Enhancing a Domain-Specific Digital Library with Metadata Based on Hierarchical Controlled Vocabularies

Mathew Jon Weaver

Bachelor of Science, Computer Science, Brigham Young University (1999)

Master of Science, Computer Science and Engineering, Oregon Graduate Institute (2003)

A dissertation presented to the faculty of the
OGI School of Science & Engineering
at Oregon Health & Science University
in partial fulfillment of the
requirements for the degree
Doctor of Philosophy
in
Computer Science and Engineering

August 2005

The dissertation "Enhancing a Domain-Specific Digital Library with Metadata Based on Hierarchical Controlled Vocabularies" by Mathew Jon Weaver has been examined and approved by the following Examination Committee:

_____
Dr. Lois Delcambre, Thesis Advisor
Professor, Portland State University

_____
Dr. Lillian Cassel
Professor, Villanova University

_____
Dr. Richard Fairley
Professor, OGI School of Science & Engineering,
Oregon Health and Science University

_____
Dr. David Maier
Professor, Portland State University

_____
Dr. Leonard Shapiro
Professor, Portland State University

Acknowledgements

I would like to thank Dr. Lois Delcambre, my thesis advisor, for her advice, support, and assistance over the years. She provided valuable input and helped me discover and refine ideas throughout the research process. I am grateful for her continual encouragement and dedication. I would also like to acknowledge Timothy Tolle, Marianne Lykke Nielsen, and the entire Forest Project research team for their many contributions and support. In addition, I express gratitude to the United States National Science Foundation (Award #9983518) and the United States Department of Defense (National Defense Science and Engineering Graduate Fellowship, 2001) for financial support.

Foremost, I would like to express my appreciation and love for my dear wife, Katie Ann Weaver. She is an invaluable part of my life and has been a great support to me throughout my education. I am truly grateful for her patience, encouragement, and love.

Table of Contents

List of Tables

List of Figures

Abstract

Enhancing a Domain-Specific Digital Library
with Metadata Based on Hierarchical Controlled Vocabularies

Mathew Jon Weaver, B.S., M.S.

Ph.D., OGI School of Science & Engineering
at Oregon Health & Science University

August 2005

Thesis Advisor: Dr. Lois Delcambre

Natural resource managers make decisions that affect numerous organizations, individuals, and the environment. Such decisions are based on a broad range of information gleaned from a variety of reports such as *Decision Notices*, *Environmental Analyses*, and *Environmental Impact Statements* – as well as various specialist reports that provide detailed scientific findings and evaluation. Many of these documents are authored by a multi-disciplinary team of experts who routinely use a wide range of terminology. The documents often contain diverse content including text, maps, scientific data, and images. This conglomeration of terminology and heterogeneous documents presents an interesting information retrieval challenge.

Our primary objective is to design, construct, and evaluate a domain-specific digital library, called Metadata++, with a focus on natural resource managers. The digital library emphasizes specialized terminology, including terms from a large number of well-established, well-known classification schemes and terminologies used by multi-disciplinary experts such as soil scientists, fish biologists, wildlife biologists, fire specialists, and hydrologists. These specialists frequently use the same terms with often subtle (and occasionally significant) differences in meaning. This dissertation presents a path-based thesaurus model that supports polyhierarchies, by distinguishing multiple occurrences of a term using the full path in the hierarchy, as well as the typical thesaurus

relationships of *synonymy* and *association*. Instead of designating a single preferred term for each concept, multiple terms with the same path (that are used interchangeably) can be listed together, separated by commas. All terms are path-based and provide the framework for the entire system – including browsing, indexing, interactive search expansion, and hierarchical search results.

We describe a study that evaluates the Metadata++ library system and assesses how easily indexers and searchers understand the path-based representation of terms. We describe multiple implementations and experiments that lead to a backend storage and retrieval mechanism optimized specifically for path-based metadata. The user interface consists of a smart client application, combined with web services, that satisfies specific architectural design objectives. We explain how Metadata++ integrates with a standard geographic information system to support both spatial and keyword-based information retrieval. We conclude with a comparison to other thesaurus-based systems and a description of future work.

1 Introduction

Suppose you are asked to decide whether or not to build a new campground next to a backcountry lake. Or perhaps you must evaluate how a proposed timber harvest would affect the wildlife within a particular watershed. These examples illustrate just a few of the many issues faced by natural resource managers as part of their daily responsibilities. Natural resource management is fundamentally interdisciplinary, with almost every project involving disciplines such as soil, forestry, vegetation, climatology, hydrology, wildlife and fish biology, recreation, and range land. In some areas, such as the Pacific Northwest of the United States, decisions often involve issues concerning cultural heritage, particularly for Native Americans.

1.1 Why we need a Digital Library for Natural Resource Management

Natural resource managers gather information necessary to make decisions about the environment from a wide spectrum of documents generated by various individuals for various purposes, including *Watershed Assessments* describing the health of a particular watershed, *Environmental Impact Statements* describing the short-term and long-term ramifications of a proposed action, and *Decision Notices,* including a complete account of the public appeal process. Many of these documents focus on numerous topics about a particular location; an environmental impact statement is such a document, often hundreds of pages in length. Other documents focus on a specific topic or issue – such as a wildlife survey for a particular location. Such specialist reports are often referenced by the larger, multi-disciplinary documents. The interdisciplinary collaboration typical in this domain is exemplified in a monitoring plan [91] with ten different authors: four fish biologists, a supervisory biological scientist, a research aquatic biologist, a research forester, a biologist, an ecologist, and a hydrologist who are from five different agencies: National Marine Fisheries Service (United States Department of Commerce), Forest

Service (United States Department of Agriculture), Bureau of Land Management (United States Department of the Interior), Fish and Wildlife Service (USDI), the United States Geological Survey (USDI), and the United States Environmental Protection Agency.

The conglomerate and heterogeneous nature of natural resource management produces a significant information management and retrieval challenge. Each project or decision generates a variety of documents as mandated by law (including the National Environmental Policy Act). Important information resides everywhere from bookshelves tucked away in an agency office, to a hard drive in someone's workstation, to a simple website managed by just one of many organizations. Such disparate information sources present a real problem for natural resource managers. The challenge of finding relevant information affects a variety of information needs including such things as making important decisions with significant environmental impact, responding to Freedom of Information Act (FOIA) requests, or conducting scientific studies.

One of the most popular "information retrieval" methods in natural resource management is to ask someone else who may have the information or know where to find it [8]. Although common in other domains as well [10], this word-of-mouth retrieval system does have drawbacks when it comes to scalability, predictability, scope, latency, and several other factors. Our research focuses on building a digital library system in order to provide easy access to interdisciplinary information [26] so users can more readily benefit from previous scientific findings and assessments. We collaborated extensively with Region 6 of the USDA Forest Service as part of a National Science Foundation Digital Government project[1].

1.2 Why we need human indexing

A long-standing debate in the field of digital libraries and information retrieval pits human indexing against machine indexing. A common belief states that human indexing is high quality and expensive, whereas machine indexing is low quality and inexpensive. Modern information retrieval techniques [7] significantly increase the quality of machine indexing – while retaining the relatively low cost. However, recent

[1] This work is supported in part by the National Science Foundation, grant number EIA 9983518. Any opinions, findings, conclusions, or recommendations expressed here are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

research suggests that both approaches are effective and have advantages [5]. Many of the advantages of human indexing are particularly applicable in natural resource management.

As is common with scientific data and reports [74], natural resource management documents are often large and heterogeneous in nature. For example, an Environmental Impact Statement (EIS) typically includes a variety of content such as text, maps, images, tables, and scientific data. Such diverse content reduces the effectiveness of machine indexing – which is primarily focused on text. While machine indexing algorithms may perform well on the textual parts of the document, the other parts would likely be skipped during indexing.

Most natural resource documents are targeted for printed medium, so few are published in hypertext. One of the most popular web search engines, Google, uses a page ranking algorithm based on hyperlinks [13]. During automatic indexing, the Google indexing engine searches for hyperlinks within the text. One of the contributing metrics when computing the page rank for a particular page is the relative importance of the page, which is measured by the number of other pages that link to that page and the relative importance of those referencing pages. So the importance of page A increases if page B contains a hyperlink that points to page A – and the amount of increase relates to the importance of page B. This hyperlink-based page ranking algorithm works very well for trillions of hyperlinked pages – but it would be less effective in a natural resource management library with only thousands of documents and few, if any, hyperlinks.

Many modern information retrieval algorithms utilize some form of statistical term occurrence to compute the rank of a document with respect to a particular query [7]. The statistical calculation is typically based on the number of times the search term appears in the document, the number of total words in the document, and the number of times the search term appears in the entire corpus of documents. For example, a term that is relatively infrequent in the corpus, but relatively frequent in the document will increase the rank of the document. A term that occurs frequently in the corpus or infrequently in the document will decrease rank of the document. These statistical metrics work well with a large, diverse corpus of documents – but they are not as suitable for a domain-specific digital library. For example, most natural resource documents relate to a specific

geographic place – whether it is a forest, a watershed, a campground, timber land, or some other type of place. The document usually contains the name of the location – typically on the title page or in the abstract – but the location name may appear relatively few times in the rest of the document (particularly for large documents). The relatively low term frequency within the document can reduce the ranking – even though the location is an important keyword for the document.

Statistical term occurrence is also used for document clustering [7], where documents are automatically categorized into clusters based on similarities between the documents. Many natural resource documents are mandated by law and have to follow a particular format. For example, a Record of Decision documents the entire process of making an environmental decision. Every Record of Decision must follow a specified format – including sections that describe the decision, alternatives (including action alternatives and no-action alternatives), compliance with regulation, and public involvement. The "boiler plate" nature of these documents results in the same terminology occurring in every document. A typical document clustering algorithm may cluster all of these documents together – even when the actual meaning (the useful information contained within the document) of the documents is diverse.

A large number of existing natural resource management documents are not available in electronic form. The time and resources required to digitize several hundred pages (including scanning, optical character recognition, and so forth) for a single document significantly exceed the time and resources required to manually index that document. In this case, the indexer can create a short electronic abstract of the document (including title, author(s), physical location of the document, etc.) and then manually index the abstract as if it were the entire document.

1.3 Why we need a thesaurus-based system

As explained by Svenonius [105], discussions regarding the necessity for controlled vocabularies in effective information retrieval began many years ago – long before the Internet or even the microchip. Even though controlled vocabularies can be costly to build, they are built and used because they improve retrieval [38]. Research

indicates that controlled vocabularies improve consistency in indexing and make indexing more predictable [5].

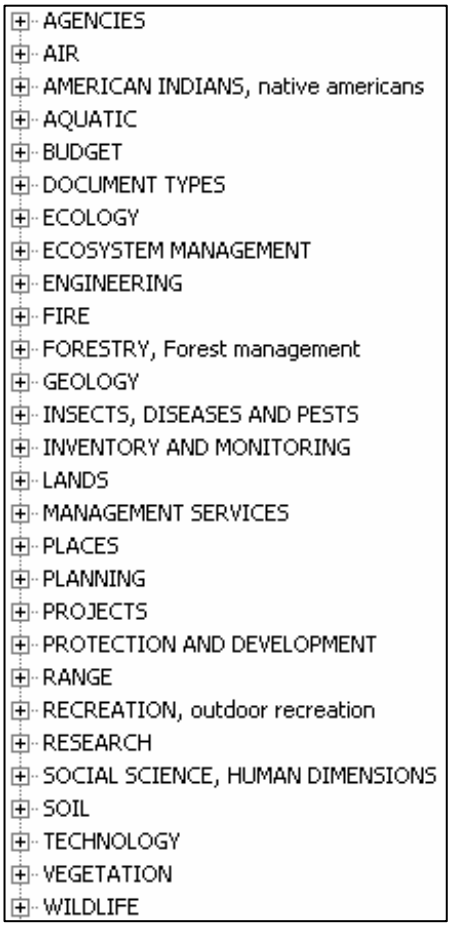


Figure 1-1: Top-level discourses

The need for a controlled vocabulary, or thesaurus, is particularly evident in the interdisciplinary domain of natural resource management. The terminology of interest in this application domain spans a number of subject areas, as shown in Figure 1-1. Each subject area, or discourse[2], includes one or more controlled vocabularies containing terms that describe the important concepts and ideas. These terms are words, or more frequently phrases, routinely used by scientists, managers, and other experts during their work. Most of the terms come from existing sources – published glossaries, terminologies, and taxonomies commonly used in natural resource management. As part of our work, our team researched and evaluated existing sources to determine their

[2] A *discourse* is defined as a way to talk about and understand experiences and concepts within the social world. A discourse seeks to control and freeze one particular meaning and understanding of the world [16].

suitability for use in our natural resource thesaurus [107]. In other cases, a committee of experts organized or defined terms of interest, when published vocabularies were not available. As an example, the *Wetlands* vocabulary (within the *AQUATIC* discourse) was compiled from multiple sources: United States Army Corps of Engineers Wetlands Classification System, the United States Fish and Wildlife Service (USDI) Wetlands Glossary, the National Wildlife Federation Wetlands Glossary, and the United States Forest Service (USDA) Wetlands Classification System. Similarly, the *Watershed Management* vocabulary (also within the *AQUATIC* discourse) includes terms from: the United States Forest Service (USDA) manual for organization, StreamNet [104], the United States Geological Survey water glossary, and the Natural Resources Conservation Service (USDA) water glossary. Many of the discourses shown in Figure 1-1 contain several different controlled vocabularies. For example, there are approximately a dozen vocabularies just for describing places. These vocabularies include administrative places (USDA Forest Service, USDI Bureau of Land Management, etc.), political places (states, counties, etc.), and watersheds.

A thesaurus helps to translate between keywords chosen by indexers and search terms used by searchers [42]. For example a scientist may publish a report about typical habitat of *Chamaecyparis lawsonian*; and a manager may be looking for information about growing conditions for *White Cedar*. Since these terms are defined as synonyms within the thesaurus, the manager can find the relevant document. Relationships among terms can also lead searchers from one discourse to another [21]. The manager may see that *White Cedar* is related to a particular soil type and follow that relationship to other relevant terms in the *SOIL* discourse, thereby finding more potentially relevant information. Many studies [7] show the benefits of thesaurus-based query expansion – and research [38] also indicates that users prefer using a thesaurus during searching a large percentage of the time.

1.4 Why we need to enhance the traditional thesaurus model

Various thesaurus models [1] support a variety of relationships that provide a rich framework for information retrieval. A fundamental step in the NISO standard for thesaurus construction is defining inter-related concepts [3]. Each concept represents a well-defined thing or idea – but can be identified by a number of labels or descriptors.

For each concept, one descriptor is identified as preferred while all others are designated as non-preferred descriptors. For example, the concept might be a particular species of tree and the preferred descriptor, or term, to identify that concept might be the scientific name for that species. Other common names for that species would be related to the scientific name and designated as non-preferred terms. For example, *Chamaecyparis lawsoniana* might be a preferred term that is related to "White Cedar" (a non-preferred term).

A fundamental goal of our natural resource digital library is to use existing terminology "as-is": to take vocabularies as they currently exist and use them in a thesaurus-based retrieval system. By preserving the terminology and structure of existing vocabularies, we maintain user familiarity and avoid the overhead of defining and organizing concepts. By allowing terms to exist in multiple vocabularies, we enable differences in connotation distinguished by the hierarchical path to the term. Building a digital library system based on this hierarchy of terminology from numerous controlled vocabularies provides a framework for efficient and intuitive interdisciplinary retrieval.

