'}, "<BR>
#Server-software: ", $ENV{'SERVER_SOFTWARE'}, "<BR>
#Server-protocol: ", $ENV{'SERVER_PROTOCOL'}, "<BR>
#Server-port: ", $ENV{'SERVER_PORT'}, "<BR>
#Request-method: ", $method, "<BR>
#Path-info: ", $ENV{'PATH_INFO'}, "<BR>
#Path-translated: ", $ENV{'PATH_TRANSLATED'}, "<BR>
#Script-name: ", $ENV{'SCRIPT_NAME'}, "<BR>
#Document-root: ", $ENV{'DOCUMENT_ROOT'}, "<BR>
#Remote-host: ", $ENV{'REMOTE_HOST'}, "<BR>
#Remote-addr: ", $ENV{'REMOTE_ADDR'}, "<BR>
#Auth-type: ", $ENV{'AUTH_TYPE'}, "<BR>
#Remote-user: ", $ENV{'REMOTE_USER'}, "<BR>
#Remote-ident: ", $ENV{'REMOTE_IDENT'}, "<BR>
#Content-type: ", $ENV{'CONTENT_TYPE'}, "<BR>
#Content-length: ", $n, "<BR>
#HTTP-from: ", $ENV{'HTTP_FROM'}, "<BR>
#HTTP-accept: ", $ENV{'HTTP_ACCEPT'}, "<BR>
#HTTP-user-agent: ", $ENV{'HTTP_USER_AGENT'}, "<BR>
```

```
#HTTP-referer: ", $ENV{'HTTP_REFERER'}, "
#</P><P>
#";



#########################################################################
#                                                                       #
#        Open "metadata.html" and write new file as a link              #
#                                                                       #
#########################################################################

open (METADATAFILE, ">>$MDATA/metadata.html");
print METADATAFILE  "<A HREF=\"MetaDataFiles/metadata",
$MetaDataFileId, ".html\" TARGET=\"level2\">", $Title, "</A><BR>\n";
close  METADATAFILE;



#########################################################################
#                                                                       #
#        Create new file for metadata                                   #
#                                                                       #
#########################################################################

open (METADATAINCIDENCE, ">$MDATA/MetaDataFiles/metadata
$MetaDataFileId.html");

print METADATAINCIDENCE "
<HTML>
<HEAD>
```

```
<script language=\"JavaScript\">
  <!-- hide from old browsers

    function copydc(){
      document.forms[0].CopyFlag.value = \"true\";
      top.display.level3.location.href= \"../DbCore/dbcore\" +
document.forms[0].MetaDataID.value + \".html\";
    }

    function copy(directory, object){
      document.forms[0].CopyFlag.value = \"true\";
      top.display.level3.location.href= \"../FGDC/\" + directory +
\"/\" + object + document.forms[0].MetaDataID.value + \".html\";
    }

    function loadID(){
      top.display.level3.location.href= \"../FGDC/Identification/id\"
+ document.forms[0].MetaDataID.value + \".html\";
    }

  // -->
</script>

</HEAD>

<BODY onLoad=loadID() BGCOLOR=\"White\">

<CENTER><B>", $Title, "</B><CENTER><HR>

<FORM>

<INPUT type=hidden name=\"MetaDataID\" value=\"", $MetaDataFileId,
```

```
"\">
<INPUT type=hidden name=\"CopyFlag\" value=\"false\">

<TABLE BORDER=2>

  <TR>
    <TD COLSPAN=2><A HREF=\"../DbCore/dbcore", $MetaDataFileId,
".html\" TARGET=\"level3\"> Dublin Core</A></TD>
    <TD><input type=\"button\" name=\"buttondc\" value=\"Copy\"
onClick=\"copydc()\"></TD>
  </TR>

  <TR>
    <TD COLSPAN=3> FGDC</TD>
  </TR>
";


foreach $MetadataObject (@MetadataObjects){
  print METADATAINCIDENCE "
  <TR>
    <TD></TD>
    <TD><A HREF=\"../FGDC/", $MetadataNames{$MetadataObject}, "/",
$MetadataObject, $MetaDataFileId, ".html\" TARGET=\"level3\"> ",
$MetadataNames{$MetadataObject}, "</A></TD>
    <TD><input type=\"button\" name=\"button", $MetadataObject, "\"
value=\"Copy\" onClick=\"copy('", $MetadataNames{$MetadataObject},
"', '", $MetadataObject, "')\"></TD>
  </TR>
";
}
```

```
print METADATAINCIDENCE "
</TABLE>
</FORM>
</BODY>
</HTML>
";

close METADATAINCIDENCE;



#########################################################################
#                                                                       #
#        Create new files for Metadata                                  #
#                                                                       #
#########################################################################

open (DBCOREFILE, ">>/$DBCORE/dbcore$MetaDataFileId.html");
foreach $MetadataObject (@MetadataObjects){
    open ($MetadataFiles{$MetadataObject}, ">>$FGDC/$MetadataNames
{$MetadataObject}/$MetadataObject$MetaDataFileId.html");
}



#########################################################################
#                                                                       #
#        Print Dublin Core Metadata to file                             #
#                                                                       #
#########################################################################

print DBCOREFILE $filetextblock1;
```

```
foreach $name (@Dbcore){
    if ($contents{$name}){
print DBCOREFILE "        top.display.level3.document.
forms[0].elements[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print DBCOREFILE $filetextblock2;

foreach $name (@Dbcore){
    if ($contents{$name}){
print DBCOREFILE "        top.form.document.forms[0].elements
[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print DBCOREFILE $filetextblock3;

print DBCOREFILE "
<CENTER><B>$Title - Dublin Core Metadata Standard</B></CENTER><HR>

<FORM>

:
:
:
(Print elements of the standard to HTML file - Truncated)
";
```

```
#########################################################################
#                                                                       #
#        Print FGDC/Identification Metadata to file                     #
#                                                                       #
#########################################################################


print IDFILE $filetextblock1;

foreach $name (@Identification){
    if ($contents{$name}){
print IDFILE "        top.display.level3.document.forms[0].
elements[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print IDFILE $filetextblock2;

foreach $name (@Identification){
    if ($contents{$name}){
print IDFILE "        top.form.document.forms[0].elements
[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print IDFILE $filetextblock3;

print IDFILE "
<CENTER><B>$Title - Identification Information</B></CENTER><HR>

<FORM>
```

```
:
:
:
(Print elements of the standard to HTML file - Truncated)
";


#############################################################################
#                                                                           #
#        Print FGDC/Data Quality Metadata to file                           #
#                                                                           #
#############################################################################

print DQFILE $filetextblock1;

foreach $name (@DataQuality){
    if ($contents{$name}){
print DQFILE "        top.display.level3.document.forms[0].
elements[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print DQFILE $filetextblock2;

foreach $name (@DataQuality){
    if ($contents{$name}){
print DQFILE "        top.form.document.forms[0].elements
[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print DQFILE $filetextblock3;
```

```
print DQFILE "
<CENTER><B>$Title - Data Quality Information</B></CENTER><HR>

<FORM>

:
:
:
(Print elements of the standard to HTML file - Truncated)
";



#########################################################################
#                                                                       #
#        Print FGDC/Spatial Data Organization Metadata to file          #
#                                                                       #
#########################################################################

print SOFILE $filetextblock1;

foreach $name (@SpatialOrganization){
    if ($contents{$name}){
print SOFILE "        top.display.level3.document.forms[0].
elements[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print SOFILE $filetextblock2;

foreach $name (@SpatialOrganization){
```

```
    if ($contents{$name}){
print SOFILE "        top.form.document.forms[0].elements
[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print SOFILE $filetextblock3;

print SOFILE "
<CENTER><B>$Title - Spatial Data Organization Information</B></CENTER>
<HR>

<FORM>

:
:
:
(Print elements of the standard to HTML file - Truncated)
";




#########################################################################
#                                                                       #
#        Print FGDC/Spatial Data Reference Metadata to file             #
#                                                                       #
#########################################################################

print SRFILE $filetextblock1;

foreach $name (@SpatialReference){
    if ($contents{$name}){
```

```
print SRFILE "        top.display.level3.document.forms[0].
elements[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print SRFILE $filetextblock2;

foreach $name (@SpatialReference){
    if ($contents{$name}){
print SRFILE "        top.form.document.forms[0].elements
[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print SRFILE $filetextblock3;

print SRFILE "
<CENTER><B>$Title - Spatial Data Reference Information</B></CENTER>
<HR>

<FORM>

:
:
:
(Print elements of the standard to HTML file - Truncated)
";

#########################################################################
#                                                                       #
```

```
#        Print FGDC/Entity Metadata to file                              #
#                                                                        #
##########################################################################

print ENFILE $filetextblock1;

foreach $name (@Entity){
    if ($contents{$name}){
print ENFILE "        top.display.level3.document.forms[0].
elements[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print ENFILE $filetextblock2;

foreach $name (@Entity){
    if ($contents{$name}){
print ENFILE "        top.form.document.forms[0].elements
[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print ENFILE $filetextblock3;

print ENFILE "
<CENTER><B>$Title - Entity Information</B></CENTER><HR>

<FORM>

:
:
:
```

```
(Print elements of the standard to HTML file - Truncated)
";



#########################################################################
#                                                                       #
#        Print FGDC/Distribution Metadata to file                       #
#                                                                       #
#########################################################################

print DBFILE $filetextblock1;

foreach $name (@Distribution){
    if ($contents{$name}){
print DBFILE "        top.display.level3.document.forms[0].
elements[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print DBFILE $filetextblock2;

foreach $name (@Distribution){
    if ($contents{$name}){
print DBFILE "        top.form.document.forms[0].elements
[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print DBFILE $filetextblock3;

print DBFILE "
```

```
<CENTER><B>$Title - Distribution Information</B></CENTER><HR>

<FORM>

:

:

:

(Print elements of the standard to HTML file - Truncated)

";



##########################################################################

#                                                                        #

#        Print FGDC/Metadata Reference Metadata to file                  #

#                                                                        #

##########################################################################

print MRFILE $filetextblock1;

foreach $name (@MetadataReference){

    if ($contents{$name}){

print MRFILE "        top.display.level3.document.forms[0].

elements[\"$name\"].value = \"$contents{$name}\"\n";

    }

}

print MRFILE $filetextblock2;

foreach $name (@MetadataReference){

    if ($contents{$name}){

print MRFILE "        top.form.document.forms[0].elements
```

```
[\"$name\"].value = \"$contents{$name}\"\n";
    }
}

print MRFILE $filetextblock3;

print MRFILE "
<CENTER><B>$Title - Metadata Reference Information</B></CENTER><HR>

<FORM>

:
:
:
(Print elements of the standard to HTML file - Truncated)
";



##############################################################################
#                                                                            #
#        Close files                                                         #
#                                                                            #
##############################################################################


close DBCOREFILE;
close IDFILE;
close DQFILE;
close SOFILE;
close SRFILE;
close ENFILE;
```