1.5 Statement of Research

We define a thesaurus-based conceptual model that uses path-based terms to retain existing terminology and accommodate the same term in different contexts. Based on this model, we implemented a storage mechanism that provides efficient storage and retrieval of path-based metadata, combines both machine and human indexing, and supports concurrent maintenance of terminology. Additionally, we built a digital library application that allows natural resource managers to effectively:

- Explore the hierarchy of terminology
- Find terms of interest and understand the context of those terms
- Index documents with path-based keywords
- Interactively search for documents with path-based search terms
- Comprehend search results within context of the hierarchy

Chapter 2 explains how we enhanced the thesaurus model and built a natural resource management digital library called Metadata++. We conducted a user study of the application, which is described in Chapter 3. Chapter 4 formally defines the

conceptual model using set theory and first-order logic. In Chapter 5 we explain our implementation of a path-based storage and retrieval mechanism and Chapter 6 discusses how we built the software application. Chapter 7 describes how we integrated the path-based hierarchy of terms with a geographic information system. Chapter 8 summarizes related work, conclusions, and future work.

2 The Metadata++ Digital Library

The Metadata++ Digital Library, designed specifically for natural resource management, is the result of years of research by many individuals from a variety of backgrounds. This chapter describes how the system assists users in accomplishing various tasks – such as browsing the hierarchy of controlled vocabularies, finding terms of interest, indexing documents with path-based keywords, and searching for documents using path-based search terms.

2.1 Hierarchy of Controlled Vocabularies

The framework of the Metadata++ digital library is a hierarchy containing multiple controlled vocabularies from a variety of discourses within the natural resource management domain. In order to support different users from different disciplines using different discourses, Metadata++ stores and presents each discourse (and vocabulary) "as-is" (exactly as defined by the domain experts). Thus each user may use the vocabularies most familiar to him or her and see the terms in the organization that he or she is familiar with. This hierarchy of terms is a core functional component of the system, as well as the basis for the Metadata++ user interface.

2.1.1 Path-based Terms

In Metadata++, a term is always presented with its corresponding path in the hierarchy. The user never encounters an individual term by itself – only as it occurs within the context of the hierarchy. Path-based terms give more context than a simple list of keywords. The user can infer, based on the term along with the path – the meaning or connotation intended for that particular term (without requiring a dictionary or glossary definition). Each path contains a sequence of terms – beginning with the top-level discourse and ending with the term of interest. Terms deeper in the hierarchy are typically more specific, while higher level terms are typically more generic.

The relationship between each successive term in a path is analogous to the Narrower Term (NT) relationship as defined in a traditional thesaurus [3]. The NISO standard for thesaurus construction [3] also suggests the use of more specific hierarchical relationships – including the whole-part relationship ('central nervous system' is a part of the 'nervous system') and the instance relationship ('Cinderella' is an instance of 'fairy tale'). Controlled vocabularies in the natural resource domain use a wide variety of semantic meanings in the hierarchical relationships among terms. In some cases, the narrower term relationship represents taxonomic classification (as in the taxonomy of species). In other cases, the narrower term relationship represents spatial containment (as in USGS hydrologic unit codes [110]). In fact, often a single path uses different types of relationships at different points in the path. For example, consider the path *PLACES\Administrative\National Forest System\National Forests\Pacific Northwest Region, Region 6\Wenatchee NF\Chelan RD*. The first relationship represents a specific type – *Administrative* as a type of *PLACES*. The next term implies an instance relationship – the *National Forest System* is one instance of administrative places. Towards the end of the path (...*Wenatchee NF\Chelan RD*), we see two different semantic meanings within the same relationship – *Chelan RD* (ranger district) is geographically contained within *Wenatchee NF* (national forest), and *Chelan RD* is also an organizational unit within *Wenatchee NF*. As illustrated by the previous example, controlled vocabularies may contain a variety of semantic meanings for the hierarchical relationship. Metadata++ does not enforce specific types of semantics on the hierarchical relationship, nor limit the types of semantics implied by the hierarchy. Instead, terms (and paths) are arranged as appropriate based on the controlled vocabulary.

2.1.1.1 Browse

Bates [9] describes the "docking" process that occurs when any user first approaches an information system. She relates this process to people approaching a librarian and first asking a general question (as most people do), such as "Where are the books about math?", when the real need is to find something more specific, such as books about linear equations. This docking process allows the person to become familiar with the system (or the librarian) before proceeding with the actual search process.

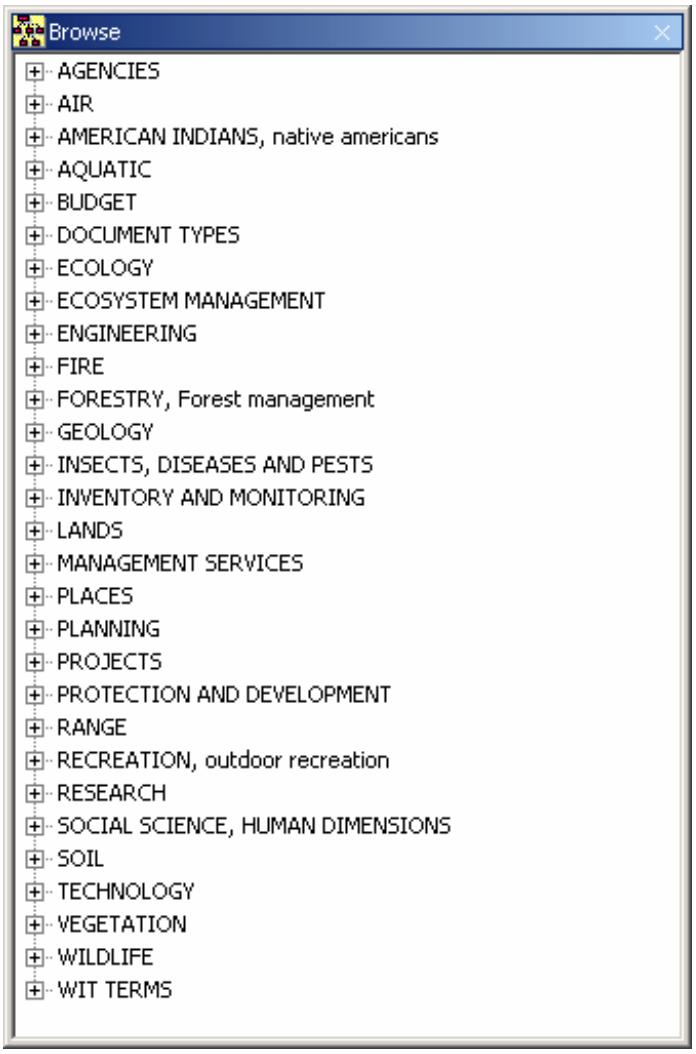


Figure 2-1: Screenshot of Browse window

The Metadata++ application starts by presenting the user with the Browse window, as illustrated in Figure 2-1. The Browse window allows the user to become familiar with the system – similar to starting a conversation with a librarian or browsing a card catalog. Browsing the hierarchy is particularly useful for first-time users and casual users [71] who may not be familiar with the content of the hierarchy. Even experienced users receive benefit from browsing – especially when exploring vocabularies from unfamiliar subjects. The user may expand or collapse a term using typical mouse functions (either clicking on the plus/minus icon or double-clicking on the term itself).

2.1.1.2 Find

One advantage to a digital library (as opposed to a traditional card catalog) is the ability to search for individual words within subject headings [9]. For example, suppose a user approaches the system interested in finding out about old growth forests. The user may not have the time (or patience) to browse through the various controlled vocabularies in the hierarchy looking for terms referring to old growth. Instead the user may use the Find window as illustrated in Figure 2-2.

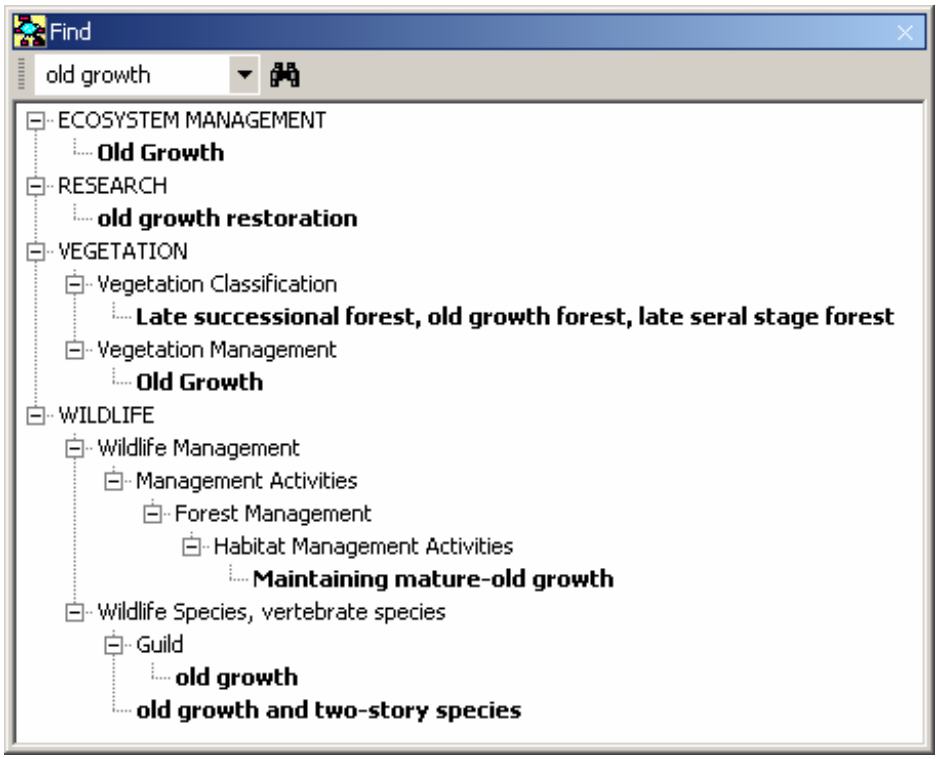


Figure 2-2: Screenshot of Find window

The Find window provides a quick and easy way to navigate the hierarchy by typing in words of interest. The user simply clicks in the text box at the top of the window and types in what he or she is looking for, then either clicks the binoculars icon or presses the <ENTER> key on the keyboard. The application will find all terms in the system matching the search string based on a wildcard match (e.g. when the term contains the search string). As shown in Figure 2-2, the Find window displays matching terms with their corresponding paths in context of the hierarchy. This hierarchical display of terms helps the user understand where the terms fit within the various controlled vocabularies. If no matching terms exist, the Find window will simply inform

the user that no matches were found. The Find window will remember all of the term searches during the session, so the user can select from previous term searches by clicking the down arrow and selecting from the list. It is important to note that the Find window only searches the hierarchy of terms – it does not search for documents.

The Find window displays the entire path of each matching term, but it does not display all of the siblings of each term. For example, the term *WILDLIFE\Wildlife Management\Management Activities\Forest Management\Habitat Management Activities* has multiple child terms – but only ...*Maintaining mature-old growth* is displayed in the Find window (in Figure 2-2) because it is the only child term that matches 'old growth'. However, suppose the user wants to view other habitat management activities. He or she could click on the Browse window and manually browse to the same location in the hierarchy; but the user may also double-click on the ...*Maintaining mature-old growth* term in the Find window. Double-clicking a term in the Find window will automatically browse to that same term in the Browse window – so the user can easily see all of the other habitat management activities.

In addition to first-time or casual users, the Find window is convenient for experienced users who know the vocabularies very well. For example, suppose a botanist is using the system to find information about a particular species of violet. He or she may know exactly where the species resides in the vocabulary of taxonomic species – but browsing to that term may require several mouse clicks while expanding the various levels of the hierarchy. Instead, the user can type the species name into the Find window, double-click on the term when it is found, and the system will automatically highlight that term in the Browse window – all with just a few key strokes and mouse clicks.

2.1.2 Exploring terms

Bates' design model [9] suggests a series of possible actions that a user might initiate to find out more information about any specific term. These actions include:

- Show me other words for the same subject (i.e., synonyms)
- Show me other related topics (i.e., related terms)
- Show me some book titles on this subject

Metadata++ supports these and similar actions while browsing the hierarchy. Clicking on any term in the hierarchy (in either the Browse window or the Find window) brings focus to that term. The user can find out additional information about that term by right-clicking on the term and reviewing the popup context menu (as shown in Figure 2-3). This section explains the information displayed in the context menu.

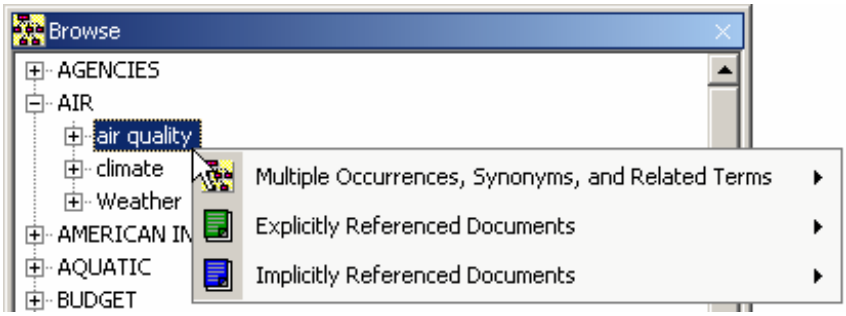


Figure 2-3: Screenshot of right-click context menu

2.1.2.1 Synonyms

Most thesaurus-based systems support some form of synonymy [35,101] between concepts. The NISO thesaurus standard [3] uses the Use (U) and Used For (UF) relationships to relate the preferred term (which may be used during indexing) with non-preferred terms (which may not be used during indexing). WordNet [37] uses *synsets* (a group of terms that describe the same concept). Metadata++ also supports synonyms – a relationship between terms (including the entire path) whose meanings are regarded as the same, or nearly the same, in a wide range of contexts [3]. For example a common name for a particular plant and the scientific name of the same species could be designated as synonyms. The user can browse synonyms within the hierarchy – and can also automatically add synonyms to a search (as explained later in this chapter). The natural resource domain presents several unique situations that merit something more than the traditional synonymy relationship. This section explains these situations and describes how Metadata++ accommodates each using specialized relationships.

2.1.2.1.1 Multiple Occurrences

Natural languages, particularly the English language, are filled with ambiguous words. These ambiguities include words that sound the same (homonyms) and words that look the same (homographs) but have entirely different meanings. For example, the word ‘crane’ can mean a bird or a machine designed for lifting large objects. In this case

'crane' is both a homonym and a homograph. The NISO thesaurus standard recommends distinguishing such ambiguities using parenthetical qualifiers [3]: 'crane (bird)' and 'crane (lifting equipment)'. A similar situation occurs when the same term (or concept) logically belongs under two different broader concepts. For example, a piano is both a string instrument and a percussion instrument – but it is still a piano. The thesaurus standard accommodates this situation using polyhierarchies – where a single concept has more than one broader concept.

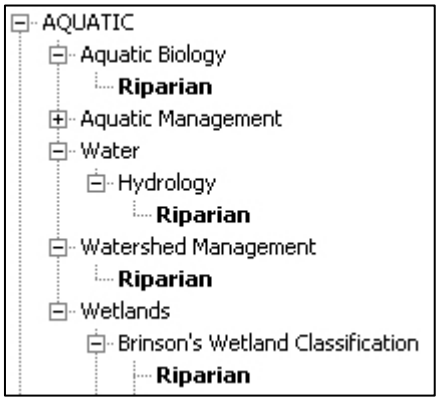


Figure 2-4: Multiple Occurrences

In natural resource management terms frequently exist in multiple controlled vocabularies. For example, aquatic biologists and fish biologists work with similar ideas and concepts – but these ideas and concepts are organized and related differently. Because vocabularies are not combined into a single ontology, as in some ontology-based systems [23,103], we allow the same term to appear in various places in the hierarchy. Figure 2-4 illustrates a few of the many multiple occurrences of the term *Riparian* (which means on or near the banks of a body of water). Multiple occurrences of a term may appear anywhere in the hierarchy. For example, one occurrence of a particular term may be several levels deep in the hierarchy and have no narrower terms, while another occurrence of the same term may be near the top of the hierarchy and have many narrower terms.

When the same term exists multiple times within the same (or similar) discourse(s), it may be describing the same general concept with slightly different connotations. For example, Figure 2-4 illustrates the term *Riparian* as a child term for both *Watershed Management* (referring to how the riparian area affects the encompassing

watershed) and *Wetlands* (referring to a particular type of wetland that is not permanently inundated but is close to surface water). The following two excerpts from actual documents illustrate the differences between these subtly different connotations. In the first document [91], we read (pages 46-47, emphasis added):

> Maintain and restore water quality necessary to support healthy *riparian*, aquatic, and *wetland* ecosystems. Water quality must remain within the range that maintains the biological, physical, and chemical integrity of the system and benefits survival, growth, reproduction, and migration of individuals composing aquatic and *riparian* communities. …
>
> Maintain and restore the species composition and structural diversity of plant communities in *riparian* areas and *wetlands* to provide adequate summer and winter thermal regulation, nutrient filtering, appropriate rates of surface erosion, bank erosion, and channel migration and to supply amounts and distributions of coarse woody debris sufficient to sustain physical complexity and stability.

This document refers to riparian wetlands – and what issues need to be monitored in order to maintain the health of these areas – so the appropriate keyword for this document is *AQUATIC\Wetlands\Riparian*. In the second document [81], we read (page 69, emphasis added):

> Design and implement *watershed* analysis: to determine *watershed* status, resilience and capabilities; examine fish ecological relationships; establish watershed-specific boundaries for *Riparian* Habitat Conservation Areas and *Riparian* Management Objectives; and identify *watershed* restoration and monitoring objectives, strategies, and priorities.

This document describes a management plan for particular watersheds – including riparian management objectives – so the appropriate keyword for this document is *AQUATIC\Watershed Management\Riparian*. As illustrated by these documents, these two different occurrences of *Riparian* are similar, but distinct. Each connotation is necessary and important – as specific keywords for these (and other) documents and to support the information needs of natural resource managers.

Multiple occurrences of a term are also used to represent homographs. For example, the term *dolphin* is a marine mammal found within the *WILDLIFE* discourse and is also a term used to describe a submerged piling within *Aquatic Habitat Elements*. These terms are not subtly different connotations of the same concept; they are entirely different meanings for the same word. In this case, the path associated with these two occurrences of the term easily distinguishes the intended meaning (the second path is shown in Figure 2-5).

Multiple occurrences in Metadata++ unify the issues addressed by parenthetical qualifiers and polyhierarchies – by letting the path distinguish the various connotations (or distinct meanings) of the term. Regardless of the number of times a term appears within the hierarchy – and regardless of whether the various occurrences represent slightly different connotations or entirely different meanings – the user can easily select one (or more) occurrences. For example, a user interested in riparian areas relating only to watershed management can select *AQUATIC\Watershed Management\Riparian* (and not any of the other occurrences of *Riparian* shown in Figure 2-4). A different user, beginning a watershed analysis, might use information about all aspects of *Riparian*, for a particular location. The user can select any one of the occurrences, then simply right-click (as described later in this chapter) and automatically add all of the other multiple occurrences – finding all documents related to *Riparian*.

2.1.2.1.2 Polyterms

Unlike a traditional thesaurus, Metadata++ also supports *polyterms*. A polyterm exists when two or more terms are interchangeable and have the same parent (i.e. broader term). Even a specialist within that discourse would not need to distinguish between the individual terms within a polyterm – because they all have similar meanings and connotations within that vocabulary. A polyterm may also describe a class or collection of items that is named by a list of its elements. Figure 2-5 illustrates a polyterm within the *Wildlife Habitat* vocabulary. As child terms of ...*\Anthropogenic – Related Habitat Elements*, the terms *mooring pile*, *dolphin*, and *buoy* all describe inanimate, partially (or completely) submerged objects. This collection of similar objects does not have a name so it is represented by a list of its elements (a polyterm).

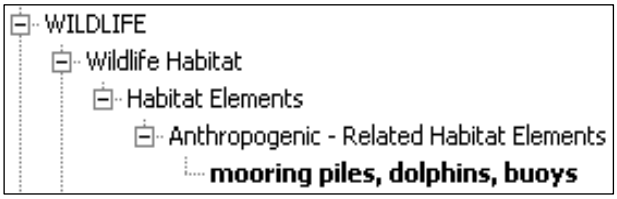


Figure 2-5: Example of Polyterm

Another example of polyterms is a term combined with its own abbreviation or acronym (i.e. lexical variants [3]). For example, the terms *United States Department of Agriculture* and *USDA* are used interchangeably – so these terms are combined into a

single polyterm (*United States Department of Agriculture, USDA*). A typical thesaurus would require that one term be designated as the preferred term, and the UF (Use For) relationship would relate the preferred term to all of the non-preferred terms. In this case, only the preferred term could be used anywhere else in the system, such as during document indexing or searching.

Different users may prefer different terms; some users may even be unfamiliar with various terms (or even entire controlled vocabularies). So instead of forcing the users to agree on a single preferred term, Metadata++ combines all of the interchangeable terms into a polyterm. In addition, terms that constitute a polyterm still function as individual terms elsewhere in the system – such as a multiple occurrence. For example, the term *dolphin* is part of the polyterm described above – but it also occurs within the vocabulary describing marine mammals.

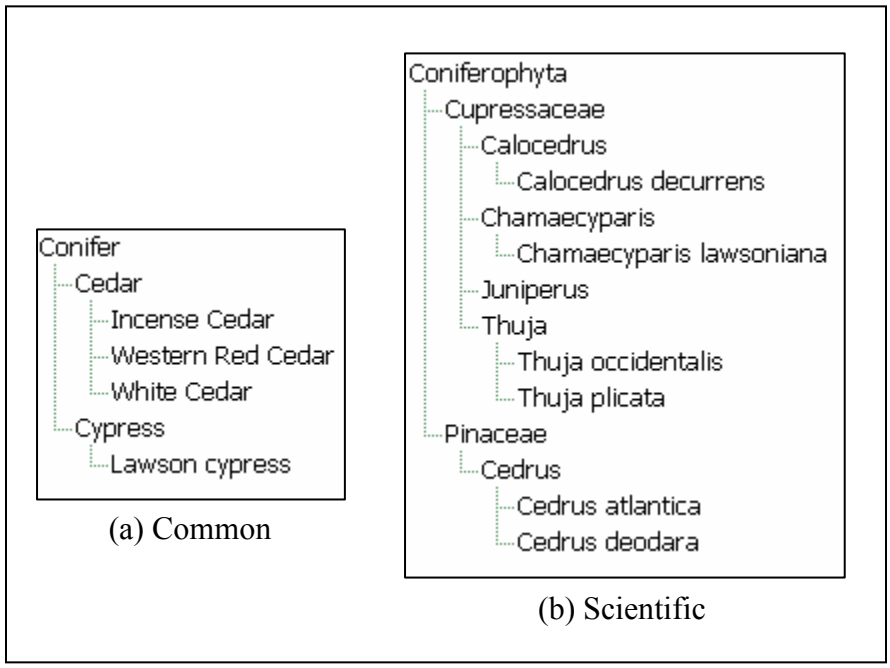


Figure 2-6: Excerpts from two vocabularies about conifers

2.1.2.1.3 Non-transitive synonyms

Unlike traditional notions of synonymy, synonymy within Metadata++ is not transitive. Situations occur in natural resource management where transitive synonymy is not valid. For example, consider the vocabularies illustrated in Figure 2-6. Figure 2-6.a is an excerpt of a vocabulary of common names for conifers. Figure 2-6.b is an excerpt of a vocabulary of scientific names of conifers. In practice, there is not an exact one-to-

one mapping between common names and scientific names. The term *White Cedar* refers to the species *Chamaecyparis lawsoniana* – so these terms would be designated as synonyms in Metadata++. However, *White Cedar* also refers to the species *Thuja occidentalis* – so those two terms would also be designated as synonyms. In this case *Chamaecyparis lawsoniana* is synonymous with *White Cedar* and *White Cedar* is synonymous with *Thuja occidentalis* – but *Chamaecyparis lawsoniana* is not synonymous with *Thuja occidentalis*. (They are completely different species belonging to different genera.)

A lack of transitivity among synonyms is a unique and powerful feature of Metadata++. Consider a natural resource manager searching for documents about *White Cedar*. Since *White Cedar* is synonymous with both *Chamaecyparis lawsoniana* and *Thuja occidentalis*, documents may be found that are about either (or both) species. However, consider a botanist searching for documents about *Thuja occidentalis*. Metadata++ would find documents specifically related to that species and documents related to the more general term *White Cedar*. The search would not, however, automatically find documents related specifically to *Chamaecyparis lawsoniana* – because the synonym relationship is not transitive in Metadata++.

2.1.2.2 Related Terms

Metadata++ also supports general relationships between terms that are not synonymous, but that have some association or correlation within the application domain. The NISO thesaurus standard [3] describes an associative relationship as a "relationship [that] covers associations between descriptors that are neither equivalent nor hierarchical, yet the terms are semantically or conceptually associated to such an extent that the link between them should be made explicit in the thesaurus" and goes on to say that "whenever one term is used, the other should always be implied. … Moreover, one of the terms is often a necessary component in any explanation or definition of the other." Metadata++ more broadly defines associative relationships to capture any arbitrary correlation between terms specified by the user. For example, a relationship may be used to relate *Wenatchee National Forest* and *Western Hemlock* because Western Hemlock trees are found in the Wenatchee National Forest.

Relating terms is a powerful tool for capturing domain knowledge within the hierarchy. Each term can participate in multiple relationships; so *Western Hemlock* may also be associated with *AIR\climate\classification\Koeppen\Koppen Climate Classes\Dry climates\arid* and *SOIL\soil classification\Orders, Sub-Orders and Great Groups\Alfisols* – to indicate that these trees typically grow in the specified climate and soil type. Associative relationships are used to represent any functional or observational correlation between terms, and, like synonymy, association is not transitive. Because of the generic nature of associative relationships, Metadata++ does not implicitly traverse these relationships when executing a search. Instead, related terms are displayed so that the user may easily extend or refine a particular search.

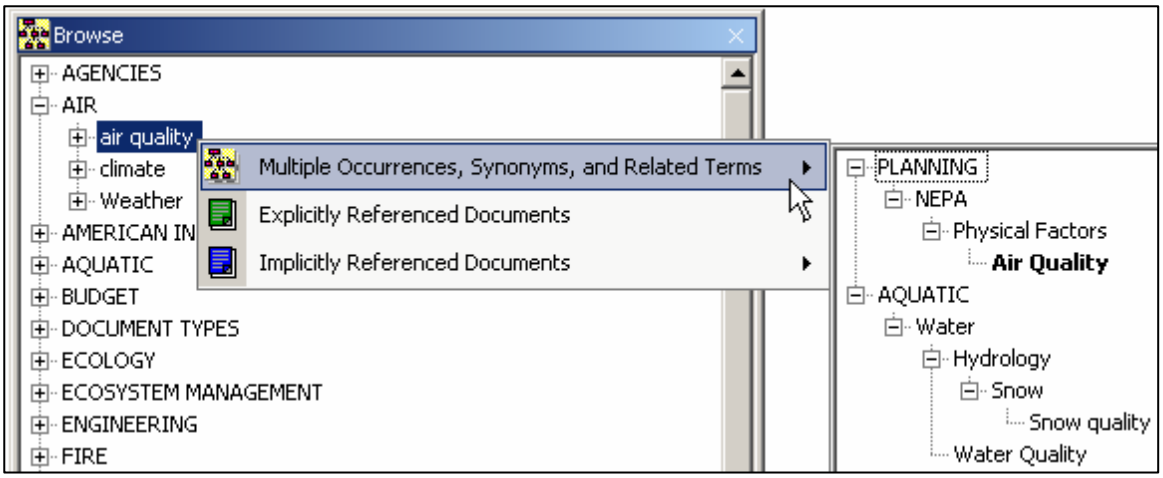


Figure 2-7: Cropped screenshot of Related Terms

When right-clicking on a term to show additional context information, the first menu item on the popup menu is called "Multiple Occurrences, Synonyms, and Related Terms" (as show in Figure 2-7). Moving the mouse over this menu item will display a popup tree containing all of the multiple occurrences, synonyms, and related terms (distinguished by color) for the current term. The popup tree displays all of these terms using their full paths – in context of the hierarchy. Clicking on any of these terms will automatically locate that term in the Browse window – providing a quick and easy way to navigate the knowledge contained within the various relationships among terms.

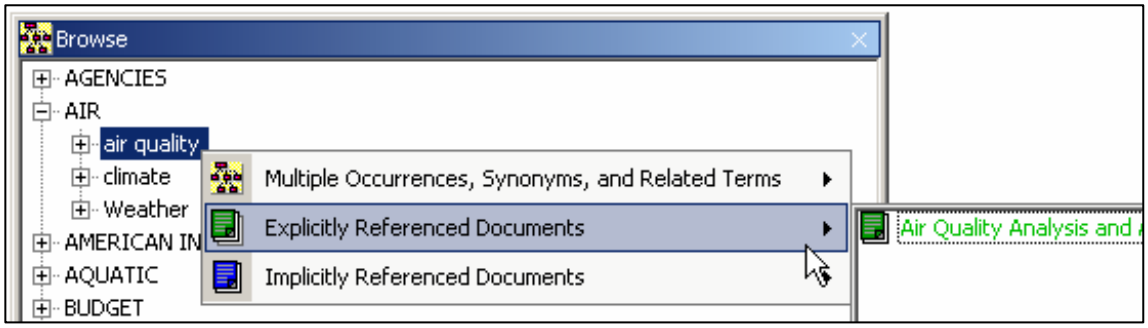


Figure 2-8: Cropped screenshot of Explicitly Referenced Documents

2.1.2.3 Documents

Research indicates the human indexing and machine indexing are each effective in certain situations and complement each other [6,38]. Metadata++ combines the advantages of both types of indexing. For any term in the hierarchy, the user may easily view explicitly referenced documents (as shown in Figure 2-8) and implicitly referenced documents (as shown in Figure 2-9). Clicking on a document in either list will open a separate window containing the metadata for that document – including path-based keywords and summary information (as described in Section 2.2.2). "Explicitly Referenced Documents" are those documents for which a human indexer explicitly selected that term as a keyword (regardless of whether or not the document actually contains the term).

"Implicitly Referenced Documents" are those documents that actually contain the term based on a full-text index within Microsoft Index Server (as described in Section 5.2.4). When finding implicitly referenced documents for a particular term, Metadata++ queries Index Server for the exact text of the term regardless of the path. Since the path is ignored, the list of implicitly referenced documents would be the same for multiple occurrences of the same term. Terms consisting of multiple words are treated as a phrase, so only documents containing the exact phrase are returned. For example, the implicitly referenced documents for the term *AQUATIC\Watershed Management* are all those documents that contain the phrase 'watershed management' (uppercase versus lowercase is ignored). In this example, a document containing the individual words 'watershed' and 'management', but not the exact phrase 'watershed management', would not be included in the list of implicitly referenced documents. The order of the implicitly referenced documents is determined by an unpublished ranking algorithm within Index

Server that uses word density and inflection. Metadata++ uses each term within a polyterm separately when finding implicitly referenced documents (e.g. searching for the term within the content of the document). So a document containing the term *mooring pile* would be implicitly referenced by the polyterm above – even if it did not contain *dolphin* or *buoy* (see Figure 2-5).