```
close DBFILE;
close MRFILE;


exit(0);
```

Appendix B

Ontology Creator

B.1 Ontology Creator User Manual

B.1.1 Installation

The Poseidon Ontology Creator has been written completely in Java as an applet and application daemon. It has a graphical user interface that can run on a standard Java-enabled web browser, an application back-end running on the Poseidon server and a socket connection to communicate between the two.

In order to install the ontology creator, the server address, port number and the name of the ontology have to be included in the code and the code needs to be recompiled to run on the selected platform. The port number must be entered as the "listenport" in "DataServerThread.java" and the "port" in "ontologyAppletSocke.java." The server address has to be entered in DataServerThread() constructor in "DataServerThread.java." The ontology name must be listed in "OntologyFileIO.java" as "inputFileName" and in "OntologyApplet.java" as "ontologyName." In order to provide edit access to the ontology, create username/password pairs by editing "Login.java" as necessary.

Once the above files have been edited and all the source code files of the Ontology Creator (OntologyApplet, OntologyAppletSocket, Login, MessageBox, DataServer, DataServerThread, OntologyFileIO, Ontology, OntologyVector and ArraySort) have been compiled, create a directory called "OntologyGUI" and copy the OntologyApplet, OntologyAppletSocket, Login, MessageBox, Ontology, OntologyVector and ArraySort classes to this directory. Also copy the "OntologyApplet.html" file to this directory and change the <APPLET CODEBASE>

tag to show the complete URL for the location of the "OntologyApplet.class" file (this URL is used by the applet to communicate with the socket). The Ontology Server and the GUI have to be on the same server. Make sure all the files in this folder can be read and executed by the world.

Create a different directory named "OntologyServer" and copy DataServer, DataServerThread, OntologyFileIO, Ontology, OntologyVector, and ArraySort classes to it. Create a text file called <OntologyName>.txt (delete spaces in the ontology name) following the ontology element file structure described in figure 3-2 and appendix B.5 to contain the top element of the ontology. Also create a directory with the same name as the file (without the ".txt" extension) in the "OntologyServer" directory. Make sure that the directory and the files allow read, write and execute access to the world.

Run the server as a background daemon using "java DataServer &" (Unix) or corresponding command.

An installation of the Ontology Creator is available in the Department of Ocean Engineering at MIT, accessible via the Poseidon web page at <http://czms.mit.edu/poseidon/> to authorized users.

B.1.2 Ontology Creation

The Ontology Creator allows all users to browse the existing ontology and allows authenticated users to add, edit and delete elements in the ontology. When the system starts up it will bring up a log-in screen. A user can also log-in later by choosing the log-in button.

In order to add an element, select the area in which the element is to be added (parent) and press the "Add Child" button. The interface will then provide a screen with three open fields (name, definition and reference) to allow the user to add the element to the vocabulary. Once the user has entered the information, it can be submitted, cleared, or the operation cancelled by pressing the appropriate button. If the operation is cancelled the system returns to browsing mode. If the new element is submitted, the local and the remote copies of the ontology are updated and the process is acknowledged. Elements can only be added to the lower two levels of the ontology.

Similarly the user can edit the definition or reference fields of an element. The element name cannot be changed once created (it is possible to delete it and re-create a new element with the new name). The element to be edited must be selected prior to choosing the edit

option. All elements can be edited.

An element can be deleted by selecting it and pressing "delete." An element can be deleted only if it is either a lowest level element or has no children and can be only done by a user with "editor" authorization. Once deleted an element cannot be recovered.

The system also allows users to view the history of modification for an element by selecting the element and choosing the "show history" option.

B.1.3 Known Limitations

While the Poseidon Ontology Creator User Interface has been written in Java which is supposed to be platform independent, the applet has only been found to work in a Unix (IRIX running on SGI workstations) and Linux (running on a PC) environments. When tested on a PC running NT, it generated an "Array index out of bounds" error on a "Netscape" browser, and a "Microsoft Security Exception" on "Microsoft Explorer." Further analysis is needed to identify and correct the causes of these incompatibilities.

The message boxes generated by the system to acknowledge operation or warn the user of error will only be displayed once the user moves the mouse.

The log-in function is not truly secure since the information can be accessed by decompiling the source code.

There is potential for some data to be lost if two users create or update the same element without re-loading the ontology from the server after the first add/update has been performed. (If user 'B' accesses the system before user 'A' submits a new or updated element to the ontology, the information submitted by 'A' will be lost when user 'B' adds or updates the same element without re-loading the ontology.) A possible race condition may also exist if two or more users attempt to write the same file at the same instant in time. If this becomes a problem in practice, it is suggested that a checkpointing scheme be adopted to prevent multiple users (threads) accessing the same information simultaneously.

B.2 Ontology Creator User-Interface Code

B.2.1 Browser Interface (OntologyApplet.html)

```
<HTML>
<BODY>
<CENTER><APPLET CODEBASE=http://deslab.mit.edu/~pwariyap/Poseidon/
Java/OntologyCreator.V0_6/ CODE=OntologyApplet.class WIDTH=510
HEIGHT=810>
</APPLET></CENTER>
</BODY>
</HTML>
```

B.2.2 Applet Generator (OntologyApplet.java)

```
import java.awt.*;
import java.util.*;
import java.util.Hashtable.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.lang.*;


public class OntologyApplet extends Applet {
    String ontologyName = "Poseidon Ontology";

    Vector ontology;
    OntologyVector onvect;
    Ontology element;
    Ontology parentElement;
    MessageBox messageBox;

    Panel ontologyPanel;
```

```
Panel ontologyHoldingPanel;
Panel topElementPanel;
Panel childElementPanel;
Panel topChildElementPanel;
Panel bottomChildElementPanel;
Panel topActionPanel;

Panel elementPanel;
Panel idPanel;
Panel dataPanel;
Panel childPanel;
Panel defPanel;
Panel addDataPanel;
Panel actionPanel;

Button topElement;
Button[] topChildElement;
Label bottomChildLabel;
List bottomChildElement;

Button Login;
Button Add;
Button Edit;
Button Delete;
Button History;

TextField ID;
TextArea Definition;
TextArea Reference;
TextField Level;
TextField Parent;
TextField NumChild;
```

```
    List Child;

    Vector childrenVector;

    OntologyAppletSocket onsocket;
    InetAddress address;
    DatagramSocket socket = null;

    String creator = "Default";
    Login login;
    boolean verified = false;

    public OntologyApplet() {
        setLayout(new BorderLayout(5,5));
ontology = new Vector();
onvect = new OntologyVector();
login = new Login(this);
    }


    public void init() {
        resize(510, 800);
    }

    public void setCreator(String newCreator) {
        creator = newCreator;
verified = true;

        element = onvect.findElement(ontologyName);

modifyBottomChildElementPanel((Ontology)onvect.findChildren
(element).elementAt(0));
```

```
modifyTopActionPanel(true);
modifyElementPanel(false,false);
createActionPanel("Submit", "Clear", false);
validate();
    }

    public void start(){
        try {
            address = InetAddress.getByName(getCodeBase().getHost());
        } catch (UnknownHostException e) {}

onsocket = new OntologyAppletSocket(address);
onsocket.initSocket();
onsocket.createOntology();
ontology = onsocket.getOntology();

onvect.storeOntology(ontology);

createGUI();
    }

    private void createGUI() {
element = onvect.findElement(ontologyName);

        hide();
        removeAll();

ontologyPanel = new Panel();
ontologyPanel.resize(510,400);
ontologyPanel.setLayout(new BorderLayout(5,5));

ontologyHoldingPanel = new Panel();
```

```
ontologyHoldingPanel.resize(510,350);
ontologyHoldingPanel.setLayout(new GridLayout(1,3,5,5));

topElementPanel = new Panel();
topElementPanel.resize(150, 350);
topElementPanel.setLayout(new BorderLayout(5,5));

topChildElementPanel = new Panel();
topChildElementPanel.resize(150,350);

bottomChildElementPanel = new Panel();
bottomChildElementPanel.resize(150,350);

topActionPanel = new Panel();
topActionPanel.resize(510,50);
topActionPanel.setBackground(Color.green);

createOntologyPanel(ontologyName);
add("North", ontologyPanel);

elementPanel = new Panel();
elementPanel.resize(510,400);
elementPanel.setLayout(new BorderLayout(5,5));

idPanel = new Panel();
idPanel.resize(500, 50);
idPanel.setLayout(new GridLayout(1,2,5,5));

dataPanel = new Panel();
dataPanel.resize(510,250);
dataPanel.setLayout(new GridLayout(2,1,5,5));
```

```
defPanel = new Panel();
defPanel.resize(510,150);
defPanel.setLayout(new GridLayout(2,2,5,5));

addDataPanel = new Panel();
addDataPanel.resize(510,100);
addDataPanel.setLayout(new GridLayout(3,2,5,5));

childPanel = new Panel();
childPanel.resize(510,100);
childPanel.setLayout(new GridLayout(1,2,5,5));

createElementPanel(false);
add("Center", elementPanel);

actionPanel = new Panel();
actionPanel.resize(510,50);
actionPanel.setBackground(Color.green);

createActionPanel("Submit", "Clear", false);
add("South", actionPanel);

validate();
show();
    }


    private void createOntologyPanel(String ontologyName) {
int nc;
String[] childNames;

ontologyPanel.removeAll();
```

```
  ontologyHoldingPanel.removeAll();

    topElementPanel.removeAll();

      topElement = new Button(ontologyName);

    topElementPanel.add("Center", topElement);

  ontologyHoldingPanel.add(topElementPanel);

    element = onvect.findElement(ontologyName);
    nc = element.getNumChildren();
    childNames = element.getChildren();

    topChildElementPanel.removeAll();
    topChildElementPanel.setLayout(new GridLayout(nc,1,5,5));

      topChildElement = new Button[nc];

      for (int i=0; i<nc; i++) {
          topChildElement[i] = new Button(childNames[i]);
  topChildElementPanel.add(topChildElement[i]);
      }

  ontologyHoldingPanel.add(topChildElementPanel);

  createBottomChildElementPanel((Ontology)onvect.findChildren
(element).elementAt(0));
  ontologyHoldingPanel.add(bottomChildElementPanel);

ontologyPanel.add("Center", ontologyHoldingPanel);
```

```
createTopActionPanel(true);
ontologyPanel.add("South", topActionPanel);
    }

    private void modifyOntologyPanel(String ontologyName) {
int nc;
String[] childNames;

topElement.setLabel(ontologyName);

element = onvect.findElement(ontologyName);
nc = element.getNumChildren();
childNames = element.getChildren();

topChildElementPanel.hide();
topChildElementPanel.removeAll();
topChildElementPanel.setLayout(new GridLayout(nc,1,5,5));

topChildElement = new Button[nc];

for (int i=0; i<nc; i++) {
    topChildElement[i] = new Button(childNames[i]);
    topChildElementPanel.add(topChildElement[i]);
}
topChildElementPanel.show();

modifyBottomChildElementPanel((Ontology)onvect.findChildren
(element).elementAt(0));
modifyTopActionPanel(true);
    }
```