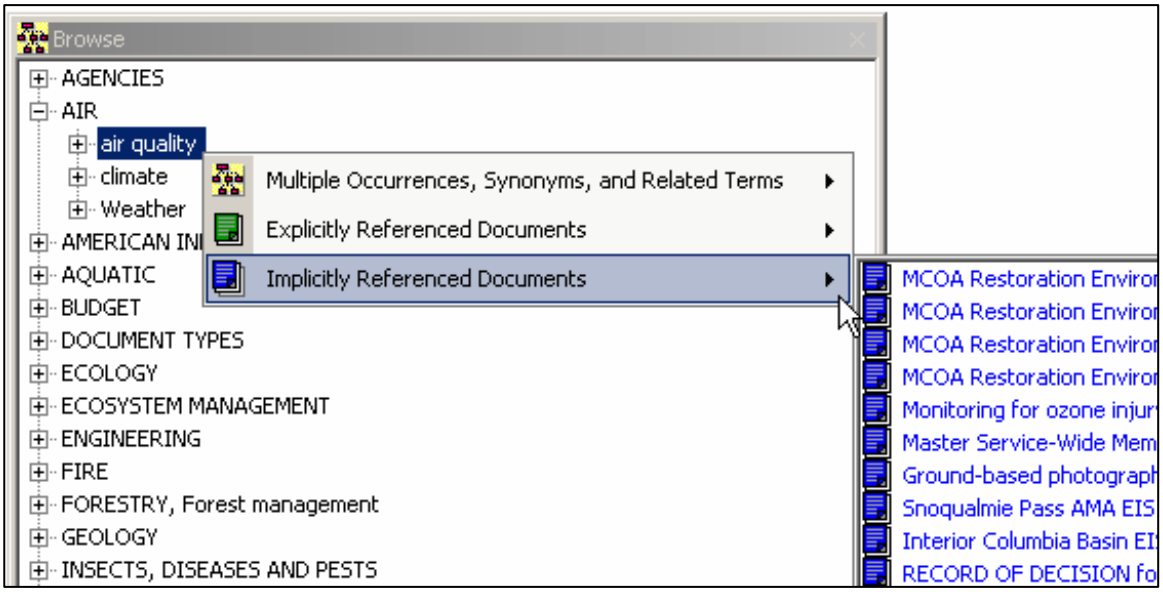


Figure 2-9: Cropped screenshot of Implicitly Referenced Documents

2.2 Indexing

A long-standing issue in information retrieval is the comparison of human indexing with machine indexing. Human indexing is expensive and time consuming. Some individuals within natural resource management abide an unwritten rule: if it takes longer than 15 minutes to provide metadata for a document, it will not be done. Despite the lower costs of machine indexing, the higher quality of human indexing still has advantages [5,6]. Mai [69] suggests a domain-centered approach to indexing instead of the common document-centered approach to indexing. A domain-centered indexer considers how the document relates to the collective domain knowledge – instead of just the content of the document itself. Nielsen [86] shows the benefits of using a thesaurus to support human indexing.

In Metadata++, document metadata consists of two types: summary information and keywords. Users view document metadata as shown in Figure 2-10. Clicking on the

URL will open the actual document in a separate window – so the user may review the document content. This section explains how Metadata++ simplifies the process of providing high-quality human indexing.

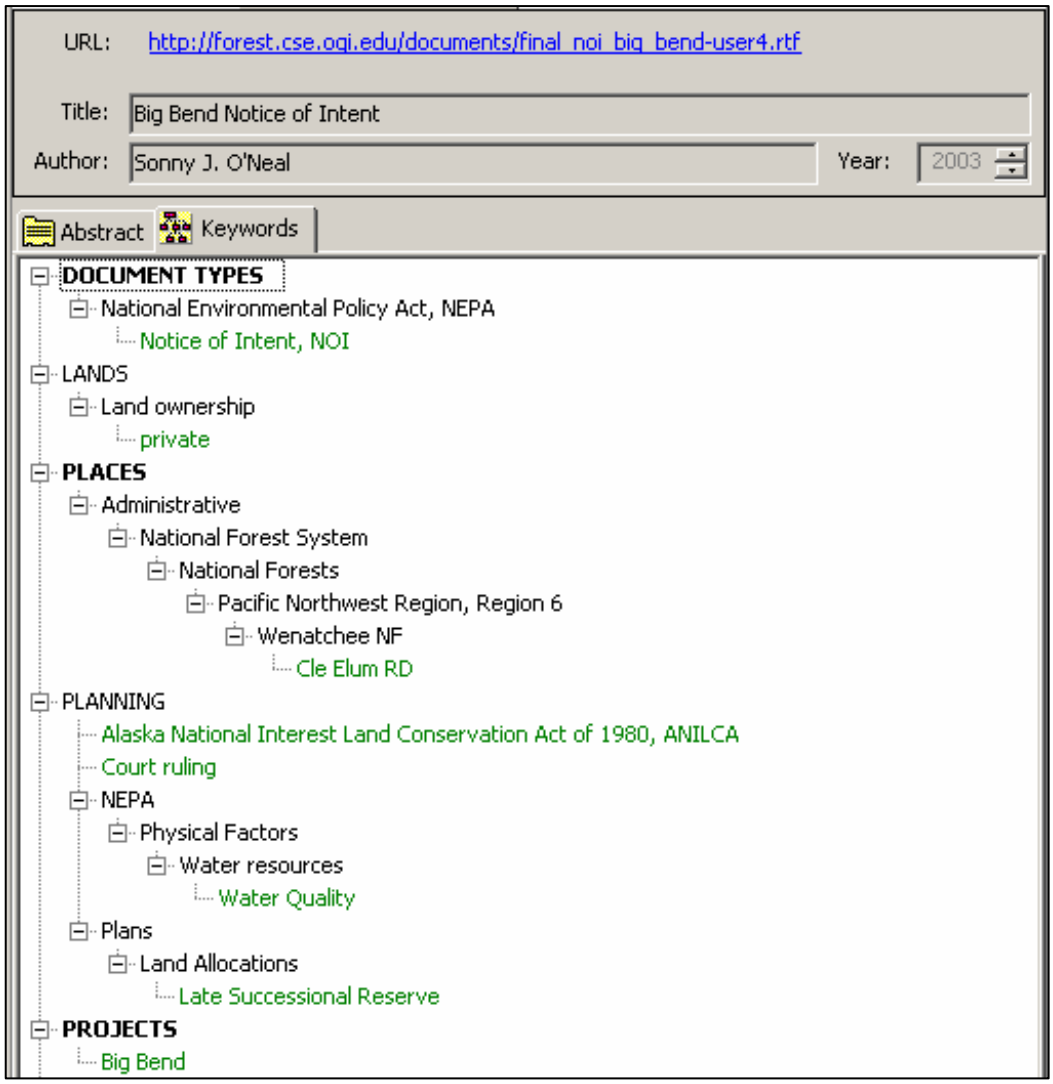


Figure 2-10: Cropped screenshot of document metadata

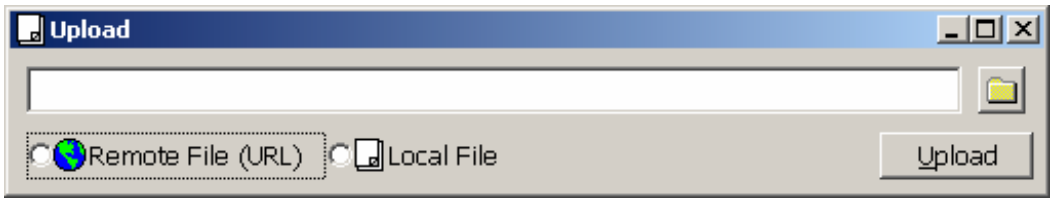


Figure 2-11: Screenshot of document upload

2.2.1 Submitting Documents

Authors (or indexers) must login to the system before submitting new documents to the system. The Upload window (as shown in Figure 2-11) allows two types of document submission. If the document already exists at a publicly available URL, the author may type (or paste) in that URL. Upon submission, Metadata++ will make an HTTP request for the document and store a local copy of the document (for full-text indexing) on the Metadata++ server. The external URL is retained – so whenever a user views that metadata for that document (path-based keywords and summary information), he or she will see the original URL (as submitted). Metadata++ assumes that the external document will not change and does not automatically synchronize the local copy of the document with the external document after initial submittal. If the document does not already exist at a public URL, the author may browse to the file on his or her computer and upload it. Metadata++ stores the file on the server and gives it a URL. If a particular document is not available in electronic format, the indexer may create an electronic abstract for that document and submit the abstract as any other document.

2.2.2 Summary Information

Metadata++ stores each document as a separate file on the server. The summary information for each document includes the following: title, author(s), date of publication, and abstract (not visible in Figure 2-10). The summary information resides in the document properties [75] of each file as defined by the Object Linking and Embedding (OLE) standard produced by Microsoft Corporation. OLE does support custom document properties, so the summary information could be extended to include attributes from other metadata standards such as Dublin Core [30].

The preferred method for providing summary information is for the author(s) to add the information to the file before publishing the document. Both Microsoft Word® and Adobe Acrobat® provide a mechanism to set the summary properties of a document. In Microsoft Word®, document properties may be set by choosing ‘Properties’ from the ‘File’ menu. In Adobe Acrobat®, document properties may be set by clicking the ‘File’ menu, then choosing ‘General’ from the ‘Document Info’ submenu. Summary properties can also be provided in HTML using the appropriate <META> tags. If the summary information already exists when the author or indexer submits the document to

Metadata++, the system will automatically recognize and display this information. If the author(s) did not include the summary information when the document was published, this information may be added or modified after the document has been submitted to Metadata++ – either the document metadata window (as shown in Figure 2-10) or by updating the file directly on the server.

2.2.3 Keywords

Consistent with the rest of the Metadata++ application, document keywords are hierarchical (as opposed to a flat list of terms). A document keyword is a term (including path) that the indexer selects from the hierarchy during indexing and attaches to the document. The indexer attaches keywords by selecting any term in the hierarchy and dragging the term onto the document metadata window. Any term in the hierarchy may be used as a keyword, regardless of where it appears and whether or not it has narrower terms. The indexer may drag terms from either the Browse window or the Find window. Using the hierarchy of vocabularies as the framework for indexing supports domain-centered indexing [69]. The indexer can review the document and the domain – and select keywords that relate the document to all relevant aspects of the domain. Three of the top-level discourses, *DOCUMENT TYPES*, *PLACES*, and *PROJECTS* are very important to nearly all documents in natural resource management. Because of their importance, these three terms are shown in bold in the keyword hierarchy – as a reminder to indexers that they need to select at least one term from each of these discourses. This feature is a convenience built into the software application but is not formally part of the Metadata++ model (see Section 4.1.3). After adding the desired keywords, the indexer must click the Save button to save the changes.

One of the drawbacks of human indexing is the time (and corresponding cost) required to provide a complete and thorough set of keywords for a document. Metadata++ simplifies this process by enabling the indexer to more easily find relevant, related terms during indexing. The indexer may right-click on any existing keyword to see a popup context menu that displays all of the multiple occurrences, synonyms, and related terms (as illustrated in Figure 2-7). If any of those terms are relevant to the document, the indexer may click on the relevant term within the popup tree. That term (and corresponding path) will automatically be added as a keyword. For example,

suppose you are indexing a document that discusses contamination caused by a chemical spill. You select the *AIR\air quality* term as a keyword because it relates to the document. You then right-click on that keyword and see the popup tree of related terms – including *AQUATIC\Water\Water Quality*. The document also discusses the contamination of the water supply – so you click on *AQUATIC\Water\Water Quality* and that term is automatically added as keyword for the document. This mechanism exploits the relationships defined in the hierarchy to simplify the process of providing thorough metadata.

2.3 Searching

The main goal of any library, including digital libraries, is to warehouse and organize information and make it more accessible to users. Accessibility of information is based on how easily users can search for, and find, relevant information. Nielsen [86] reports that users wanted and needed a thesaurus – primarily for query formulation. Bates [9] suggests two different thesauri – one thesaurus for indexers and a different thesaurus for searchers. Metadata++ uses the same hierarchy for both tasks, providing consistency between tasks. This section describes how Metadata++ supports searchers in exploratory searches to find documents of interest.

2.3.1 Interactive query expression/expansion

Bates [9] makes some interesting observations in regards to searching and the nature of information systems. She states that it is impossible to accurately predict what specific aspects of a topic a search will pursue and which specific terms the searcher will use. She continues with this suggestion:

> "[We should] stop trying to design systems that will *target* the desired information through perfect pinpoint match on the one best term; rather, design systems to *encompass* the answer by displaying and making it easy to explore a variety of descriptive terms. Show searchers a wide range of descriptive terms and thereby implicitly educate them on the need to produce variety." ([9], pg. 361)

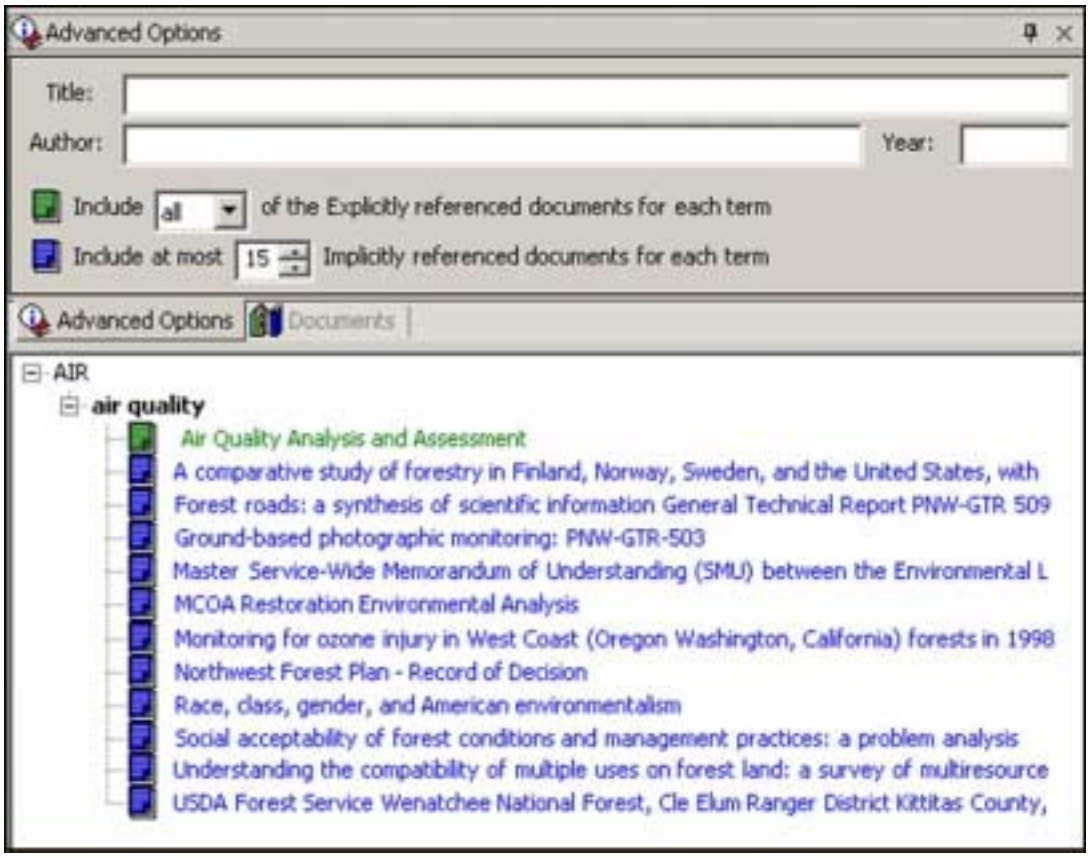


Figure 2-12: Cropped screenshot of Search window

Metadata++ adheres to this philosophy. Instead of trying to guess what the user intended, our system makes it easy and intuitive for the user to interactively expand the search with a variety of terms. Instead of typing in free-text search terms, searchers must select path-based terms from the hierarchy; but the user may type any search string into the Find window to find possible path-based terms that may then be added to the search. Similar to selecting terms as keywords during indexing, searchers may drag any term from the Browse window or the Find window and drop it on the Search window. Dropping the term on the Search window will automatically show the results for that term, which include both explicitly referenced documents (in green) and implicitly referenced documents (in blue). Figure 2-12 shows the Search window after dropping the *AIR\air quality* term into the search. Research [52,86] shows that selecting search terms from a controlled vocabulary is useful for expressing searches and improves search consistency.

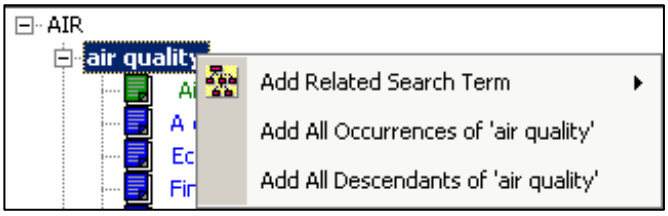


Figure 2-13: Cropped screenshot showing the context menu for Search term

Additional research shows [9,20,35] that thesaurus-based search expansion is effective at improving search results. Various automatic thesaurus-based expansion algorithms exist, but further research [20,56,63] suggests that interactive user-driven query expansion can be more effective and preferable when compared to automated query expansion. Metadata++ supports the user in interactively expanding the search. After right-clicking on any search term in the Search window, the searcher will see a popup context menu (illustrated in Figure 2-13). The first item in this menu will display a popup tree containing the multiple occurrences, synonyms, and related terms for the selected term (as shown in Figure 2-7). This popup tree provides suggested terms that may be of interest to the searcher based on their relationship to the current search term. If desired, the searcher may click on any term in popup tree and that term will automatically be added to the search.

The second item in the context menu shown in Figure 2-13 provides a convenient way for adding all multiple occurrences of a term. The user could manually click on each multiple occurrence as displayed in the popup tree – but if the searcher just wants to quickly add all multiple occurrences, he or she can do so by selecting the "Add All Occurrences" item on the context menu. For example, suppose the searcher added *AQUATIC\Aquatic Biology\Riparian* as a search term. If desired, the searcher could right-click and choose the second menu item to easily add all other occurrences of the term *Riparian* (as shown in Figure 2-4). The third menu item works in a similar fashion; it provides a quick and easy way to expand a search to include all narrower terms of the current search term. In addition to adding search terms, the user may easily remove an unwanted search term by simply deleting it from the Search window (click on the term in the hierarchy and press the <DELETE> key). Using the hierarchy of terms and intuitive

user-interface features, Metadata++ makes it easy for searchers to interactively expand searches to explore a variety of related documents.

2.3.2 Search Result

Metadata++ uses the hierarchy of path-based terms to display the actual search result (as show in Figure 2-12). Petrelli et al. [90] determined that users expect to see a ranked list of documents and anything else is considered annoying. However other research [9,31,73] indicates that showing search results in context of the subject hierarchy increases user satisfaction. By showing the search results in context of the hierarchy, the searcher can infer the relevance of any particular document by seeing where that document appears in the search results. For example, suppose the searcher added all of the descendant terms of a particular search term. Documents related to the immediate child terms may be considered more relevant than documents related to other descendant terms that are several generations below the original search term. A document related to more than one search term (and thus appearing in more than one place in the search result) might have a greater degree of relevance. Viewing the documents in context of the hierarchy gives a clearer picture of why and how any particular document relates to the search – as opposed to a ranked list where the user does not necessarily understand why or how documents are ranked.

In addition to the hierarchical search result, the user may also choose to view a simple list of documents. This list (as illustrated in Figure 2-14) shows the same set of documents that appear in the hierarchical view sorted by count (the number of times the document appears in the hierarchical view). For example, Figure 2-14 shows the document list for a search containing two search terms: *AIR\air quality* and *AIR\Weather*. Two of the documents (the first two in the list) are implicitly related to both terms. In the hierarchical view, these documents appear twice – once under each term. In the list view, these documents appear only once, but with a count of 2.

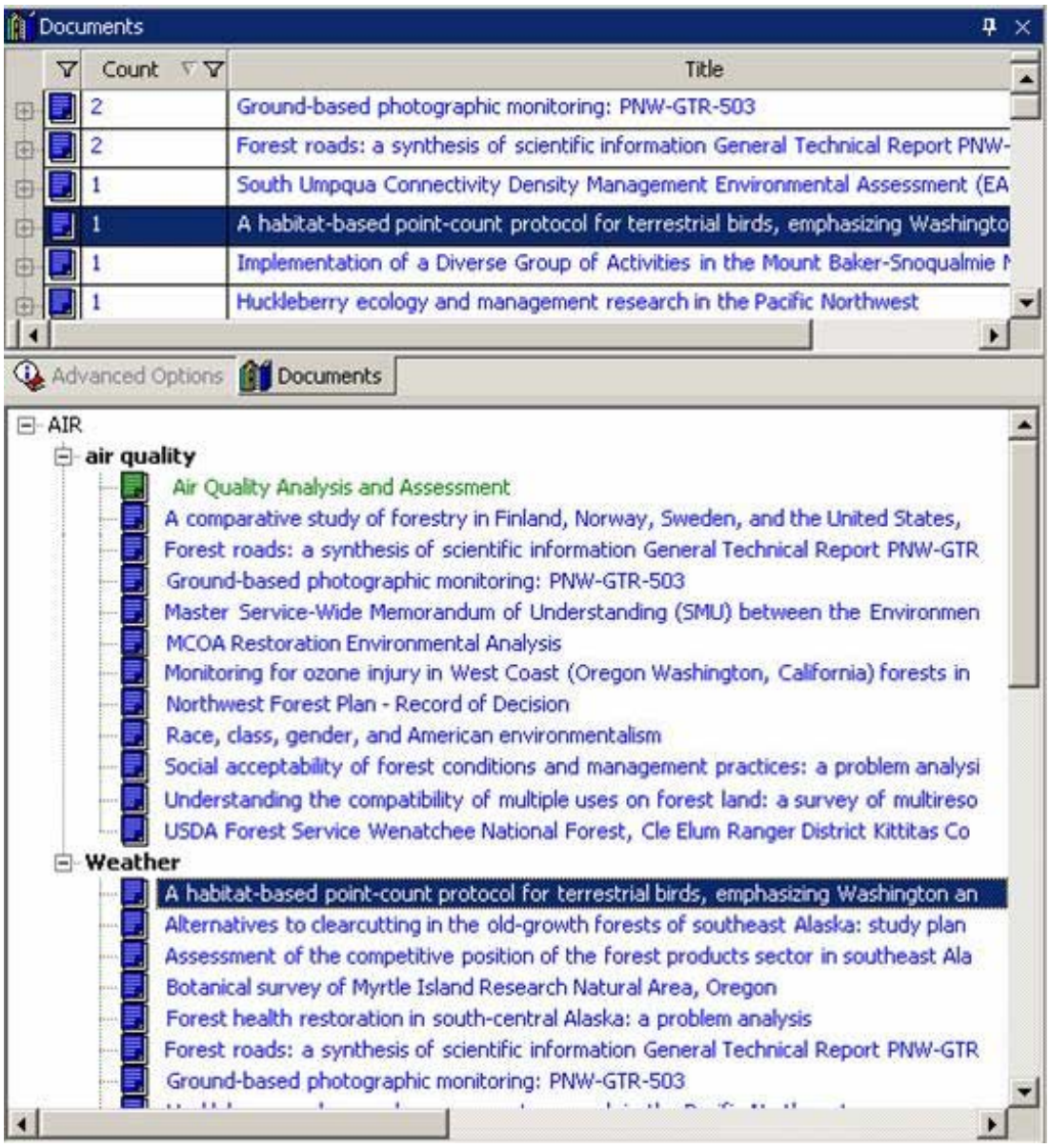


Figure 2-14: Screenshot of Search result document list

The list of documents can easily be sorted by count or title simply by clicking on the corresponding column header. If desired, the searcher may also filter the list. For example, to see documents that appear at least twice in the list, the searcher would simply filter the list for rows where count is greater than or equal to 2. As with the Document-View Pattern in software architecture, the hierarchical view and the list view are views of the same search result. For example, if the searcher clicks on any document in the hierarchical view, that same document is automatically selected in the list view (as shown Figure 2-14) and vice versa.

In addition, the searcher may choose to delete terms (and associated documents) from the search result. For example, suppose the searcher adds all descendants for a particular search term. After reviewing the result, the searcher may decide that only some of the descendants apply to the search topic. He or she may click on the inapplicable terms in the hierarchical view and press the <DELETE> key on the keyboard. That term (and all of its descendant terms and documents) will be deleted from the search result. The searcher can add (or remove) search terms at any time during the search process – providing an interactive and exploratory experience that helps the searcher find relevant information.

2.3.3 Advanced Options

Most information retrieval systems provide the ability to filter search results based on document summary information [9], such as title, author, and date. As illustrated in Figure 2-12, the user may use the advanced options to pre-filter the search results. For example, suppose a searcher is only interested in documents authored by Timothy Tolle. The searcher would type 'Tolle' in the Author box before adding search terms to the search. When the system displays the search results, only those documents where Tolle is an author (i.e., where 'Tolle' is contained in the author property) will be displayed.

As described in Section 2.1.2.3, Metadata++ combines the advantages of human-indexing (explicit keywords) and machine indexing (implicit keywords). The default search result will include all explicitly referenced documents for each search term (shown in green in Figure 2-12) and will include no more than fifteen implicitly referenced documents (show in blue in Figure 2-12) for each search term. Figure 2-12 shows that only one explicitly referenced document exists for *Air\air quality*. The remaining documents are implicitly referenced (meaning the documents actually contain the phrase 'air quality'). These options are configurable – so the user may choose to ignore implicitly referenced documents altogether (by changing 15 to 0) or to ignore explicitly referenced documents (by changing 'All' to 'None' using the drop-down list).

2.3.4 Compound Search

A search task within natural resource management typically involves more than one concept [8]. For example, a natural resource manager may be trying to ascertain the

overall water quality within a particular region. This task would comprise two different search concepts: water quality and the region of interest. Users may perform compound searches in Metadata++ by interactively expanding each individual search concept (as described in the previous section) then combining the results of each concept into a compound search.

Figure 2-15 illustrates the results of a simple compound search involving *AIR\air quality* and *AIR\Weather*. We produced this screenshot by first creating a Search window and adding only the *AIR\air quality* term. Next, we created a second Search window and added the term *AIR\Weather*. We could have further refined each search concept by adding (or deleting) additional search terms. Finally, we intersected the two Search windows, which produced the visible tab shown in Figure 2-15, labeled "air quality & weather".

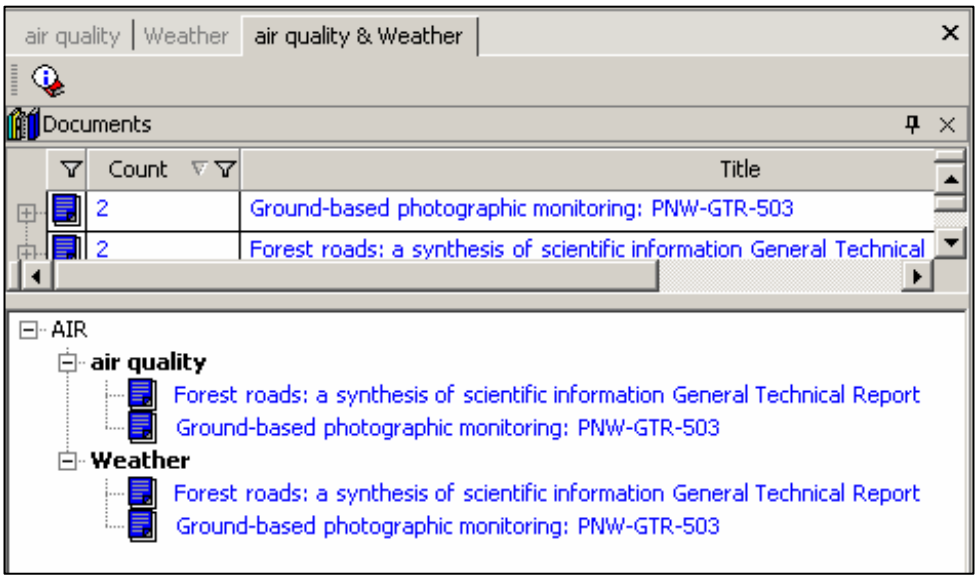


Figure 2-15: Cropped screenshot of a compound search

When doing an intersection, Metadata++ combines the hierarchical structure from both existing Search windows – then removes all documents that do not appear in both concepts. As shown in Figure 2-15, only those documents that were in both original Search windows (the first two tabs) appear in the compound Search window (the third tab). The current Metadata++ application only supports intersection – but the Metadata++ conceptual model supports two additional compound operators (as defined in Section 4.2.4):

- *Union*: include all path-based terms and documents from both searches
- *Difference*: include all path-based terms from the first search, but remove all documents that appear in the second search

2.4 Summary

The Metadata++ Digital Library is based on a hierarchy of path-based terms found in numerous controlled vocabularies commonly used within natural resource management. The user may peruse vocabularies by browsing the hierarchy of terms or by typing the desired word(s) into the Find window. Additional information for each term, available by right-clicking the term, includes: multiple occurrences, synonyms, related terms, explicitly referenced documents, and implicitly referenced documents. The user may select any number of path-based terms during indexing or searching by dragging the desired term onto the appropriate window. Both indexing and searching retain the full path of the selected term, so the user will always see terms within the context of the hierarchy. The hierarchy of path-based terms provides an intuitive framework for both indexing and searching.

3 User Study and Evaluation of Metadata++

This chapter describes a user study and evaluation of the Metadata++ digital library system conducted as part of our research. Section 3.1, outlines the objectives of our study and is followed by a detailed description of the study in Section 3.2. Section 3.3 summarizes the results and observations. The chapter concludes with a review of other digital library evaluations in Section 3.4.

3.1 Objectives

Marchionini [71] states that evaluating a digital library is like judging the success of a marriage – "[much] depends on how successful the partners are as individuals as well as the emergent conditions made possible by the union." A digital library is essentially the union of a library (content) and a digital information retrieval system, where "success for an individual partner is typically necessary but not sufficient to ensure success for the marriage" [71]. So how, then, does one effectively evaluate a digital library? While no universal set of criteria or methods exist for digital library evaluation, significant research has been done in this area. Saracevic [92] explains that evaluation criteria should be divided into three categories: library criteria, information retrieval criteria, and human-computer interaction (HCI) criteria. Library criteria include such things as the quality, quantity, and scope of the content – similar to measuring the holdings of a physical library. Information retrieval criteria focus on relevance issues – such as precision and recall. HCI criteria focus on the user interface, including such things as usability, functionality, navigation, browsing, etc. An effective evaluation must begin by clearly defining which criteria will (and will not) be evaluated and the metrics to be used for each criterion.

Buttenfield [15] points out that, from the users' perspective, the user interface of a digital library *is* the system. If the user does not succeed at using the application, then the

contents of the library are inaccessible – regardless of the value or quality of those contents (and the relevance achieved by the information retrieval techniques). Landauer [64] claims that usability (ease of operation) and usefulness (serving the intended purpose) are important aspects of any human-computer interface. He also explains that usability and usefulness are closely related and hard to evaluate separately. Based on the scope of our user study, we selected two evaluation criteria that would help us: (1) determine the overall usability of the system based on the level of user interaction and number of errors, and (2) evaluate the path-based representation of terms – particularly multiple occurrences. Our objectives did not include evaluating the content of the system (vocabularies and documents) or measuring the retrieval performance (precision and recall). The results of our evaluation are explained in Section 3.3

3.2 Description of User Study

Our research team member from the USDA Forest Service took the lead on defining four tasks based on realistic information needs and existing documents in natural resource management [12]. He also worked closely with a library scientist to prepare the test system to ensure that sufficient vocabularies and documents (with keywords) existed to allow completion of the tasks. Summarized in Table 3-1, these tasks included two searching tasks and two indexing tasks that simulate real world information needs that the participants might encounter during daily activities.