```
    private void createBottomChildElementPanel(Ontology childElement)
{
String[] childNames;
        element = childElement;

bottomChildElementPanel.removeAll();
bottomChildElementPanel.setLayout(new BorderLayout(5,5));

  bottomChildLabel = new Label(element.getID(), Label.CENTER);

bottomChildElementPanel.add("North", bottomChildLabel);

  bottomChildElement = new List(10, false);

  childNames = element.getChildren();

  if (element.getNumChildren() > 0){
      for (int i=0; i<element.getNumChildren(); i++) {
  bottomChildElement.addItem(childNames[i]);
      }
  }else {
      bottomChildElement.addItem("None");
  }

bottomChildElementPanel.add("Center", bottomChildElement);
    }

    private void modifyBottomChildElementPanel(Ontology childElement)
{
String[] childNames;
int nc;
```

```
        element = childElement;
nc = element.getNumChildren();
childNames = element.getChildren();

bottomChildLabel.setText(element.getID());

bottomChildElement.delItems(0, bottomChildElement.countItems()
 - 1);
if (nc != 0)
    for (int i=0; i<nc; i++)
      bottomChildElement.addItem(childNames[i]);
else
    bottomChildElement.addItem("None");
    }


    private void createTopActionPanel(boolean editable) {
topActionPanel.removeAll();

Login = new Button("Login");
Add = new Button("Add Child");
Edit = new Button("Edit Element");
Delete = new Button("Delete Element");
History = new Button("Show History");

topActionPanel.add(Login);
topActionPanel.add(Add);
topActionPanel.add(Edit);
topActionPanel.add(Delete);
topActionPanel.add(History);
```

```
if (editable) {
    if (element.getLevel() == 0)
        Edit.disable();

    if (element.getLevel() == 2)
        Add.disable();

    if ((element.getNumChildren() != 0) ||
(element.getLevel() == 0) || !creator.equals("Creator"))
        Delete.disable();
} else {
    Add.disable();
    Edit.disable();
    Delete.disable();
}
    }

    private void modifyTopActionPanel(boolean editable) {
Login.enable();
        Add.enable();
Edit.enable();
Delete.enable();

if (editable) {
    if (element.getLevel() == 0)
        Edit.disable();

    if (element.getLevel() == 2)
        Add.disable();

    if ((element.getNumChildren() != 0) ||
(element.getLevel() == 0) || !creator.equals("Creator"))
```

```
        Delete.disable();
} else {
    Login.disable();
    Add.disable();
    Edit.disable();
    Delete.disable();
}
    }


    private void createElementPanel(boolean edit) {
        if (element.getLevel() == 0) {
    topElement.setBackground(Color.gray);
    for (int i=0; i<topChildElement.length; i++)
        topChildElement[i].setBackground(Color.lightGray);
} else if (element.getLevel() == 1) {
    topElement.setBackground(Color.lightGray);
    for (int i=0; i<topChildElement.length; i++)
        if (topChildElement[i].getLabel().equals(element.
getID()))
    topChildElement[i].setBackground(Color.gray);
else
    topChildElement[i].setBackground(Color.lightGray);
} else {
    topElement.setBackground(Color.lightGray);
    for (int i=0; i<topChildElement.length; i++)
        topChildElement[i].setBackground(Color.lightGray);
}

int nc = element.getNumChildren();
String[] childNames = element.getChildren();
```

```
elementPanel.removeAll();

  idPanel.removeAll();

  idPanel.add(new Label("Element Name"));
  ID = new TextField(element.getID(), 30);
  idPanel.add(ID);
  ID.setEditable(edit);

elementPanel.add("North", idPanel);

  dataPanel.removeAll();

    defPanel.removeAll();

    defPanel.add(new Label("Element Definition"));
    Definition = new TextArea(element.getDef(), 5, 60);
    defPanel.add(Definition);
    Definition.setEditable(edit);

    defPanel.add(new Label("Reference Source for
Definition"));
    Reference = new TextArea(element.getRef(), 5, 60);
    defPanel.add(Reference);
    Reference.setEditable(edit);

          dataPanel.add(defPanel);

    addDataPanel.removeAll();

      addDataPanel.add(new Label("Element Level"));
      Level = new TextField(Integer.toString(element.
```

```
getLevel()), 5);
      addDataPanel.add(Level);
      Level.setEditable(false);

      addDataPanel.add(new Label("Parent"));
      Parent = new TextField(element.getParent(), 30);
      addDataPanel.add(Parent);
      Parent.setEditable(false);

      addDataPanel.add(new Label("Number of Children"));
      NumChild = new TextField(Integer.toString(nc), 5);
      addDataPanel.add(NumChild);
      NumChild.setEditable(false);

          dataPanel.add(addDataPanel);

        elementPanel.add("Center", dataPanel);


  childPanel.removeAll();

  childPanel.add(new Label("Children", Label.LEFT));

    Child = new List(5, false);
    Child.enable(false);

    if (nc != 0)
        for (int i=0; i<nc; i++)
    Child.addItem(childNames[i]);
    else
        Child.addItem("None");
```

```
  childPanel.add(Child);

        elementPanel.add("South", childPanel);
    }


    private void modifyElementPanel(boolean edit, boolean add) {
int nc = element.getNumChildren();
String[] childNames = element.getChildren();

        if (element.getLevel() == 0) {
    topElement.setBackground(Color.gray);
    for (int i=0; i<topChildElement.length; i++)
        topChildElement[i].setBackground(Color.lightGray);
} else if (element.getLevel() == 1) {
    topElement.setBackground(Color.lightGray);
    for (int i=0; i<topChildElement.length; i++)
        if (topChildElement[i].getLabel().equals(element.
getID()))
    topChildElement[i].setBackground(Color.gray);
else
    topChildElement[i].setBackground(Color.lightGray);
} else {
    topElement.setBackground(Color.lightGray);
    for (int i=0; i<topChildElement.length; i++)
        topChildElement[i].setBackground(Color.lightGray);
}

        ID.setText(element.getID());
ID.setEditable(edit);

if (add) {
```

```
    defPanel.hide();
    defPanel.removeAll();

    defPanel.add(new Label("Element Definition"));
    Definition = new TextArea(element.getDef(), 5, 60);
    defPanel.add(Definition);
    Definition.setEditable(edit);

    defPanel.add(new Label("Reference Source for
Definition"));
    Reference = new TextArea(element.getRef(), 5, 60);
    defPanel.add(Reference);
    Reference.setEditable(edit);
    defPanel.show();
} else {
    Definition.setText(element.getDef());
    Definition.setEditable(edit);
    Reference.setText(element.getRef());
    Reference.setEditable(edit);
}


Level.setText(Integer.toString(element.getLevel()));
Parent.setText(element.getParent());
NumChild.setText(Integer.toString(nc));

Child.delItems(0, Child.countItems() - 1);
if (nc != 0)
    for (int i=0; i<nc; i++)
        Child.addItem(childNames[i]);
else
    Child.addItem("None");
```

```
    }

    private void createActionPanel(String button1Name, String
button2Name, boolean edit) {
        actionPanel.hide();
actionPanel.removeAll();

Button button1 = new Button(button1Name);
Button button2 = new Button(button2Name);

actionPanel.add(button1);
actionPanel.add(button2);

if (!edit) {
      button1.disable();
      button2.disable();
}
actionPanel.show();
    }

    private void createActionPanel(String button1Name, String
button2Name, String button3Name, boolean edit) {
        actionPanel.hide();
actionPanel.removeAll();

Button button1 = new Button(button1Name);
Button button2 = new Button(button2Name);
Button button3 = new Button(button3Name);

actionPanel.add(button1);
actionPanel.add(button2);
actionPanel.add(button3);
```

```
if (!edit) {
      button1.disable();
      button2.disable();
      button3.disable();
}
actionPanel.show();
    }


    public boolean handleEvent(Event event){
        if (event.target instanceof List) {
    List list = (List)(event.target);

    if (event.id == Event.LIST_SELECT) {
        int listIndex = ((Integer)event.arg).intValue();
boolean editable = true;

if (!list.getItem(listIndex).equals("None")){

    element = onvect.findElement(ontologyName);
    childrenVector = onvect.findChildren(element);

    for (int i=0; i<childrenVector.size(); i++){
        if (bottomChildLabel.getText().
equals(((Ontology)childrenVector.elementAt(i)).getID())) {
    element = (Ontology)childrenVector.
elementAt(i);
    childrenVector = onvect.
findChildren(element);
    element = (Ontology)childrenVector.
elementAt(listIndex);
```

```
}
    }
}else {
    element = new Ontology();
    editable = false;
}

modifyTopActionPanel(editable);
modifyElementPanel(false,false);
    }
}
        if (event.id == Event.WINDOW_DESTROY) {
    System.exit(0);
}
        return super.handleEvent(event);
    }



    public boolean action(Event event, Object arg) {
        if (event.target instanceof Button) {
    if (((String)arg).equals("Login")){
login = new Login(this);
return true;
    }
    if (((String)arg).equals("Add Child")){
        if (verified)
    addChild();
return true;
    }
    if (((String)arg).equals("Submit")){
        if (verified)
    processAddChild();
```

```
return true;
    }
    if (((String)arg).equals("Clear")){
        if (verified)
    clearAddChild();
return true;
    }
    if (((String)arg).equals("Edit Element")){
        if (verified)
    editElement();
return true;
    }
    if (((String)arg).equals("Accept Changes")){
        if (verified)
    processEditElement();
return true;
    }
    if (((String)arg).equals("Reset Values")){
        if (verified)
    resetEditElement();
return true;
    }
    if (((String)arg).equals("Delete Element")){
        if (verified)
    deleteElement();
return true;
    }
    if (((String)arg).equals("Delete")){
        if (verified)
    processDeleteElement();
return true;
    }
```

```
    if (((String)arg).equals("Cancel")){
        if (verified)
    cancelCommand();
return true;
    }
    if (((String)arg).equals("Show History")){
        if (verified)
    showHistory();
return true;
    }

    element = onvect.findElement((String)arg);

    if (element.getLevel() != 0) {
        modifyBottomChildElementPanel(element);
    }

    modifyTopActionPanel(true);
    modifyElementPanel(false,false);

    createActionPanel("Submit", "Clear", false);

    validate();
}
return true;
    }


    private void addChild(){
        parentElement = element;
        element = new Ontology(parentElement.getID(), parentElement.
getLevel() + 1);
```

```
modifyTopActionPanel(false);
modifyElementPanel(true,true);

createActionPanel("Submit", "Clear", "Cancel", true);

validate();
    }



    private void processAddChild(){
        String tempID = ID.getText().trim();
String tempDef = Definition.getText().trim().replace('\n',
' ');
String tempRef = Reference.getText().trim().replace('\n',
' ');
char ch;
Character Ch;
Frame messageBox;

for (int i=0; i<parentElement.getNumChildren(); i++){
    if ((parentElement.getChildren())[i].equals(tempID)){

        messageBox = new MessageBox("Element Already Exists.",
"Please choose new Element Name or edit existing element.",
parentElement);
ID.selectAll();
return;
    }
}

if (Character.isDigit(tempID.charAt(0))){
```

```
    ID.selectAll();
    messageBox = new MessageBox("Invalid Element Name.",
"First character Must be a letter.");
    return;
}

for (int i=0; i<tempID.length(); i++) {
    ch = tempID.charAt(i);
    if(!(Character.isLetterOrDigit(ch) || Character.
isSpace(ch))) {
ID.selectAll();
messageBox = new MessageBox("Invalid Element Name.",
"Use alphanumeric characters only.");
return;
    }
}

for (int i=0; i<tempDef.length(); i++) {
    Ch = new Character(tempDef.charAt(i));
    if(Ch.equals(new Character('$'))) {
Definition.selectAll();
messageBox = new MessageBox("Invalid Element
Definition.", "Cannot use '$' Character.");
return;
    }
}

for (int i=0; i<tempRef.length(); i++) {
    Ch = new Character(tempRef.charAt(i));
    if(Ch.equals(new Character('$'))) {
Reference.selectAll();
messageBox = new MessageBox("Invalid Element
```

```
Reference.", "Cannot use '$' Character.");
return;
    }
}

String tempParent = parentElement.getID();
int tempNC = 0;
int tempLevel = parentElement.getLevel() + 1;
String[] tempChild = null;

Calendar cal = Calendar.getInstance();
String[] tempCreator = new String[1];
tempCreator[0] = creator;
Date[] tempDate = new Date[1];
tempDate[0] = cal.getTime();
String[] tempDefHistory = new String[1];
tempDefHistory[0] = tempDef;
String[] tempRefHistory = new String[1];
tempRefHistory[0] = tempRef;

element = new Ontology(tempID, tempDef, tempRef, tempParent,
tempChild, 0, tempLevel, 1, tempCreator, tempDate, tempDefHistory,
tempRefHistory);

onsocket.processRemoteAddChild(element,false);

onvect.addVectorElement(element);

onvect.deleteVectorElement(parentElement);
parentElement.addChild(tempID);
onvect.addVectorElement(parentElement);
```

```
if (element.getLevel() == 2){
    element = parentElement;

    modifyBottomChildElementPanel(element);
    modifyTopActionPanel(true);
} else {
    modifyOntologyPanel("ontologyName");
}

modifyElementPanel(false,false);
createActionPanel("Submit", "Clear", false);

validate();
    }


    private void clearAddChild(){
modifyElementPanel(true,true);
validate();
    }

    private void editElement() {
        modifyTopActionPanel(false);
modifyElementPanel(true,false);
ID.setEditable(false);
createActionPanel("Accept Changes", "Reset Values", "Cancel",
true);
validate();
    }

    private void processEditElement(){
        String tempID = element.getID();
```

```
String tempDef = Definition.getText().trim().replace('\n', ' ');
String tempRef = Reference.getText().trim().replace('\n', ' ');
char ch;
Character Ch;
Frame messageBox;

if (!element.getDef().equals(tempDef) || !element.getRef().
equals(tempRef)){
    if (!element.getDef().equals(tempDef)) {
        for (int i=0; i<tempDef.length(); i++) {
    Ch = new Character(tempDef.charAt(i));
    if(Ch.equals(new Character('$'))) {
Definition.selectAll();
messageBox = new MessageBox("Invalid Element
Definition.", "Cannot use '$' Character.");
return;
    }
}
    }

    if (!element.getRef().equals(tempRef)) {
        for (int i=0; i<tempRef.length(); i++) {
    Ch = new Character(tempRef.charAt(i));
    if(Ch.equals(new Character('$'))) {
Reference.selectAll();
messageBox = new MessageBox("Invalid Element
Reference.", "Cannot use '$' Character.");
return;
    }
}
    }
```

```
    String tempParent = element.getParent();
    int tempNC = element.getNumChildren();
    int tempLevel = element.getLevel();
    String[] tempChild = element.getChildren();
    int tempNH = element.getNumHistory();
    String[] temp1Creator = element.getCreator();
    Date[] temp1Date = element.getDate();
    Calendar cal = Calendar.getInstance();

    String[] temp2Creator = new String[tempNH+1];
    System.arraycopy(temp1Creator, 0, temp2Creator, 0,
tempNH);
    temp2Creator[tempNH] = creator;

    Date[] temp2Date = new Date[tempNH+1];
    System.arraycopy(temp1Date, 0, temp2Date, 0, tempNH);
    temp2Date[tempNH] = cal.getTime();

    String[] tempDefHistory = new String[tempNH+1];
    System.arraycopy(element.getDefHistory(), 0,
tempDefHistory, 0, tempNH);
    tempDefHistory[tempNH] = element.getDef();

    String[] tempRefHistory = new String[tempNH+1];
    System.arraycopy(element.getRefHistory(), 0,
tempRefHistory, 0, tempNH);
    tempRefHistory[tempNH] = element.getRef();

    tempNH++;

    parentElement = onvect.findParent(element);
```

```
    onsocket.processRemoteDeleteElement(element,true);

    onvect.deleteVectorElement(element);

    onvect.deleteVectorElement(parentElement);
    parentElement.deleteChild(tempID);
    onvect.addVectorElement(parentElement);


    element = new Ontology(tempID, tempDef, tempRef,
tempParent, tempChild, tempNC, tempLevel, tempNH, temp2Creator,
temp2Date, tempDefHistory, tempRefHistory);

    onsocket.processRemoteAddChild(element,true);

    onvect.addVectorElement(element);

    onvect.deleteVectorElement(parentElement);
    parentElement.addChild(tempID);
    onvect.addVectorElement(parentElement);
}

        modifyTopActionPanel(true);
modifyElementPanel(false,false);
createActionPanel("Submit", "Clear", false);
validate();
    }

    private void resetEditElement() {
modifyElementPanel(true,false);
```

```
ID.setEditable(false);
validate();
    }

    private void deleteElement() {
        modifyTopActionPanel(false);
modifyElementPanel(false,false);
createActionPanel("Delete", "Cancel", true);
validate();
    }


    private void processDeleteElement(){
        String tempID = element.getID();

  onsocket.processRemoteDeleteElement(element, false);

parentElement = onvect.findParent(element);

       onvect.deleteVectorElement(element);

       onvect.deleteVectorElement(parentElement);
parentElement.deleteChild(tempID);
        onvect.addVectorElement(parentElement);

if (element.getLevel() == 2){
    element = parentElement;

    modifyBottomChildElementPanel(element);
    modifyTopActionPanel(true);
} else
    modifyOntologyPanel(ontologyName);
```

```
modifyElementPanel(false,false);
createActionPanel("Submit", "Clear", false);
validate();
    }


    private void cancelCommand() {
        element = onvect.findElement(ontologyName);

modifyBottomChildElementPanel((Ontology)onvect.
findChildren(element).elementAt(0));
modifyTopActionPanel(true);
modifyElementPanel(false,false);
createActionPanel("Submit", "Clear", false);
validate();
    }

    private void showHistory() {
        messageBox = new MessageBox("Element Creation History.",
element);
    }

}
```

B.2.3 Socket Interface Tool (OntologyAppletSocket.java)

```
import java.util.*;
import java.io.*;
import java.net.*;
import java.lang.*;
```

```
class OntologyAppletSocket {
    int port = 8020;

    Vector ontology;

    InetAddress address;
    DatagramSocket socket = null;


    OntologyAppletSocket(InetAddress newAddress) {
        address = newAddress;
ontology = new Vector();
    }

    public void initSocket() {
        try {
            socket = new DatagramSocket();
        } catch (IOException e) {
            return;
        }
    }

    public void createOntology() {
        DatagramPacket packet;
        byte[] buf;
String dataString;
String received;
boolean done = false;

try {                                          // send request
    dataString = "Get Element";
    buf = dataString.getBytes();
```

```
    packet = new DatagramPacket(buf, buf.length, address,
port);
    socket.send(packet);
} catch (IOException e) {
    return;
}


while (!done) {
    try {
        buf = new byte[8192];
packet = new DatagramPacket(buf, 8192);
        socket.receive(packet);
    } catch (IOException e) {
return;
    }

    received = null;
    received = new String(packet.getData());
    done = parseInputString(received);
}
       socket.close();
    }

    public Vector getOntology() {
return ontology;
    }

    public void processRemoteAddChild(Ontology element, boolean edit)
{
        DatagramPacket packet;
        byte[] buf;
```

```
String dataString = null;
String received;

dataString = new String("Add Element$" +
element.getID() + "$" +
element.getDef() + "$" +
element.getRef() + "$" +
element.getParent() + "$" +
element.getLevel() + "$" +
element.getNumChildren() + "$");
for (int j=0; j<element.getNumChildren(); j++) {
    dataString = dataString.concat((element.getChildren())[j]
+ "$");
}
dataString = dataString.concat(element.getNumHistory() + "$");
for (int j=0; j<element.getNumHistory(); j++) {
    dataString = dataString.concat((element.getCreator())[j]
+ "$");
    dataString = dataString.concat(((element.getDate())[j]).
toString() + "$");
    dataString = dataString.concat((element.
getDefHistory())[j] + "$");
    dataString = dataString.concat((element.
getRefHistory())[j] + "$");
}

initSocket();
try {                                          // send request
    buf = dataString.getBytes();
    packet = new DatagramPacket(buf, buf.length, address,
port);
    socket.send(packet);
```

```
} catch (IOException e) {
    return;
}

try {
    buf = new byte[8192];
    packet = new DatagramPacket(buf, 8192);
    socket.receive(packet);
} catch (IOException e) {
    return;
}

received = null;
received = new String(packet.getData());

if (!edit) {
    if (received.substring(0,13).equals("Element Added")) {
        new MessageBox("Added Element.", "Click below to
continue.");
    } else {
        new MessageBox("Failed to add remote element.", "Click
below to continue.");
    }
}
socket.close();
    }


    public void processRemoteDeleteElement(Ontology element, boolean
edit) {
        DatagramPacket packet;
        byte[] buf;
```

```
String dataString = null;
String received;

dataString = new String("Del Element$" +
element.getID() + "$" +
element.getDef() + "$" +
element.getRef() + "$" +
element.getParent() + "$" +
element.getLevel() + "$" +
element.getNumChildren() + "$");
for (int j=0; j<element.getNumChildren(); j++) {
    dataString = dataString.concat((element.getChildren())[j]
+ "$");
}
dataString = dataString.concat(element.getNumHistory() + "$");
for (int j=0; j<element.getNumHistory(); j++) {
    dataString = dataString.concat((element.getCreator())[j]
+ "$");
    dataString = dataString.concat(((element.getDate())[j]).
toString() + "$");
    dataString = dataString.concat((element.
getDefHistory())[j] + "$");
    dataString = dataString.concat((element.
getRefHistory())[j] + "$");
}

initSocket();
try {                                          // send request
    buf = dataString.getBytes();
    packet = new DatagramPacket(buf, buf.length, address,
port);
    socket.send(packet);
```