Table 3-1: Summary of User Tasks

Search 1	You are the Planning Team leader of the Cle Elum Ranger District charged with updating the existing (upper) Yakima River Watershed Analysis. You want to provide your interdisciplinary team (IDT) with a list of relevant documents. In the first WA, watershed restoration projects were identified as the priority task needed. Your line officer feels it is time to tackle the next priority of projects. She assigned a forester, recreation specialist, watershed restoration specialist, wildlife biologist, aquatic specialist, zone transportation planner and archaeologist to the IDT and a public affairs officer to work with the team leader. Find documents that are most likely to help members of your interdisciplinary team.
Search 2	You are a fish biologist charged with developing an aquatic monitoring and evaluation program for the Cle Elum Ranger district. You are responsible for tying together your monitoring data with that of the regional Ecosystem Office's monitoring effort, the Wenatchee NF's Forest Plan effort and any research, inventory and monitoring questions of the Snoqualmie Pass Adaptive Management Area, as well as project monitoring needs and activities and needs identified through watershed analyses. District monitoring goals come from these sources, though the scale of evaluation is for the ranger district itself. Find relevant material to help you develop such a program for the Cle Elum ranger District.
Index 1	Title = Big Bend Notice of Intent Abstract = The Forest Service, USDA, will prepare an environmental impact statement (EIS)

	to analyze and disclose the environmental impacts of a site-specific proposal to issue a permit and authorize use of an existing road and construction of additional road across National Forest System (NFS) lands located in the Little Creek Drainage. The action is proposed in response to an application seeking legal access to approximately 885 acres of non-Federal land located within the Forest Boundary on the Cle Elum Ranger District. The proposed action is located approximately 8 miles south of Cle Elum, Washington. The purpose of the EIS will be to develop and evaluate a range of alternatives including a no action Alternative, and possible additional alternatives to respond to issues identified during the scoping process. The proposed project will be in compliance with the direction in the Wenatchee National Forest Land and Resource Management Plan (March 1990) which provides the overall guidance for management of the area. The agency invites written comments on the scope of this project. In addition, the agency gives notice of this analysis so that interested and affected people are aware of how they may participate and contribute to the final decision. The major issues that have been identified to date include the following: water quality; Late Successional Reserves; habitat for spotted owl, peregrine falcon and the gray wolf; cumulative effects of both the road access and the management of the private land and Forest Service.
Index 2	Title = MCOA Restoration Environmental Analysis Abstract = It is proposed that a mining site previously used in accordance with the Mining Claim Occupancy Act (MCOA) be restored to forest conditions. The proposed action would restore the site towards a forest setting by burning approximately 30 handpiles, scarifying two structure pads and the network of natural surfaced roads, seeding scarified areas and removing the PacifiCorp power poles. The proposed action affects BLM lands in the Middle Applegate watershed.

As recommended by Borlund [12], we prepared for the user study by doing a pilot test. The pilot test participant was a computer science PhD Candidate, as well as an M.D. She had significant knowledge and experience with information retrieval systems, including thesaurus-based systems such as UMLS [83], but she was not an expert in natural resource management. Using a laptop computer without an external mouse hindered the participant during the test but she was able to complete the test and provide feedback about possible improvements. In addition to the pilot test, the system was reviewed by team members (including a natural resource manager, a library scientist, and a computer scientist) and modified according to their feedback.

In preparation for the user study, we selected two different USDA Forest Service offices that expressed interest and willingness to participate. Each office had a primary contact person who assisted us in coordinating the tests, including soliciting their colleagues to find people interested in participating. Each office coordinator designated four participants (including himself or herself) for a total of eight participants (four at each office). The participants come from a variety of backgrounds but all work within natural resource management on a daily basis [12,15]. Based on demographic questionnaires, Table 3-2 summarizes the daily responsibilities of each of the eight participants along with the type of tasks they completed during the user study and the

number of years of experience with the Forest Service (a combined total of 152 years of Forest Service experience!).

Table 3-2: Summary of Test Participants

Task	Forest Service Responsibilities	Years of Experience
Searching	Editing, document syntheses, report writing, cultural resources	20
Searching	Wildlife research/studies, GIS	7
Searching	Manage budget, guide scope of work, maintain relations with other organizations, watershed counsels	18
Searching	Data steward, data analysis, data quality	15
Indexing	Soil science research specialist	7
Indexing	Planning, Special Use Administration, NEPA document reviews, minerals administration	28
Indexing	Manage and coordinate NEPA, Appeals, Litigation & FOIA (Review documents, provide advice, develop administrative records in the event of the litigation, convey policy & direction to the field)	28
Indexing	Forest planning, hydropower re-licensing	29

At each office, we introduced the users to the Metadata++ system in a training seminar the day prior to the actual user tests. All but one of the participants attended the training session. We began the training session with a group presentation that emphasized the objectives of the user study, and well as what the participants could expect during the study. We showed them a demonstration of the system, followed by a question and answer session. We then gave each user the opportunity to use the system and become familiar with the software. Before the conclusion of the training session, each user filled out a demographic questionnaire regarding educational background, professional experience, and indexing and searching skills. At the end of the session, we asked each user to choose a preferred role for the study: either an indexer or a searcher. Each user who expressed a preference between roles participated according to his or her preference. The other users (who did not express a preference) where designated such that we had two indexers and two searchers at each of the two offices.

We conducted each test session in a conference room containing the test participant, a moderator (the author of this dissertation), and three observers. Each observer was a member of the research team with a different expertise (natural resource management, library science, and computer science). The same moderator and observers participated in all of the user study sessions.

Ideally, we would have conducted each session in the participant's own workspace – but doing so was logistically impractical. The coordinator at each office arranged for a computer in the conference room for testing purposes. At both locations, we discovered technical problems with running the application on the office computer – including missing or out-dated operating system components. We decided to use a laptop that we had prepared before-hand as a backup plan. The laptop belongs to a team member from Denmark so the keyboard and operating system were set up for Danish users. Several comments from test participants referred to the differences in the keyboard (compared to the keyboards they were familiar with).

Each of the eight users participated in a two hour session during which he or she completed either two search tasks or two indexing tasks (as described above). Each session began with a review of the system and an introduction to the tasks for that session. Each of the four indexers received a hard-copy of the two documents to be indexed (one at a time) and was then asked to review the document and select the concepts they felt were important for that particular document. They then used the Metadata++ system to select the keywords that best represented the concepts they had outlined for that document. Each of the four searchers was given the same two search tasks (one at a time). They received a written explanation of the task and were asked to review the task and outline the concepts they felt were important for that particular search task. They then used the Metadata++ system to select the terms that best represented the concepts they had outlined for the search task.

Each participant wore a microphone headset, and we instructed each participant to "think aloud" throughout the entire session [49,71]. Speaking one's thoughts is quite unnatural – and the participants occasionally became quiet [15,56,71]. The moderator frequently reminded users to vocalize what they were thinking and why they were doing each action. The remarks were recorded and later transcribed for further analysis. In

addition to the audio recordings, each of the three observers made detailed written notes through each session. In addition to written observations and verbal remarks, the application automatically logged each user action [15]. Upon completion of both tasks, each user participated in a follow-up interview [71]. During the interview, we asked specific questions regarding their use of the system, and they were invited to make additional comments.

3.3 Test Results and Observations

During the user tests, we gathered data in three ways: application log files, audio recording, and observers' written notes. We later transcribed the audio files – and compiled the log files, transcribed audio, and observations into a single chronological transcription of each user test. This compilation required timestamp adjustments to account for the differences between system clocks on the observer laptops, the test laptop, and the transcribed audio. An excerpt of the compiled transcription is show in Table 3-3. Combining data from a variety of methods gave us a clear understanding of what transpired [15]. We used these transcriptions for further analysis, both quantitative and qualitative.

Table 3-3: Excerpt of Compiled Transcription

LOG	10:36:33 AM	GetImplicitDocumentsAdvanced [Biological reference points]
LOIS	10:36:35 AM	chooses gap analysis
AUDIO	10:36:37 AM	looking for something that refers to monitoring; gap analysis; especially for fisheries monitoring.
TIM	10:36:42 AM	Then, moved to something that referred to monitoring within aquatic monitoring: gap analysis, and fisheries monitoring. And looked at monitor …
LOIS	10:36:45 AM	looking for monitoring type things

3.3.1 Usability

The first objective of our user study was to evaluate system usability – meaning the users' level of understanding, interaction, and ease of use. Though usability is largely subjective, we defined both qualitative and quantitative metrics for measuring usability. Qualitatively, we relied on user comments and feedback throughout the test and follow-up interviews. Our quantitative measure of usability consists of two parts: (1) the amount of interaction (based on application logs); and (2) the number of errors encountered by

the user. When counting the number of errors, we included both application errors (bugs, crashes, etc.) and incidents where the system differed from user expectations. We also counted the number of times users requested additional features in the application.

Table 3-4: Summary of Interactive Action

Action	Indexers	Searches	Total
Expand Term (Browsing Hierarchy)	98	83	181
Find Term	119	36	155
Get All Descendants	0	8	8
Right-click (Browse Explicit Documents)	20	180	200
Right-click (Browse Implicit Documents)	20	172	192
Right-click (Browse Related Terms)	20	36	56
Total	277	515	792

Table 3-5: Summary of Errors & Requests

Type	Number
Application Errors	6
Incidents	15
Feature Requests	22

Table 3-4 summarizes the exploratory interactions based on data from the application logs. Each of the actions listed in the table corresponds to an interaction described in Chapter 2. Indexers averaged approximately 70 interactions, while searchers averaged approximately 130 interactions. The overall average was 50 interactions per user per task. These numbers illustrate that users did frequently interact with the system while performing the information tasks. Table 3-5 summarizes the number of errors and feature requests. The errors included things such as scroll bars not working and other issues with the application. Incidents occurred when users thought something should happen a certain way, but the application did not respond according to their expectations. For example, one user expected to delete a search term by right-clicking (because right-clicking to delete is common in many computer applications). Another user was color

blind, so he could not distinguish between green (explicitly referenced) and blue (implicitly referenced) documents. Several users requested additional features, such as saving search results for later use and canceling a Find Term request. Overall, the quantitative measurements show a high level of user interaction and a low number of errors encountered by users. Usability of the system can be qualitatively described with comments from users including:

> "[I] like synonyms and like [the] tree showing context. [I] like right-clicking."
>
> "[I] likes the hierarchy but [I'm] not lost without it as [I am] used to not having it."
>
> "I do like how [the search result] defines the term rather than a Google search which just comes up with a bunch of documents. Lot easier and better to sort through."
>
> "I like the hierarchical system. It works."
>
> "Expand up and down is good. Management is different than specialists who want the detail, like an individual species."
>
> "[Multiple Occurrences] helped me think about it in various ways. It was easy to understand."

Our study indicates that users were able to understand the system and interact with it effectively. Several comments were made regarding content of the hierarchy – particularly terms that users expected to find somewhere in the hierarchy but did not. Overall, we believe the system is usable and effectively assists the user in their information tasks.

3.3.2 Multiple Occurrences

In addition to usability, we also evaluated how well the user understood and used multiple occurrences (see Section 2.1.2.1.1). Qualitatively, we reviewed user comments and feedback given during the user tests and follow-up interviews. Our quantitative metric includes: (1) how often the user encountered multiple occurrences, and (2) how the user handled each multiple occurrence.

During the various tasks, participants encountered a total of 657 unique terms with multiple occurrences. The majority of these terms (78%) had exactly 2 occurrences, but several terms had 3 or more occurrences. The maximum number of occurrences for a single term was 27. The distribution of terms and the number of occurrences is illustrated in Table 3-6.

Table 3-6: Multiple Occurrences Encountered During the User Study

Number of occurrences	Distinct terms	%
2	513	78.0
3	88	13.3
4	24	3.6
5	11	1.7
7	7	1.1
6	4	0.6
8	2	0.3
11	2	0.3
14	2	0.3
9	1	0.2
10	1	0.2
15	1	0.2
27	1	0.2
Total	657	100

When presented with multiple occurrences, participants knowingly selected the paths that were most appropriate for the particular task. In some cases, they selected a path, then later removed the path and selected a different occurrence of the same term – one that more accurately described the concept that they wanted to express. Most often the different participants selected the same path to describe a concept, but in a few instances they selected different paths for the same concept. User comments regarding multiple occurrences include:

> "Multiple occurrences may be confusing, but they point to new viewpoints"
>
> "The top term defines the meaning – visual reference"
>
> "[It's] important to be able to distinguish between jargons"
>
> "People might interpret the organization [of multiple occurrences] differently. Not all will agree in the organization"

The findings indicate that participants easily understood the differences between multiple occurrences of the same term based on the hierarchical path for each occurrence. The different perspectives and connotations of terms in Metadata++ were consistent with their domain knowledge – and the distinctions were considered to be important and

meaningful. Although the participants occasionally disagreed on the organization of multiple occurrences, they understood and accepted the differences. The participants also emphasized that multiple occurrences should represent clearly distinct connotations of the term. In several cases the multiple occurrences of terms inspired the searchers to try out a variety of search terms for each searching task [9]. Participants expressed how multiple occurrences illustrate viewpoints they had not previously considered – and provided ways to restrict or expand the search.

3.4 Related Work

As digital libraries become more prevalent, research regarding digital library evaluation is becoming more prolific. Various studies have been reported with different goals in mind, ranging from log-based analysis of search behavior [69] to holistic system design [15,71]. This section describes some other digital library evaluations and compares them to the Metadata++ user study.

The Perseus Digital Library (PDL) project [71] began in 1987 and is still a well-known online resource for education within the humanities. The original project proposal included an ongoing evaluation component targeted at evaluating the digital library with a specific emphasis on measuring the library's affect on learning. During its lifetime, the PDL has utilized a range of technologies. It began as a HyperCard system distributed on compact disc, and later moved to the World Wide Web. The evaluation component of the PDL project evolved along with the library itself. During many years of evaluation, researchers used a variety of methods – including in-class observations, interviews with educators, participant observations, think alouds, transactions logs, and online surveys. The ongoing evaluation led to important conclusions about the PDL and its affect on learning; but also brought valuable knowledge and experience about digital library evaluation methodology. Our user study has a significantly smaller scope than the PDL evaluation (a few days compared to more than a dozen years!), but we used several of the same methods and found them to be useful.

The Alexandria Digital Library (ADL), a collaborative project involving the University of California at Santa Barbara and the University of Colorado at Boulder, extends traditional topic-based queries by combining spatial and temporal queries.

Buttenfield [15] explains how evaluation is an important component of the entire system life cycle. They performed a number of user evaluations at various stages of system design and deployment. These evaluations involved methods similar to those used in our user study – including videotape (we used audio), automated logs, and exit surveys. Buttenfield states the importance of using multiple methods and combined data gathered from each method. For example, combining audio or video from talk alouds with automated transaction logs allows you to compare what the users *say* they are doing with what they are *actually* doing. This comparison often reveals users' misconceptions of the application. Buttenfield describes another benefit of using convergent methods: comparing data from multiple methods lets you not only evaluate the digital library, but also evaluate the methods themselves. When findings from one method confirm and substantiate findings from a different method, it builds confidence in the methods themselves. We noticed similar findings during our study. Combining observers' notes, audio transcriptions, and application logs reinforced our understanding an interpretation of what transpired during each user test.

NewsLink is a manually indexed full-text database containing newspaper articles published since 1994 by a well-known Scandinavian newspaper. Blomgren et al. [10] report a study that focused on both the system perspective and the user perspective. Their study included 20 participants who were all journalists employed by the newspaper. The researchers used demographic questionnaires, search-analysis protocols, post-search interviews, and web surveys to collect data. Blomgren et al. noticed a discrepancy between system-oriented results (including precision) and user-oriented results (including satisfaction). Multiple users reported high satisfaction, even when the system reported poor precision. Many users considered only the first document on the list as relevant – even when multiple documents were found. The general consensus among users was that "the overall most important information sources are oral sources and the Internet," and they only used NewsLink weekly (on average). Our study had smaller scope and focused on usability. We used similar evaluation methods and found high satisfaction among users.

Nielsen [86] reports a study focused on testing a semi-automatic word-association mechanism for thesaurus construction. Rather than an evaluation of the entire digital

library, this study evaluated the thesaurus generated by the word-association method in comparison to an existing thesaurus constructed in a traditional manner. Participants included twenty randomly selected researchers from the Research & Development department of a large pharmaceutical company. These participants regularly use the control thesaurus as part of a digital library that contains drug-related information. During the test, participants completed three realistic search jobs using the same user interface as they typically use – only with the experimental thesaurus. Data collection methods included questionnaires, observation, logging, and post-search interviews. The study found only slight differences in satisfaction between the word-association thesaurus and the control thesaurus. In both cases, users reported a need for the thesaurus during searching. Our study focused on the overall usability of the application, and the features of the thesaurus, whereas this study focused on the content of one thesaurus in comparison to another. Both studies used similar data collection methods – and both studies found that users like having a thesaurus to assist in query expansion.