```
} catch (IOException e) {
    return;
}


try {
    buf = new byte[8192];
    packet = new DatagramPacket(buf, 8192);
    socket.receive(packet);
} catch (IOException e) {
    return;
}

received = null;
received = new String(packet.getData());

if (!edit) {
    if (received.substring(0,15).equals("Element Deleted")) {
        new MessageBox("Deleted Element.", "Click below to
continue.");
    } else {
        new MessageBox("Failed to delete remote element.",
"Click below to continue.");
    }
}

       socket.close();
    }


    public boolean parseInputString(String inputString) {
        boolean Done = false;
```

```
        int startIndex = 0;
int endIndex = 0;

String id = null;
String def = null;
String ref = null;
String pa = null;
String[] childNames;
int nc = 0;
int lev = 0;

int nh = 0;
String[] creator;
Date[] date;
String[] defHistory;
String[] refHistory;


        if (inputString.substring(0,7).equals("Element")) {
    startIndex = 8;
    endIndex = inputString.indexOf('$', startIndex);
    id = inputString.substring(startIndex,endIndex);

    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    def = inputString.substring(startIndex,endIndex);

    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    ref = inputString.substring(startIndex,endIndex);

    startIndex = endIndex + 1;
```

```
    endIndex = inputString.indexOf('$', startIndex);
    pa = inputString.substring(startIndex,endIndex);

    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    lev = Integer.parseInt(inputString.substring(startIndex,
endIndex));

    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    nc = Integer.parseInt(inputString.substring(startIndex,
endIndex));

    childNames = new String[nc];

    for (int i=0; i<nc; i++) {
        startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
childNames[i] = inputString.substring(startIndex,
endIndex);
    }

    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    nh = Integer.parseInt(inputString.substring(startIndex,
endIndex));

    creator = new String[nh];
    date = new Date[nh];
    defHistory = new String[nh];
    refHistory = new String[nh];
```

```
    for (int i=0; i<nh; i++) {
        startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
creator[i] = inputString.substring(startIndex,
endIndex);

startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
date[i] = new Date(inputString.substring(startIndex,
endIndex));

        startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
defHistory[i] = inputString.substring(startIndex,
endIndex);

        startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
refHistory[i] = inputString.substring(startIndex,
endIndex);
    }

    if (inputString.substring(endIndex+1, endIndex+5).
equals("more")) {
        Done = false;
    } else {
        Done = true;
    }

    ontology.addElement(new Ontology(id, def, ref, pa,
childNames, nc, lev, nh, creator, date, defHistory, refHistory));
} else {
```

```
    Done = true;
}
return Done;
    }
}
```

B.2.4 Login Tool (Login.java)

```
import java.awt.*;
import java.util.*;
import java.io.*;
import java.lang.*;


public class Login extends Frame{
    MessageBox mbox;
    OntologyApplet onApplet;
    boolean verified = false;

    Panel loginPanel;
    Panel actionPanel;

    TextField ID;
    TextField Passwd;
    String creator = null;
    String passwd = null;

    Login(OntologyApplet newOntologyApplet) {
        onApplet = newOntologyApplet;

setLayout(new BorderLayout(5,5));

add("North", new Label("Ontology Creator Log-In",
```

```
Label.CENTER));

  loginPanel = new Panel();
  loginPanel.setLayout(new GridLayout(2,2,5,5));
  createLoginPanel();

add("Center", loginPanel);

  actionPanel = new Panel();
  actionPanel.setLayout(new GridLayout(1,2,5,5));
  actionPanel.add(new Button("OK"));
  actionPanel.add(new Button("Cancel"));

add("South", actionPanel);

pack();
show();
    }

    private void createLoginPanel() {
        loginPanel.add(new Label("Creator ID"));
ID = new TextField(30);
loginPanel.add(ID);

loginPanel.add(new Label("Password"));
Passwd = new TextField(30);
Passwd.setEchoCharacter('*');
loginPanel.add(Passwd);
    }

    public boolean login(){
        return verified;
```

```
    }

    public String getCreator(){
        return creator;
    }

    public boolean action(Event event, Object arg) {
        if (event.target instanceof Button) {
    if (((String)arg).equals("OK")) {
        creator = ID.getText().trim().toLowerCase();
        passwd = Passwd.getText().trim().toLowerCase();

if (creator.equals("creator")  && passwd.equals
("ontology")){
    verified = true;
    hide();
    onApplet.setCreator("Creator");
} else if (creator.equals("knut")  && passwd.equals
("ontology1")) {
    verified = true;
    hide();
    onApplet.setCreator("Dr. Knut Streitlien");
} else if (creator.equals("pierre")  && passwd.equals
("ontology2")) {
    verified = true;
    hide();
    onApplet.setCreator("Dr. Pierre Elisseeff");
} else if (creator.equals("steve")  && passwd.equals
("ontology3")) {
    verified = true;
    hide();
    onApplet.setCreator("Stephen Abrams");
```

```
} else if (creator.equals("pubudu")  && passwd.equals
("ontology4")) {
    verified = true;
    hide();
    onApplet.setCreator("Pubudu Wariyapola");
} else if (creator.equals("nmp")  && passwd.equals
("ontology5")) {
    verified = true;
    hide();
    onApplet.setCreator("Dr. N. Patrikalakis");
} else {
    resetLoginPanel();
}
    } else {
        dispose();
    }
}
return true;
    }

    private void resetLoginPanel() {
        mbox = new MessageBox("Invalid Creator ID/Password.",
"Please Try Again.");
        loginPanel.removeAll();
        createLoginPanel();
        validate();
    }

    public boolean handleEvent(Event event){
        if (event.id == Event.WINDOW_DESTROY) {
    dispose();
        }
```

```
        return super.handleEvent(event);
    }
}
```

B.2.5 Messaging Window Tool (MessageBox.java)

```
import java.awt.*;
import java.io.*;
import java.lang.*;
import java.util.*;

class MessageBox extends Frame {

    MessageBox(String label1, String label2) {
setLayout(new BorderLayout(5,5));
add("North", new Label(label1, Label.CENTER));
add("Center", new Label(label2, Label.CENTER));

        add("South", new Button("OK"));
resize(300,130);
show();
    }

    MessageBox(String label1, String label2, Ontology element) {
Panel elementPanel;
Panel idPanel;
Panel dataPanel;
Panel childPanel;
Panel defPanel;
Panel addDataPanel;
Panel actionPanel;

TextField ID;
```

```
TextArea Definition;
TextArea Reference;
TextField Level;
TextField Parent;
TextField NumChild;
List Child;

elementPanel = new Panel();
elementPanel.resize(500,400);
elementPanel.setLayout(new BorderLayout(5,5));

idPanel = new Panel();
idPanel.resize(500, 50);
idPanel.setLayout(new GridLayout(1,2,5,5));

dataPanel = new Panel();
dataPanel.resize(500,250);
dataPanel.setLayout(new GridLayout(2,1,5,5));

defPanel = new Panel();
defPanel.resize(500,150);
defPanel.setLayout(new GridLayout(2,2,5,5));

addDataPanel = new Panel();
addDataPanel.resize(500,100);
addDataPanel.setLayout(new GridLayout(3,2,5,5));

childPanel = new Panel();
childPanel.resize(500,100);
childPanel.setLayout(new GridLayout(1,2,5,5));

actionPanel = new Panel();
```

```
actionPanel.resize(500,50);
actionPanel.setLayout(new GridLayout(2,1,5,5));


setLayout(new BorderLayout(5,5));
resize(500,500);
add("North", new Label(label1, Label.CENTER));

int nc = element.getNumChildren();
String[] childNames = element.getChildren();

    idPanel.add(new Label("Element Name"));
    ID = new TextField(element.getID(), 30);
    idPanel.add(ID);
    ID.setEditable(false);

  elementPanel.add("North", idPanel);

      defPanel.add(new Label("Element Definition"));
      Definition = new TextArea(element.getDef(), 5, 60);
      defPanel.add(Definition);
      Definition.setEditable(false);

      defPanel.add(new Label("Reference Source for
Definition"));
      Reference = new TextArea(element.getRef(), 5, 60);
      defPanel.add(Reference);
      Reference.setEditable(false);

    dataPanel.add(defPanel);

      addDataPanel.add(new Label("Element Level"));
```

```
      Level = new TextField(Integer.toString(element.
getLevel()), 5);
      addDataPanel.add(Level);
      Level.setEditable(false);

      addDataPanel.add(new Label("Parent"));
      Parent = new TextField(element.getParent(), 30);
      addDataPanel.add(Parent);
      Parent.setEditable(false);

      addDataPanel.add(new Label("Number of Children"));
      NumChild = new TextField(Integer.toString(nc), 5);
      addDataPanel.add(NumChild);
      NumChild.setEditable(false);

    dataPanel.add(addDataPanel);

  elementPanel.add("Center", dataPanel);

    childPanel.add(new Label("Children", Label.LEFT));

      Child = new List(5, false);
      Child.enable(false);

      if (nc != 0)
          for (int i=0; i<nc; i++)
      Child.addItem(childNames[i]);
      else
          Child.addItem("None");

    childPanel.add(Child);
```

```
  elementPanel.add("South", childPanel);

add("Center", elementPanel);

  actionPanel.add(new Label(label2, Label.CENTER));
  actionPanel.add(new Button("OK"));

add("South", actionPanel);

show();
    }

    MessageBox(String label1, Ontology element) {
setLayout(new BorderLayout(5,5));

Panel historyPanel;

historyPanel = new Panel();
historyPanel.setLayout(new GridLayout(element.
getNumHistory()+1,3,5,5));

add("North", new Label(label1, Label.CENTER));

int nh = element.getNumHistory();
String[] creator = element.getCreator();
Date[] date = element.getDate();
String[] defHistory = element.getDefHistory();
String[] refHistory = element.getRefHistory();

if (nh != 0){
    historyPanel.add(new Label("Creator", Label.CENTER));
    historyPanel.add(new Label("Date", Label.CENTER));
```

```
    historyPanel.add(new Label("Definition", Label.CENTER));
    historyPanel.add(new Label("Reference", Label.CENTER));

    for (int i=0; i<nh; i++) {
        historyPanel.add(new Label(creator[i]));
historyPanel.add(new Label(date[i].toString()));
historyPanel.add(new TextArea(defHistory[i], 3, 30));
historyPanel.add(new TextArea(refHistory[i], 3, 30));
    }
    add("Center", historyPanel);
} else {
    add("Center", new Label("No Creation History",
Label.CENTER));
}

add("South", new Button("OK"));

pack();
show();
    }

    public boolean action(Event event, Object arg) {
        if (event.target instanceof Button)
    if (((String)arg).equals("OK"))
        hide();
return true;
    }


    public boolean handleEvent(Event event){
        if (event.id == Event.WINDOW_DESTROY) {
  dispose();
```

```
        }
        return super.handleEvent(event);
    }
}
```

B.3 Ontology Creator Server Daemon Code

B.3.1 Server Startup Code (DataServer.java)

```
import java.io.*;
import java.net.*;
import java.util.*;


class DataServer {
  public static void main(String[] args) {
    new DataServerThread().start();
  }
}
```

B.3.2 Server Thread Generator (DataServerThread.java)

```
import java.io.*;
import java.awt.*;
import java.util.*;
import java.applet.*;
import java.net.*;
import java.lang.*;


class DataServerThread extends Thread {
    private DatagramSocket socket = null;
    DatagramPacket packet;
    InetAddress address;
    int listenport=8020;
    int port;
    byte[] buf;
    String dataString;
```

```
    OntologyVector onvect;
    OntologyFileIO onfile;
    Ontology element;
    Ontology parentElement;

    DataServerThread() {
        super("DataServer");

        try {
            address = InetAddress.getByName("deslab.mit.edu");
        } catch (UnknownHostException e) {}

        try {
    socket = new DatagramSocket(listenport, address);
        } catch (java.io.IOException e) {}

onvect = new OntologyVector();
onfile = new OntologyFileIO();

onvect.storeOntology(onfile.createOntology());
    }

    public void run() {
        if (socket == null)
            return;

        while (true) {
            try {    // receive request
        buf = new byte[8192];
                packet = new DatagramPacket(buf, 8192);

                socket.receive(packet);
```

```
                address = packet.getAddress();
                port = packet.getPort();

dataString = null;
dataString = new String(packet.getData());
    } catch (IOException e) {}

    if (dataString.substring(0,11).equals("Add Element")) {
parseInputString(dataString);
addLocalElement();
    }
    }
    if (dataString.substring(0,11).equals("Del Element")) {
parseInputString(dataString);
deleteLocalElement();
    }
    if (dataString.substring(0,11).equals("Get Element")) {
sendOntology();
    }
}
    }

    public void parseInputString(String inputString) {
        int startIndex = 0;
int endIndex = 0;

String id = null;
String def = null;
String ref = null;
String pa = null;
String[] childNames;
```

```
int nc = 0;
int lev = 0;

int nh = 0;
String[] creator;
Date[] date;
String[] defHistory;
String[] refHistory;

startIndex = 12;
endIndex = inputString.indexOf('$', startIndex);
id = inputString.substring(startIndex,endIndex);

startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
def = inputString.substring(startIndex,endIndex);

startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
ref = inputString.substring(startIndex,endIndex);

startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
pa = inputString.substring(startIndex,endIndex);

startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
lev = Integer.parseInt(inputString.substring(startIndex,
endIndex));

startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
```

```
nc = Integer.parseInt(inputString.substring(startIndex,
endIndex));

childNames = new String[nc];

for (int i=0; i<nc; i++) {
    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    childNames[i] = inputString.substring(startIndex,
endIndex);
}

startIndex = endIndex + 1;
endIndex = inputString.indexOf('$', startIndex);
nh = Integer.parseInt(inputString.substring(startIndex,
endIndex));

creator = new String[nh];
date = new Date[nh];
defHistory = new String[nh];
refHistory = new String[nh];

for (int i=0; i<nh; i++) {
    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    creator[i] = inputString.substring(startIndex,endIndex);

    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    date[i] = new Date(inputString.substring(startIndex,
endIndex));
```

```
    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    defHistory[i] = inputString.substring(startIndex,
endIndex);

    startIndex = endIndex + 1;
    endIndex = inputString.indexOf('$', startIndex);
    refHistory[i] = inputString.substring(startIndex,
endIndex);
}

element = new Ontology(id, def, ref, pa, childNames, nc, lev,
nh, creator, date, defHistory, refHistory);
    }

    public void sendOntology() {                    // send response
        Vector ontology = onvect.getOntology();

        for (int i=0; i<ontology.size(); i++) {
    element = (Ontology)ontology.elementAt(i);
    dataString = null;

    dataString = new String("Element$" +
    element.getID() + "$" +
    element.getDef() + "$" +
    element.getRef() + "$" +
    element.getParent() + "$" +
    element.getLevel() + "$" +
    element.getNumChildren() + "$");
    for (int j=0; j<element.getNumChildren(); j++) {
        dataString = dataString.concat((element.
getChildren())[j] + "$");
```

```
    }
    dataString = dataString.concat(element.getNumHistory()
+ "$");
    for (int j=0; j<element.getNumHistory(); j++) {
        dataString = dataString.concat((element.
getCreator())[j] + "$");
        dataString = dataString.concat(((element.
getDate())[j]).toString() + "$");
        dataString = dataString.concat((element.
getDefHistory())[j] + "$");
        dataString = dataString.concat((element.
getRefHistory())[j] + "$");
    }

    if (i == ontology.size() - 1) {
        dataString = dataString.concat("done");
    } else {
        dataString = dataString.concat("more");
    }

            try {
        buf = dataString.getBytes();
packet = new DatagramPacket(buf, buf.length, address,
port);
socket.send(packet);
    } catch (IOException e) {}
}
    }


    public void addLocalElement() {
        String id = element.getID();
```

```
String pa = element.getParent();

parentElement = onvect.findElement(pa);

onvect.addVectorElement(element);
onfile.writeSingleElement(element, onvect);

       onvect.deleteVectorElement(parentElement);
onfile.deleteSingleElement(parentElement, onvect);

parentElement.addChild(id);
        onvect.addVectorElement(parentElement);
onfile.writeSingleElement(parentElement, onvect);

try {
    dataString = new String("Element Added");
    buf = dataString.getBytes();
    packet = new DatagramPacket(buf, buf.length, address,
port);
    socket.send(packet);
} catch (IOException e) {}
    }


    public void deleteLocalElement() {
        String id = element.getID();
String pa = element.getParent();

element = onvect.findElement(id, pa);
parentElement = onvect.findParent(element);

       onvect.deleteVectorElement(element);
```

```
onfile.deleteSingleElement(element, onvect);

element = parentElement;
       onvect.deleteVectorElement(parentElement);
onfile.deleteSingleElement(parentElement, onvect);

element.deleteChild(id);
        onvect.addVectorElement(parentElement);
onfile.writeSingleElement(parentElement, onvect);



try {
    dataString = new String("Element Deleted");
    buf = dataString.getBytes();
    packet = new DatagramPacket(buf, buf.length, address,
port);
    socket.send(packet);
} catch (IOException e) {}
    }



    protected void finalize() {
        if (socket != null) {
    socket.close();
    socket = null;
        }
    }
}
```

B.3.3 File Interface Tool (OntologyFileIO.java)

```
import java.io.*;
import java.util.*;
```

```
import java.lang.*;


class OntologyFileIO {
    Vector ontology;
    ArraySort asort;

    OntologyFileIO () {
ontology = new Vector();
asort = new ArraySort();
    }


    public Vector createOntology() {
String inputFileName = "PoseidonOntology";

readOntology(inputFileName);
return ontology;
    }


    public void readOntology(String inputFileName){
String newInputFileName = "";
        String id = null;
String def = null;
String ref = null;
String pa = null;
Vector child = new Vector();
String[] childNames;
int nc = 0;
int lev = 0;
```

```
int nh = 0;
String[] creator = null;
Date[] date = null;
String[] defHistory = null;
String[] refHistory = null;

String tempInputFileName;
StringBuffer newInputFileNameBuf;
char ch;

try {
    File inputFile = new File(inputFileName + ".txt");
    BufferedReader dbf = new BufferedReader(new
FileReader(inputFile));

    id = dbf.readLine();
    def = dbf.readLine();
    ref = dbf.readLine();
    pa = dbf.readLine();
    lev = (new Integer(dbf.readLine())).intValue();
    nc = (new Integer(dbf.readLine())).intValue();

    childNames = new String[nc];

    for(int i=0; i< nc; i++){
        childNames[i] = dbf.readLine();
child.addElement(childNames[i]);
    }

    childNames = asort.sortArray(childNames);

    nh = (new Integer(dbf.readLine())).intValue();
```

```
    creator = new String[nh];
    date = new Date[nh];
    defHistory = new String[nh];
    refHistory = new String[nh];

    for(int i=0; i< nh; i++){
        creator[i] = dbf.readLine();
date[i] = new Date(dbf.readLine());
        defHistory[i] = dbf.readLine();
refHistory[i] = dbf.readLine();
    }

    ontology.addElement(new Ontology(id, def, ref, pa,
childNames, nc, lev, nh, creator, date, defHistory, refHistory));

} catch (NumberFormatException e) {
} catch (FileNotFoundException e) {
} catch (IOException e) {
}

for ( int i=0; i<nc; i++) {
    tempInputFileName = (String)child.elementAt(i);
    newInputFileNameBuf = new StringBuffer();

    for (int j=0; j<tempInputFileName.length(); j++) {
        ch = tempInputFileName.charAt(j);
        if (!Character.isSpace(ch))
    newInputFileNameBuf.append(ch);
    }

    newInputFileName = inputFileName + "/" +
```

```
newInputFileNameBuf.toString();
    readOntology(newInputFileName);
}
    }


    public void writeSingleElement(Ontology element, OntologyVector
onvect){
        String id = element.getID();
String def = element.getDef();
String ref = element.getRef();
String pa = element.getParent();
int lev = element.getLevel();
int nc = element.getNumChildren();
String[] child = element.getChildren();

int nh = element.getNumHistory();
String[] creator = element.getCreator();
Date[] date = element.getDate() ;
String[] defHistory = element.getDefHistory();
String[] refHistory = element.getRefHistory();

StringBuffer outputDirNameBuf;
StringBuffer outputFileNameBuf;
char ch;

try {
    String outputDirName = new String();
    String outputFileName = new String();

    if (lev == 0)
        outputDirName = ".";
```

```
    if (lev == 1)
        outputDirName = pa;
    if (lev == 2){
        String grandpa = onvect.findElement(pa).getParent();
outputDirName = grandpa + "/" + pa;
    }

    outputDirNameBuf = new StringBuffer();
    for (int j=0; j<outputDirName.length(); j++) {
        ch = outputDirName.charAt(j);
if (!Character.isSpace(ch))
    outputDirNameBuf.append(ch);
    }
    outputDirName = outputDirNameBuf.toString();

    outputFileName = id;
    outputFileNameBuf = new StringBuffer();
    for (int j=0; j<outputFileName.length(); j++) {
        ch = outputFileName.charAt(j);
if (!Character.isSpace(ch))
    outputFileNameBuf.append(ch);
    }
    outputFileName = outputFileNameBuf.toString();


    File outputDir = new File(outputDirName);
    File outputFile = new File(outputDirName, outputFileName +
".txt");

    outputDir.mkdirs();
    PrintWriter obf = new PrintWriter(new
FileWriter(outputFile));
```

```
    obf.println(id);
    obf.println(def);
    obf.println(ref);
    obf.println(pa);
    obf.println(lev);
    obf.println(nc);

    for(int j=0; j<nc; j++){
        obf.println(child[j]);
    }

    obf.println(nh);

    for(int j=0; j<nh; j++){
        obf.println(creator[j]);
        obf.println(date[j].toString());
        obf.println(defHistory[j]);
obf.println(refHistory[j]);
    }
    obf.close();
} catch (IOException e) {}
    }


    public void deleteSingleElement(Ontology element, OntologyVector
onvect){

        String id = element.getID();
String pa = element.getParent();
int lev = element.getLevel();
```