3.5 Summary

We designed and conducted a user study to evaluate the usability of the Metadata++ Digital Library and determine how well users comprehend path-based terms. Users consisted of eight USDA Forest Service employees (four from each of two different offices). Each user completed either two indexing tasks or two searching tasks. A moderator and three observers conducted each user study session and gathered data using automated logging, transcribed audio recording, and written observer notes. The data show a high level of user interaction as well as a strong comprehension of path-based terms for indexing and searching. Overall, users gave a very positive response to the system and its usability within the natural resource management domain.

4 The Metadata++ Model

This chapter formally describes and defines the Metadata++ conceptual model (as described informally in the description of the Metadata++ application in Chapter 2). In Section 4.1, we discuss alternative model features that we considered during the course of our research, but excluded from the current Metadata++ conceptual model. Section 4.2 defines the current conceptual model using set theory and first-order logic. In Section 4.3, we compare the Metadata++ model to other formal thesaurus models.

4.1 Alternative Features

Over the course of our research, we discussed and implemented several features that are not included in the current conceptual model. This section describes these alternative features and explains why we exclude them from the current model.

4.1.1 Properties vs. Terms

During the early days of our project, we conceptualized terms, documents, and properties all as first-class objects [111]. The associations and relationships in this model connected properties, terms, and documents to express metadata. Associations were triples that related documents with properties and terms as illustrated in Figure 4-1. For example, the document in the bottom right was authored by Parley Pratt – so an association (solid line) exists between the document, the AUTHOR property, and the term 'Parley Pratt'. The document in the bottom left was edited by Parley Pratt, so a different association exists between that document, the EDITOR property, and the same term ('Parley Pratt'). An association could also be defined between terms – as illustrated by the 'Willamette National Forest' term associated with the 'McKenzie Ranger District' term by the SPATIAL CONTAINMENT property (because the ranger district is spatially contained within the national forest).

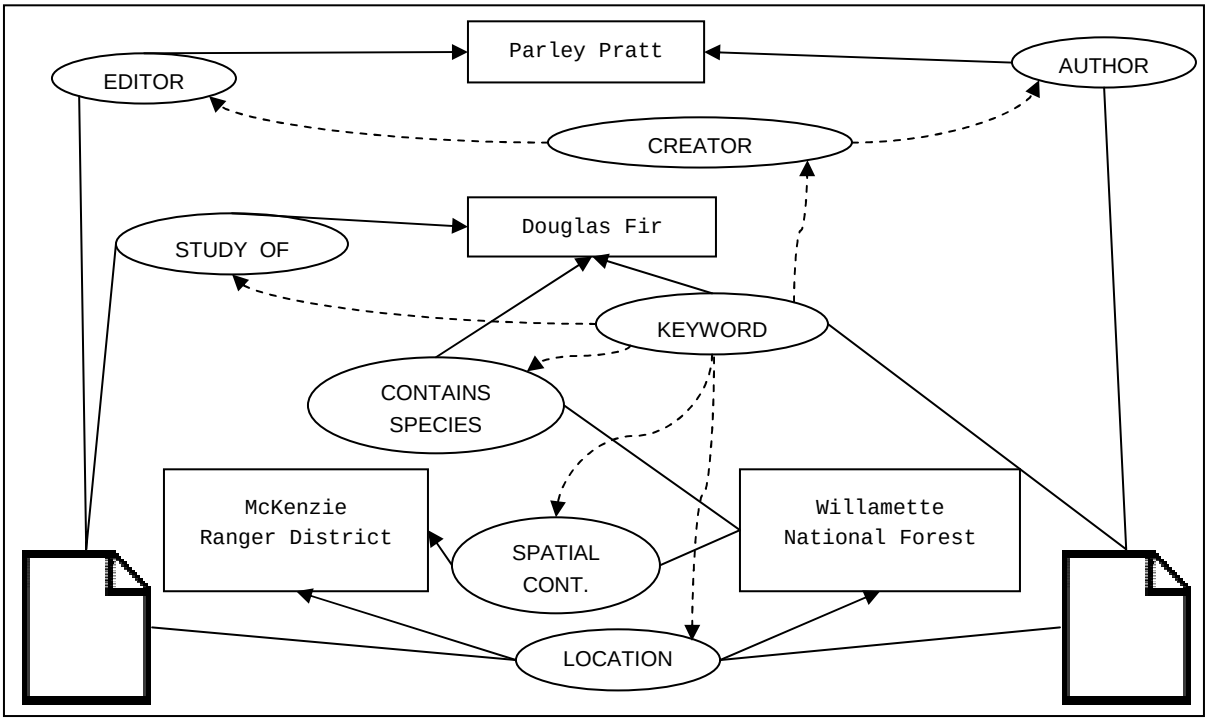


Figure 4-1: Associated terms and related properties
[Associations = solid line; Relationships = dashed line]

Figure 4-1 uses dashed lines to represent relationships, which are directed edges between properties. For example, the AUTHOR property and the EDITOR property are both more specific instances of the CREATOR property – so the CREATOR property has outgoing (i.e., more specific) relationships to both EDITOR and AUTHOR. The top-level property is KEYWORD – of which all other properties are more specific descendants. For example, the KEYWORD property is the parent of more specific properties including STUDY OF and LOCATION. Any arbitrary number of properties could be introduced into the system using the hierarchical relationship among properties. Properties generalized typical metadata such as Dublin Core [30] and provided a uniform way to associate terms with other terms and to associate terms with documents. Over time, we discovered that this model was too complex for practical users. It was tedious and unintuitive for users to select both properties and terms – because the term typically implied the property. For example, McKenzie Ranger District is a place, so selecting 'McKenzie Ranger District' as a keyword for a document implies that the document is relevant to that location (explicitly selecting the LOCATION property is unnecessary).

We eventually unified properties and terms into a single hierarchy (as defined by the current Metadata++ model).

4.1.2 User Perspectives

We explored the possibility of allowing individual users the ability to construct their own individual hierarchies out of the same set of shared terms. Each hierarchy, or perspective, could be entirely different from others. Perspectives allowed users to organize a subset of terminology for their own use. For example, Figure 4-2 illustrates four different user perspectives. Parts (a) and (b) both consist of location terms (geographic place names) arranged in different hierarchies. User 2 divided the location hierarchy into two branches, Authorizing Agency and Proposing Agency. This distinction implied a particular meaning for a selected term – depending on which branch it was selected from (similar to explicit properties as explained the previous section). Parts (c) and (d) both show similar terms about air conditions, but the climatologist arranged the terms one way and the forester arranged the terms a different way.

Perspectives allowed users to browse the terms in whatever way was convenient and intuitive to them – without being affected by how other users might choose to browse the same set of terms. When creating their own perspective, each user (or type of user) would select terms from the same set of terms. The terms existed independently of the perspective(s) in which they were found – so one term had the same meaning in every instance. For example, parts (c) and (d) of Figure 4-2 both contain the term ‘smoke’. Regardless of where the term ‘smoke’ appeared in each hierarchy, it was still the same term (i.e. ‘smoke’ is ‘smoke’, regardless of how or where it is used).

Documents were attached directly to terms – independent of perspectives. So when the climatologist attached a document to the term ‘smoke’ (because ‘smoke’ is a keyword for the document), that same document would automatically appear beneath ‘smoke’ in the forester perspective as well (even though the perspective contains ‘smoke’ in a different location in the hierarchy).

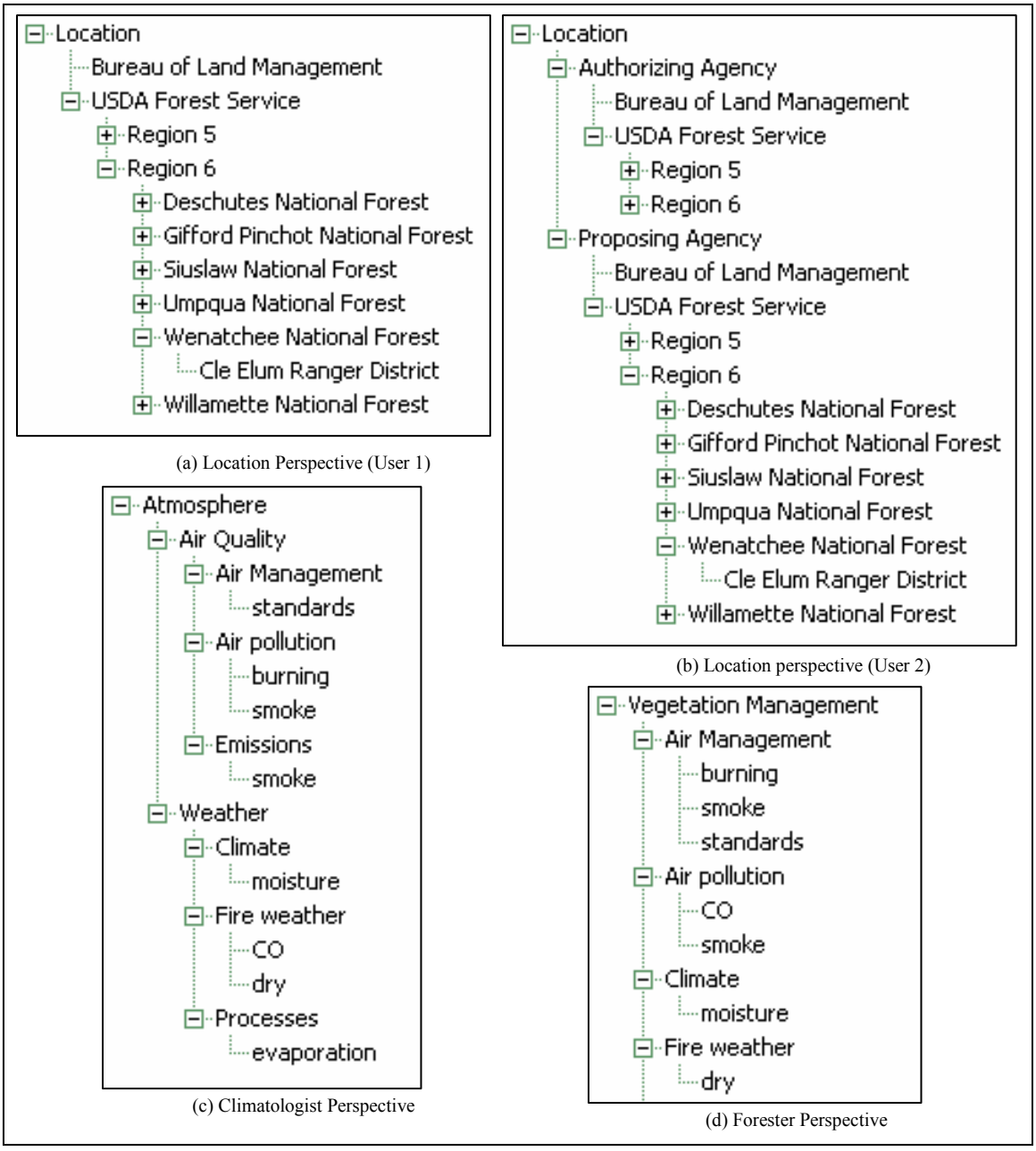


Figure 4-2: User Perspectives

Upon further investigation, we learned that users were not necessarily interested in creating their own personal perspectives – however they were interested in using their own subject-matter controlled vocabularies "as-is". As described in Chapter 1, each group of users has their own way of describing things. Climatologists describe things differently than foresters. By combining all of the controlled vocabularies into a single hierarchical structure, we accommodated the users' needs. We also discovered that 'smoke' is not necessarily 'smoke' – that is to say that the same term within different

vocabularies may have different connotations. For this reason, we invented multiple occurrences (as described in Section 2.1.2.1.1).

4.1.3 Templates

Many document managements and digital libraries include templates for creating document metadata. We explored various ways of integrating templates with the hierarchy of terms. Although the Metadata++ application does prompt the author (or indexer) to select terms from 'required' vocabularies (see Section 2.2.3), this feature is not fully defined and is not included in the formal conceptual model for Metadata++.

4.2 Formalization of Model

This section defines the current Metadata++ conceptual model using set theory and first order logic (with recursion). The sets, axioms, and functions defined in this section work together to express a precise conceptualization of Metadata++.

4.2.1 Sets

The sets defined in this section constitute the fundamental domains with the Metadata++ formalization.

$$T = \text{finite set of terms}$$

$$D = \text{finite set of documents}$$

The elements of set T are terms – words or phrases found in the various controlled vocabularies from the application domain. An element in set T is just the term – it has no path or context. The set D contains documents, and each document (regardless of file type, layout, format, etc.) has its own metadata including a collection of explicit keywords. The sets T and D, by themselves, provide no structure, organization, metadata, or association of terms or documents; they are simply sets of terms and documents that exist within the system.

$$N = \text{finite set of nodes}$$

$$E = \{ (x,y) \mid x \in N \wedge y \in N \wedge x \neq y \}$$

$$E \subseteq N \times N$$

$$M = \{ (x,y) \mid x \in N \wedge y \in (T \cup D)\}$$

$$M \subseteq N \times (T \cup D)$$

The hierarchical structure of controlled vocabularies is defined by a directed acyclic graph of nodes, where each node maps to either one (or more) term(s) or a document. N is a set of nodes – and E is a set of directed edges (ordered pairs of distinct nodes). The set M defines a mapping – where each element is an ordered pair consisting of a node paired with either a term or a document. Mapping nodes to terms defines the path (or context) for the term within the hierarchy. For example, consider the small example illustrated in Figure 4-3 showing the path *Air\air quality*.

```
T = {'Air', 'air quality'}
N = {n1, n2}
E = {(n1, n2)}
M = {(n1, 'Air'),  (n2, 'air quality')}
```

Figure 4-3: Example of Formalization

Mapping nodes to terms also allows for terms to appear in more than one place in the hierarchy (multiple occurrences) and allows for a single node to map to multiple terms (polyterms). Mapping nodes to documents allows the creation of explicit keywords (during indexing) by creating an edge from the term node (the node mapped to the term selected as a keyword for the document) to the document node (a node mapped to the document). For example, suppose an indexer selects a particular path-based term as a keyword for a document. The system would first create a new node and establish a mapping from the new node to the document. The system would then determine the node that maps to that particular path-based term, and create an edge from that node to the newly created node (which maps to the document).

The set S and the set R both contain ordered pairs of distinct nodes – representing synonymy and association (i.e., related terms), respectively. It is important to note that both sets use nodes, not terms, because terms by themselves cannot be related; only nodes within the hierarchy can be related.

$$S = \{ (x,y) \mid x \in N \wedge y \in N \wedge x \neq y \}$$

$$S \subseteq N \times N$$

$$R = \{ (x,y) \mid x \in N \wedge y \in N \wedge x \neq y \}$$

$$R \subseteq N \times N$$

4.2.2 Predicates

This section defines predicates that are used in axioms and functions defined in the following sections. The first seven predicates defined in this section are only for syntactical convenience and correspond to the sets defined in the previous section. For example, Predicate 4-1 is true if and only if $x \in T$.

Predicate 4-1: x is a term

$$T(x) \equiv x \in T$$

Predicate 4-2: x is a document

$$D(x) \equiv x \in D$$

Predicate 4-3: x is a node

$$N(x) \equiv x \in N$$

Predicate 4-4: Edge exists from node x to node y

$$E(x,y) \equiv (x,y) \in E$$

Predicate 4-5: Node x is mapped to y

$$M(x,y) \equiv (x,y) \in M$$

Predicate 4-6: Node y is synonym of node x

$$S(x,y) \equiv (x,y) \in S$$

Predicate 4-7: Node y is related to node x

$$R(x,y) \equiv (x,y) \in R$$

Predicate 4-8: Node x is an ancestor of node y

$$A(x,y) \equiv E(x,y) \vee (\exists z\ E(x,z) \wedge A(z,y))$$

Predicate 4-8 is true if and only if node x is an ancestor of node y – meaning that, starting at node x, directed edges exist that will lead to node y. This predicate is important in the axioms defined in the next section – specifically for preventing cycles within the graph.