```
String outputDirName = new String();
String outputFileName = new String();

StringBuffer outputDirNameBuf;
StringBuffer outputFileNameBuf;
char ch;

if (lev == 0)
    outputDirName = ".";
if (lev == 1)
    outputDirName = pa;
if (lev == 2){
    String grandpa = onvect.findElement(pa).getParent();
    outputDirName = grandpa + "/" + pa;
}

outputDirNameBuf = new StringBuffer();
for (int j=0; j<outputDirName.length(); j++) {
    ch = outputDirName.charAt(j);
    if (!Character.isSpace(ch))
        outputDirNameBuf.append(ch);
}
outputDirName = outputDirNameBuf.toString();

outputFileName = id;
outputFileNameBuf = new StringBuffer();
for (int j=0; j<outputFileName.length(); j++) {
    ch = outputFileName.charAt(j);
    if (!Character.isSpace(ch))
        outputFileNameBuf.append(ch);
}
outputFileName = outputFileNameBuf.toString();
```

```
if (lev == 1) {
    File outputDir = new File(outputDirName + "/" +
outputFileName);
    try {
        outputDir.delete();
    } catch (SecurityException e){}
}

File outputFile = new File(outputDirName, outputFileName +
".txt");
outputFile.delete();
    }
}
```

B.4 Ontology Creator Utility Code

B.4.1 Ontology Element Object (Ontology.java)

```
import java.util.*;
import java.io.*;
import java.lang.*;


class Ontology extends Object {
    String OntologyID = null;
    String OntologyDef = null;
    String OntologyRef = null;
    String OntologyParent = null;
    int OntologyLevel = 2;

    int OntologyNumChildren = 0;
    String[] OntologyChild = null;
```

```
    int OntologyNumHistory = 0;
    Date[] OntologyDate = null;
    String[] OntologyCreator = null;
    String[] DefHistory = null;
    String[] RefHistory = null;

    Ontology(){}

    Ontology(String id, String def, String ref, String pa, String[]
child, int nc, int lev, int nh, String[] creator, Date[] date,
String[] defHistory, String[] refHistory){
OntologyID = id;
OntologyDef = def;
OntologyRef = ref;
OntologyParent = pa;
OntologyChild = child;
OntologyNumChildren = nc;
OntologyLevel = lev;
OntologyNumHistory = nh;
OntologyDate = date;
OntologyCreator = creator;
DefHistory = defHistory;
RefHistory = refHistory;
    }

    Ontology(String id, String def, String ref, String pa, String[]
child, int nc, int lev){
OntologyID = id;
OntologyDef = def;
OntologyRef = ref;
OntologyParent = pa;
```

```
OntologyChild = child;
OntologyNumChildren = nc;
OntologyLevel = lev;
    }

    Ontology(String id, String def, String pa, String[] child, int nc,
int lev){
OntologyID = id;
OntologyDef = def;
OntologyParent = pa;
OntologyChild = child;
OntologyNumChildren = nc;
OntologyLevel = lev;
    }

    Ontology(String pa, int lev){
OntologyParent = pa;
OntologyLevel = lev;
    }

    public void printElement() {
        System.out.println("Identifier : " + OntologyID);
        System.out.println("Definition : " + OntologyDef);
        System.out.println("Reference  : " + OntologyRef);
        System.out.println("Parent     : " + OntologyParent);
        System.out.println("Level      : " + OntologyLevel);
        System.out.println("Num Child  : " + OntologyNumChildren);

for(int i=0; i< OntologyNumChildren; i++){
            System.out.println("Child " + i + "    : " +
OntologyChild[i]);
}
```

```
System.out.println("Creation History");
        System.out.println("Num History: " + OntologyNumHistory);
for(int i=0; i<OntologyNumHistory; i++){
     System.out.println(OntologyCreator[i]);
             System.out.println(OntologyDate[i].toString());
}
    }

    public String getID(){
        return this.OntologyID;
    }

    public String getDef(){
        return this.OntologyDef;
    }

    public String getRef(){
        return this.OntologyRef;
    }

    public String getParent(){
        return this.OntologyParent;
    }

    public int getLevel(){
        return this.OntologyLevel;
    }

    public int getNumChildren(){
        return this.OntologyNumChildren;
    }
```

```
public String[] getChildren(){
    return this.OntologyChild;
}

public int getNumHistory(){
    return this.OntologyNumHistory;
}

public Date[] getDate(){
    return this.OntologyDate;
}

public String[] getCreator(){
    return this.OntologyCreator;
}

public String[] getDefHistory(){
    return this.DefHistory;
}

public String[] getRefHistory(){
    return this.RefHistory;
}


public void addChild(String childID) {
    ArraySort asort = new ArraySort();

    if (OntologyNumChildren == 0 ) {
OntologyChild = new String[1];
OntologyChild[0] = childID;
```

```
    OntologyNumChildren = 1;
} else {
    String[] tempChild = OntologyChild;
    OntologyChild = new String[OntologyNumChildren + 1];
    System.arraycopy(tempChild, 0, OntologyChild, 0,
OntologyNumChildren);
    OntologyChild[OntologyNumChildren] = childID;
    OntologyChild = asort.sortArray(OntologyChild);
    OntologyNumChildren++;
}
    }

    public void deleteChild(String childID) {
        int index = 0;

        if (OntologyNumChildren == 1 ) {
    OntologyChild = null;
    OntologyNumChildren = 0;
} else {
    String[] tempChild = OntologyChild;
    OntologyChild = new String[OntologyNumChildren - 1];

    for (int i=0; i<OntologyNumChildren; i++)
        if (tempChild[i].equals(childID))
    index = i;

    System.arraycopy(tempChild, 0, OntologyChild, 0, index);
    System.arraycopy(tempChild, index+1, OntologyChild, index,
OntologyNumChildren - (index + 1));
    OntologyNumChildren--;
}
    }
```

```
}
```

B.4.2 Ontology Vector Object (OntologyVector.java)

```
import java.io.*;
import java.util.*;
import java.lang.*;

class OntologyVector {
    Vector ontology;


    public OntologyVector() {
        ontology = new Vector();
    }


    public void storeOntology(Vector newVector){
        ontology = newVector;
    }


    public Ontology findElement(String id) {
        Ontology element = new Ontology();

        for (int i=0; i<ontology.size(); i++) {
    if(((Ontology)ontology.elementAt(i)).getID().equals(id)) {
                element = (Ontology)ontology.elementAt(i);
    }
}
return element;
    }
```

```
    public Ontology findElement(String id, String pa) {
        Ontology element = new Ontology();

        for (int i=0; i<ontology.size(); i++) {
    if((((Ontology)ontology.elementAt(i)).getID().equals(id))
&& (((Ontology)ontology.elementAt(i)).getParent().equals(pa))) {
                element = (Ontology)ontology.elementAt(i);
    }
}
return element;
    }


    public Ontology findParent(Ontology element){
        Ontology parent = new Ontology();
String[] childNames;

        for (int i=0; i<ontology.size(); i++) {
    if(((Ontology)ontology.elementAt(i)).getID().
equals(element.getParent())) {
        childNames = ((Ontology)ontology.elementAt(i)).
getChildren();
        for (int j=0; j<((Ontology)ontology.elementAt(i)).
getNumChildren(); j++) {
    if (childNames[j].equals(element.getID()));
        parent = (Ontology)ontology.elementAt(i);
}
    }
}
return parent;
    }
```

```
    public Vector findChildren(Ontology element){
        Vector childrenVector = new Vector(element.getNumChildren());
String[] childNames = element.getChildren();

        for (int i=0; i<element.getNumChildren(); i++)
    for(int j=0; j<ontology.size(); j++)
        if ((childNames[i].equals(((Ontology)ontology.
elementAt(j)).getID())) && (element.getID().equals(((Ontology)
ontology.elementAt(j)).getParent())))
    childrenVector.addElement((Ontology)ontology.
elementAt(j));

childrenVector.trimToSize();
return childrenVector;
    }


    public void addVectorElement(Ontology element) {
        ontology.addElement(element);
    }


    public void deleteVectorElement(Ontology element) {
ontology.removeElement(element);
    }


    public Vector getOntology() {
        return ontology;
    }
```

```
}
```

B.4.3 String Array Sorting Utility (ArraySort.java)

```
import java.util.*;
import java.io.*;
import java.lang.*;


public class ArraySort{

    ArraySort () {}

    public String[] sortArray(String[] array) {
        String temp;

        for (int i=0; i<array.length; i++) {
    for (int j=i+1; j<array.length; j++) {
        if (array[i].compareTo(array[j]) >= 0){
    temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}
    }
}
        return array;
    }
}
```

B.5 Ontology Element File (Acoustical.txt)

```
Acoustical
The study of sound propagation in the ocean and its underlying
```

sediments. This ranges from the earliest use of depth soundings to chart the ocean floor to the use of SONAR to locate schools of fish, underwater vehicles and ocean drifters to the most recent applications of acoustic tomography to infer large-scale properties of the ocean and the ocean floor.

http://www-ocean.tamu.edu/~baum/paleo/paleogloss

Poseidon Ontology

1

21

array gain

decibel

hydrophone

limiting ray

noise limited condition

object backscattering differential target strength

refraction

relative reverberation level

reverberation limited condition

sea noise

shaded transducer

sonar background noise

sonar dome

sonar dome insertion loss

sonar dome loss directivity pattern

sonar equations

sonar self noise

sonar source level axial source level

surface layer

underwater radiated noise

underwater sound projector

2

Creator

Wed Jul 22 16:18:04 EDT 1998

Describes Sound.

Creator

Tue Nov 03 16:23:18 EST 1998

The study of sound propagation in the ocean and its underlying sediments. This ranges from the earliest use of depth soundings to chart the ocean floor to the use of SONAR to locate schools of fish, underwater vehicles and ocean drifters to the most recent applications of acoustic tomography to infer large-scale properties of the ocean and the ocean floor.

http://www-ocean.tamu.edu/~baum/paleo/paleogloss

Bibliography

[1] American Library Association, ALCTS/LITA/RUSA, Machine-Readable Bibliographic Information Committee. The USMARC formats: Background and principles. Library of Congress, Network Development and MARC Standards Office, Washington, DC, November 1996. <http://lcweb.loc.gov/marc/ 96principl.html>.

[2] ARC/INFO Document.aml. ESRI Software, Redlands, CA, July 14, 1998. <http://www.esri.com/software/arcinfo/docaml721.html>.

[3] T. Baker. Languages for Dublin Core. *D-Lib Magazine*, December 1998. <http://www.dlib.org/dlib/december98/12baker.html>.

[4] Biological data profile of the content standard for digital geospatial metadata. Federal Geographic Data Committee Standards Working Group and USGS Biological Resources Division, August 31, 1998. <http://www.fgdc.gov/standards/documents/standards/biodata/bioprofi.pdf>.

[5] T. Bray, J. Paoli and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. World Wide Web Consortium Recommendation, February 10, 1998 <http://www.w3.org/TR/REC-xml>.

[6] M. Bunge. *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World.* Reidel, Boston, 1977.

[7] M. Bunge. *Treatise on Basic Philosophy: Vol. 3: Ontology II: A World of Systems.* Reidel, Boston, 1979.

[8] H. Casanova, J. J. Dongarra, and K. Moore. Network-enabled solvers and the NetSolve project. *SIAM News*, January 1998. <http://www.siam.org/siamnews/01-98/janfeb.htm>.

[9] D. E. Comer and L. Stevens. *Internetworking with TCP/IP Volume 2: Design, Implementation, and Internals.* Prentice Hall, 1991.

[10] D. E. Comer. *Internetworking with TCP/IP: Principles, Protocols, and Architectures.* Prentice Hall, 1995.

[11] Coordinating geographic data acquisition and access: The National Spatial Data Infrastructure. Executive Order 12906, *Federal Register*, April 13, 1994. <http://www.fgdc.gov/publications/ documents/geninfo/execord.html>.

[12] V. Christophides, C. Houstis, S. Lalis, H. Tsalapata. Ontology-driven integration of scientific repositories. NGITS '99, New Generation Information Technologies, Lecture Notes in Computer Science, Elsevier, Hobart Habaron, Isreal, July 1999.

[13] CPAN: Comprehensive Perl Archive Network. <ftp://ftp.funet.fi/pub/languages/perl/CPAN/README.html>

[14] J. R. Davis. Creating a networked computer science technical report library. *D-Lib Magazine*, September 1995. <http://www.dlib.org/dlib/september95/ 09davis.html>

[15] G. Deconinck, J. Vounckx, R. Cuyvers, R, Lauwereins. Survey of Checkpointing and Rollback Techniques. Technical Report f93-04, Katholieke Universiteit Leuven, Belgium, June 1993 <ftp://gate.esat.kuleuven.ac.be/pub/ACCA/FTMPS/REPORTS/f93-04.ps>.

[16] T. Dopplick. The role of metadata in EOSDIS. Technical Report 160-TP-013-001, Hughes Information Technology Systems, Upper Marlboro, MD, March 1997. <http://edhs1.gsfc.nasa.gov/waisdata/sdp/pdf/tp1601301.pdf>.

[17] Dublin Core Metadata Initiative. <http://purl.oclc.org/dc/>.

[18] D. L. Eager, E. D. Lazowska and J. Zahorjan. Dynamic Load Sharing in Homogeneous Distributed Systems. *IEEE Transactions on Software Engineering*, Volume 12, Number 5, pp 662-675, May 1986.

[19] Federal Geographic Metadata Committee. Content standards for digital geospatial metadata, June 1998 <http://www.fgdc.gov/metadata/contstan.html>.

[20] R. Fikes, A. Farquhar. Distributed repositories of highly expressive reusable ontologies. *Stanford University*, March 1998. <http://ontolingua.stanford.edu/>.

[21] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. Integrating and accessing heterogeneous information sources in TSIMMIS. In *Proceedings of the AAAI Symposium on Information Gathering*, pages 61–64, March 1995. <ftp://db.stanford.edu/pub/papers/tsimmis-abstract-aaai.ps>.

[22] Global change master directory. NASA. <http://gcmd.gsfc.nasa.gov/>.

[23] Object Management Group. <http://www.omg.org/>.

[24] J. Hakala. The Nordic metadata project: Final report, July 1998. <http://linnea.helsinki.fi/meta/nmfinal.htm>.

[25] C. Houstis, C. Nikolaou, M. Marazakis, N. M. Patrikalakis, J. Sairamesh, and A. Thomasic. THETIS: Design of a data management and data visualization system for coastal zone management of the Mediterranean sea. *D-lib Magazine*, November 1997. <http://www.dlib.org/dlib/november97/thetis/ 11thetis.html>.

[26] C. Houstis, C. Nikolaou, S. Lalis, S. Kapidakis, V. Christophides, E. Simon, and A. Thomasic. Towards a next generation of open scientific data repositories and services. *CWI Quarterly (Centrum voor Wiskunde en Informatica)*, 12(2), 1999. To appear. <http://dienst.csi.forth.gr:80/Dienst/UI/2.0/Describe/ercim.forth.ics/TR98-0219>.

[27] Java technology home page, June 16, 1998. <http://www.javasoft.com/>.

[28] C. Lagoze. The Warwick Framework: A container architecture for aggregating sets of metadata. *D-Lib Magazine*, July/August 1996. <http://www.dlib.org/dlib/july96/lagoze/07lagoze.html>.

[29] C. Lagoze and J. Davis. Dienst: an architecture for distributed document libraries. *Communications of the ACM*, Volume 38, Number 4, pp 47-48, April 1995.

[30] S. Lalis, C. Houstis, and V. Christophides. Exploring knowledge for the interactive specification of scientific workflows. Technical Report FORTH-ICS/TR-229, Institute of Computer Science, Foundation for Research and Technology – Hellas, Her-

aklion, Crete, Greece, September 1998. <http://dienst.csi.forth.gr:80/Dienst/UI/2.0/Describe/ercim.forth.ics/TR98-0229>.

[31] R. Lasher and D. Cohen. A format for bibliographic records. RFC 1807, June 1995. <file://nic.merit.edu/documents/rfc/rfc1807.txt>.

[32] O. Lassila. Introduction to RDF metadata. W3C Note 1997-11-13, Cambridge, MA, November 1997. <http://www.w3.org/TR/NOTE-rdf-simple-intro-971113.html>.

[33] W. E. Leland and T. J. Ott. Load-Balancing Heuristics and Process Behavior. *Joint Conference on Computer Performance Modelling, Measurement and Evaluation*, Greensborough, NC, May 1986

[34] B. M. Leiner. The NCSTRL approach to open architecture for the confederated digital library. *D-Lib Magazine*, December 1998. <http://www.dlib.org/dlib/december98/leiner/12leiner.html>.

[35] Master Environmental Library. <http://mel.dmso.mil/>.

[36] Master Environmental Library (MEL), technical reference guide, characteristics and performance, version 1.1. Defense Modeling and Simulation Office, Washington, DC, December 17, 1998. <http://mel.dmso.mil/docs/trg/trg.html>.

[37] Metadata form (the easy way). Texas/ Mexico Borderlands Data and Information Center, Austin, TX, October 17, 1997. <http://www.bic.state.tx.us/BICenglish/explanation.htm>.

[38] E. Miller. An introduction to the Resource Description Framework. *D-Lib Magazine*, May 1998. <http://www.dlib.org/dlib/may98/miller/05miller.html>.

[39] P. Niemeyer and J. Peck. *Exploring JAVA*. O'Reilly & Associates Inc., September 1997.

[40] C. Nikolaou, C. Houstis, J. Sairamesh, and N. M. Patrikalakis. Impact of scientific advanced networks for the transfer of knowledge and technology in the field of coastal zones. In *Euro-Mediterranean Workshop on Coastal Zone Management, Alexandria, Egypt, November 1996*, 1996. <http://dienst.csi.forth.gr:80/Dienst/UI/2.0/Describe/ercim.forth.ics/TR97-0188>.

[41] N. M. Patrikalakis, editor. *Proceedings of the NSF Workshop on Distributed Information, Computation and Process Management for Scientific and Engineering Environments (DICPM), Herdon, Virginia, May 15-16, 1998*, November 1998. <http://deslab.mit.edu/DesignLab/dicpm/>.

[42] N. M. Patrikalakis, P. J. Fortier, Y. Ioannidis, C. N. Nikolaou, A. R. Robinson, J. R. Rossignac, A. Vinacua, and S. L. Abrams. Distributed Information and Computation in Scientific and Engineering Environments. *D-Lib Magazine*, April 1999. <http://www.dlib.org/dlib/april99/04abrams.html>.

[43] N. M. Patrikalakis, C. Chryssostomidis, and K. Mihanetzis. Design and Manufacturing in a Distributed Computer Environment In *Proceedings of the 10th International Conference on Computer Applications in Shipbuilding, ICCAS '99* MIT, Cambridge, MA, June 1999. <http://czms.mit.edu/poseidon/publication/>.

[44] K. P. Mihanetzis. Towards a Distributed Information System for Coastal Zone Management. *MIT, Joint Ocean Engineer and M.S. in Ocean Systems Management thesis.*, May 1999. <http://czms.mit.edu/poseidon/publication/>.

[45] H. Phillips. Metadata tools for geospatial data, October 24, 1998. <http://badger.state.wi.us/agencies/wlib/sco/metatool/mtools.htm>.

[46] Poseidon: A Distributed Information System for Ocean Processes. <http://czms.mit.edu/poseidon/>

[47] D. K. Pradhan and N. H. Vaidya. Roll-Forward and Rollback Recovery: Performance-Reliability Trade-Off. In *Proceedings of 24th International Symposium on Fault Tolerant Computing*, Austin, TX, June 1994

[48] B. Randell. System Structure for Software Fault-tolerance. *IEEE Transactions on Software Engineering*, Volume 1, Number 2, pp 221-232, June 1975.

[49] M. Roszkowski and C. Lukas. A distributed architecture for resource discovery using metadata. *D-Lib Magazine*, June 1998. <http://www.dlib.org/dlib/june98/scout/06roszkowski.html>.

[50] M. Satyanarayanan. Scalable, Secure, and Highly Available Distributed File Access. *IEEE Computer*, Volume 23, Number 5, pp 9 - 21, May 1990.

[51] P. Schweitzer. Format metadata – info and tools. United States Geological Survey, November 6, 1998. <http://geology.usgs.gov/tools/metadata/>.

[52] THETIS: A data management and data visualization system for supporting coastal zone management for the Mediterranean Sea. Institute of Computer Science, Foundation for Research and Technology – Hellas, Heraklion, Crete, Greece. <http://www.ics.forth.gr/ pleiades/THETIS/thetis.html>.

[53] The Stanford-IBM Manager of Multiple Information Sources TSIMMIS. <ftp://db.stanford.edu/www/tsimmis/tsimmis.html>.

[54] Unidata: Software and Weather Data for Universities. <http://unidata.ucar.edu/>.

[55] P. C. H. Wariyapola, N. M. Patrikalakis, S. L. Abrams, P. Elisseeff, A. R. Robinson, H. Schmidt, and K. Streitlien. Ontology and Metadata Creation for the Poseidon Distributed Coastal Zone Management System. *Proceedings of IEEE Forum on Research and Technology Advances in Digital Libraries, IEEE ADL'99.* Baltimore, Maryland, USA. pp. 180-189. May 1999. Los Alamitos, CA: IEEE, 1999.

[56] S. Weibel. Metadata: The foundation of resource description. *D-Lib Magazine*, July 1995. <http://www.dlib.org/dlib/July95/07weibel.html>.

[57] S. Weibel and J. Hakala. DC-5: The Helsinki metadata workshop: A report on the workshop and subsequent developments. *D-Lib Magazine*, February 1998. <http://www.dlib.org/dlib/february98/02weibel.html>.