Predicate 4-9: TermMatch

TermMatch(s,t): s is a string, t ∈ T
true: if the text of term t contains string s;
false: otherwise;

Predicate 4-10: DocumentMatch

DocumentMatch(t,d): t ∈ T, d ∈ D
true: if the text of term t is found in the content of document d;
false: otherwise;

Metadata++ uses the `TermMatch` predicate when finding terms that match a specified search string and Metadata++ uses the `DocumentMatch` predicate to determine if the text of a particular document contains a specific term. `DocumentMatch` provides a full-text search of the document and might be implemented using any full-text indexing engine. (We used Microsoft® Index Server as explained in Chapter 5.) Both `TermMatch` and `DocumentMatch` use just the term itself without regard to nodes or paths.

4.2.3 Axioms

The axioms defined in this section limit how the elements in each set relate to one another. The axiom definitions use predicates defined in the previous section.

Axiom 4-1: Terms, Documents, and Nodes are disjoint

$$(T \cap D = \varnothing) \wedge (T \cap N = \varnothing) \wedge (N \cap D = \varnothing)$$

The first axiom specifies that the sets of terms, documents, and nodes are all disjoint. A term cannot also be a document, nor can a document be a term. A node cannot be a term or a document.

Axiom 4-2: All nodes have an edge

$$\forall x\ N(x) \rightarrow \exists y\ E(x,y) \vee E(y,x)$$

Axiom 4-3: All nodes have at most one incoming edge

$$\forall x \forall y \forall z\ E(x,y) \wedge E(z,y) \rightarrow x = z$$

Axiom 4-4: No node is an ancestor of itself

$$\forall x \forall y \; A(x,y) \rightarrow x \neq y$$

Metadata++ restricts the directed graph (defined by the nodes in set N and the edges in set E) to be a forest (i.e., hierarchy). Axiom 4-2 specifies that every node must participate in at least one edge. This axiom prevents nodes that are disconnected from the rest of the graph. Axiom 4-3 states that a node can have at most one incoming edge – a node cannot have more than one parent node. Forcing nodes to have at most one incoming edge does not completely eliminate the possibility of cycles. If every node had exactly one incoming edge and exactly one outgoing edge, the graph would be cyclic. Axiom 4-4 eliminates cycles by preventing any node from being an ancestor of itself. These axioms work together to properly define the hierarchy within Metadata++.

Axiom 4-5: All nodes have at least one mapping

$$\forall x \; N(x) \rightarrow \exists z \; M(x,z)$$

Axiom 4-6: All terms have at least one mapping

$$\forall x \; T(x) \rightarrow \exists z \; M(z,x)$$

These axioms place necessary restrictions on the mapping between nodes and terms. Axiom 4-5 requires that every node must have at least one mapping and Axiom 4-6 requires that every term must have at least one mapping.

Axiom 4-7: Nodes mapped to documents may not have outgoing edges

$$\forall x \forall y \; D(x) \wedge M(y,x) \rightarrow (\forall z \; \neg E(y,z))$$

Axiom 4-8: Nodes mapped to documents must have an incoming edge

$$\forall x \forall y \; D(x) \wedge M(y,x) \rightarrow (\exists z \; E(z,y))$$

Axiom 4-9: A node mapped to a document may map only to that document

$$\forall n \forall x \forall y \; D(x) \wedge M(n,x) \wedge M(n,y) \rightarrow x = y$$

These axioms place additional restrictions on nodes that map to documents. Specifically, a node mapped to a document must be a leaf (Axiom 4-7) and must have a parent (Axiom 4-8). Axiom 4-9 states that a node mapped to a document maps only to that document; that node cannot also map to other documents or terms. Axiom 4-9 also states that, when a single node is mapped to two different elements, both of those

elements must be terms. This axiom supports polyterms (i.e., 'mooring pile, dolphin, buoy') – where multiple terms are all mapped to the same node. It is important to note that multiple nodes may map to the same term (multiple occurrences) and multiple nodes may map to the same document (i.e., multiple keywords for the same document).

Axiom 4-10: Only terms may be synonymous

$$\forall x \forall y\ S(x,y) \rightarrow$$
$$((\exists z\ M(x,z) \wedge T(z)) \wedge (\exists w\ M(y,w) \wedge T(w)))$$

Axiom 4-11: Only terms may be related

$$\forall x \forall y\ R(x,y) \rightarrow$$
$$((\exists z\ M(x,z) \wedge T(z)) \wedge (\exists w\ M(y,w) \wedge T(w)))$$

Axiom 4-10 and Axiom 4-11 state that only nodes mapped to terms (not nodes mapped to documents) can be synonymous or related. As explained in Section 2.1.2, synonyms and related terms are *not* assumed to be symmetric or transitive.

4.2.4 Functions

This section formally defines the functions in the Metadata++ model using set theory, recursive first-order logic, and the predicates defined in Section 4.2.2. These functions are logically divided into groups: browsing functions (for exploring the hierarchy), metadata functions (for obtaining metadata about individual documents), and searching functions (for executing document searches).

4.2.4.1 Browsing Functions

Several functions work together to provide the browsing functionality within Metadata++. These functions are defined as follows:

Function 4-1: GetRoots: ∅ → Powerset(N)

$$GetRoots() = \{ m \mid \nexists n\ E(n,m) \}$$

Function 4-2: GetChildNodes: N → Powerset(N)

$$GetChildNodes(n) = \{ m \mid E(n,m) \}$$

Function 4-3: FindTerms: string → Powerset(N)

$$FindTerms(s) =$$
$$\{ n \mid \exists t\ M(n,t) \wedge T(t) \wedge TermMatch(s,t) \}$$

Function 4-4: NodeTerms: N → Powerset(T)

$$\text{NodeTerms}(n) = \{ t \mid M(n,t) \wedge T(t) \}$$

The `GetRoots` function returns all top-level nodes (i.e., nodes without parents). The `GetChildNodes` function returns a set containing the child nodes of the specified node. The `FindTerms` function finds all nodes which map to terms that match a specified string. This function helps the user to find terms that may be of interest by typing a word or phrase and searching for terms that contain that word or phrase instead of manually browsing the hierarchy and looking for matching terms. As described in Section 5.1.1, the `FindTerms` function must execute efficiently because its scope includes the entire set of terms (and their associated nodes) in the hierarchy. The `NodeTerms` function finds all terms that map to a particular node. This function is especially important when working with polyterms; the application can determine which term(s) map to the current node in the hierarchy using `NodeTerms`.

Function 4-5: MultipleOccurrences: N → Powerset(N)

$$\text{MultipleOccurrences}(n) = \{ m \mid (\exists t\ M(n,t) \wedge M(m,t) \wedge T(t)) \wedge m \neq n \}$$

Function 4-6: GetRelatedNodes: N → Powerset(N)

$$\text{GetRelatedNodes}(n) = \{ m \mid m \in \text{MultipleOccurrences}(n) \vee S(n,m) \vee R(n,m) \}$$

The `MultipleOccurrences` function returns a set of nodes that map to the same term as the specified node. The application uses this function when the user wants to see all locations of the same term within the hierarchy. It is important to note that a multiple occurrence of a term might also be a polyterm. For example, consider the term *dolphin* as explained in Section 2.1.2.1.2). Two different nodes map to the term *dolphin*, one under *Habitat Elements* and one under *Mammals*, and the first occurrence of *dolphin* is also a polyterm (*mooring pile, dolphin, buoy*). The `GetRelatedNodes` function returns a set of nodes that are in some way related to the specified term – either a multiple occurrence, a synonym, or a related term. While browsing the hierarchy, the user may choose to see all of the terms related to the current term. Metadata++ uses the

`GetRelatedNodes` function to determine which nodes (and terms) should be displayed to the user.

4.2.4.2 Metadata Functions

The functions defined in this section provide functionality for obtaining document metadata – showing how documents are related to terms in the hierarchy.

Function 4-7: GetExplicitDocuments: N → Powerset(D)

```
GetExplicitDocuments(n) =
    { d | ∃m E(n,m) ∧ M(m,d) ∧ D(d) }
```

Function 4-8: GetImplicitDocuments: N → Powerset(D)

```
GetImplicitDocuments(n) =
    { d | ∃t M(n,t) ∧ D(d) ∧ DocumentMatch(t,d)
          ∧ d ∉ GetExplicitDocuments(n) }
```

During browsing and searching, Metadata++ shows the user a list of documents that are either implicitly related or explicitly related to the current term. When an author or librarian selects a term as a keyword for a document, the system creates a new node (that maps to the document) and a new edge between the node of the current term and the new document node. The `GetExplicitDocuments` function will return documents that are mapped to by child nodes of the specified node (the node mapped to the current term). The set returned by `GetExplicitDocuments` is unordered – so the order in which the application lists the explicit documents has no significance.

The `GetImplicitDocuments` function shows a list of documents that contain the term. The `GetImplicitDocuments` function will only return documents that are *not* explicitly related to the term. So if the document is explicitly related to the current term, then it will not be listed as an implicit document – regardless of whether or not the document contains the term. The `GetImplicitDocuments` function properly handles polyterms; it will return documents that contain any of the terms that map to the current node. The formal definition of `GetImplicitDocuments` does not imply any specific order to the documents returned by the function. However, in practice, the

application returns the implicit documents in order of ascending rank as determined by the full-text document indexing engine.

Function 4-9: GetDocumentKeywords: D → Powerset(N)

$$\texttt{GetDocumentKeywords(d)} = \{ n \mid \exists m\ M(m,d) \wedge E(n,m) \}$$

In contrast to the `GetExplicitDocuments` function (which returns one or more documents explicitly related to the current term), the `GetDocumentKeywords` function returns all nodes explicitly related to the specified document. Each node returned by this function maps to term(s) that represent document keywords, i.e., terms selected by an author or librarian during indexing.

4.2.4.3 Search Functions

In addition to browsing and metadata function, the Metadata++ model includes various functions for executing searches. Searches are specified by the user selecting one or more search terms from the hierarchy – *not* by entering a free-text query. The functions defined in this section accept or return sets of nodes. The implemented system represents nodes as paths, so a set of nodes returned by these functions is represented as a set of paths which are displayed within context of the hierarchy.

Function 4-10: GetDescendants: N → Powerset(N)

$$\texttt{GetDescendants(n)} = \{ m \mid A(n,m)\}$$

When a search result is displayed to the user, he or she may (optionally) expand the search by including all narrower terms. The `GetDescendants` function uses recursion to find all nodes that are descendants of the specified node. If the user chooses to add all descendants to a specified search term, the system uses this function to find all of the nodes (mapped to terms) that are descendents of the specified node. As described in Section 5.1.2, the `GetDescendants` function must work efficiently because it may be searching an arbitrarily large branch of the hierarchy.

Function 4-11: DocNodes: Powerset(N) → Powerset(N)

$$\texttt{DocNodes(X)} = \{ n \mid \exists w\ n \in X \wedge M(n,w) \wedge D(w) \}$$

Function 4-12: DocSet: Powerset(N) → Powerset(D)

$$\texttt{DocSet(X)} = \{ w \mid \exists n\ n \in X \wedge M(n,w) \wedge D(w) \}$$

The `DocNodes` function will return only those nodes (from the input set) that map to documents. When the user is searching for documents within the hierarchy, it is useful to remove those terms that are not related to any documents. This function is particularly useful in dense areas of the hierarchy – where a particular node might have hundreds or thousands of child nodes, but relatively few of those child nodes are explicitly related to documents. In addition to viewing documents within context of the hierarchy, the user may also choose to view a simple list of documents. The `DocSet` function will take a set of nodes and return a set of documents mapped to by nodes in the input set.

Most systems support compound searches by allowing the use of Boolean operators to combine searches. Metadata++ also supports Boolean operators, with some differences compared to traditional set-based Boolean operators [7]. All of the functions defined in this section are binary operations – they take two separate sets of nodes as input and produce a set of nodes.

Function 4-13: Union: Powerset(N) × Powerset(N) → Powerset(N)

$$Union(X,Y) = \{ n \mid n \in X \vee n \in Y \}$$

This function is the simplest operator for doing compound searches – it is simply the union of two trees. Any node that is in either (or both) of the input trees is also in the output tree. If the user wants to search for "term_x or term_y" then they will see the results of both searches combined into one hierarchy.

Function 4-14: Intersection: Powerset(N) × Powerset(N) → Powerset(N)

$$Intersection(X,Y) = \{ n \mid (n \in X \vee n \in Y) \wedge \exists z\ M(n,z) \wedge (T(z) \vee (z \in DocSet(X) \wedge z \in DocSet(Y)))\}$$

When a user searches for "term_x and term_y", Metadata++ needs to do more than just intersect the set of nodes. Instead, the system will use the `Intersect` function to return both sets (as with the `Union` function), but only with the nodes that map to documents that are mapped to by one (or more) node(s) in both input sets. If a document was found in the search result of "term_x" and the same document was also found in the search result of "term_y", then that document will still appear in place as

related to term_x and it will also still appear, in place, as related to term_y. This function allows the user to retain the context of the search while removing those documents that are not related to both search terms.

Function 4-15: Difference: Powerset(N) × Powerset(N) → Powerset(N)

$$\texttt{Difference}(X,Y) =$$

$$\{\ n \mid n \in X \wedge \exists z\ M(n,z) \wedge (z \in \texttt{DocSet}(X) \wedge z \notin \texttt{DocSet}(Y))\ \}$$

The `Difference` function will retain only those documents (along with associated terms) that are in the first tree, but are *not* in the second tree. The structure of the first tree remains the same – except all documents found in the second tree are removed.

4.2.4.4 Display Functions

Function 4-16: GetPath: N → List(N)

$$\texttt{GetPath}(n) = \{\ x_1, x_2, \ldots, x_k \mid x_1 \in \texttt{GetRoots}() \wedge (\forall i, 1 \le i \le k-1,\ E(x_i, x_{i+1}) \wedge n = x_k\ )\}$$

The `GetPath` function returns the full path (as an ordered list of nodes) for any node in the hierarchy.

Most of the functions defined previously utilize sets of nodes either as input, as output, or as both input and output. These sets of nodes contain arbitrary nodes from the set N; for example, they might be selected by the user during indexing or searching, or they might be returned by the `FindTerms` function. These sets of nodes contain only those nodes pertinent to the particular situation. For example, the `FindTerms` function may return a set containing two nodes, one that is three levels down in one vocabulary, and another that is four levels down in a different vocabulary. Before displaying this set of two nodes, Metadata++ first computes the *proper prefix* with respect to the original hierarchy. Because the original hierarchy is a forest (as defined by graph theory), every node has a unique simple path from a root node. The proper prefix includes each specified node, along with all nodes in the corresponding unique simple path leading to each specified node. Computing and displaying the proper prefix of any set of nodes

provides a consistent and meaningful visualization for the user. The proper prefix is faithful to the original hierarchy and consistently retains the structure of each controlled vocabulary. Whether finding terms, browsing term relationships, selecting keywords during indexing, interactively expressing searches, or viewing search results, the user is always "living in the hierarchy".

Function 4-17: GetProperPrefix: Powerset(N) → (Powerset(N),Powerset(E))

$$\texttt{GetProperPrefix}(N_x) = (\ N_F,\ E_F\ ) \text{ where}$$

$$N_F = \{\ m \mid m \in N_x \vee (\exists z\ A(m,z) \wedge z \in N_x)\}$$

$$E_F = \{\ x,y \mid x \in N_F \wedge y \in N_F \wedge E(x,y)\ \}$$

The `GetProperPrefix` function returns the minimal set of nodes and edges that represent the proper prefix of the input nodes with respect to the original hierarchy. The set N_F contains all of the input nodes, as well as all of the ancestors of the input nodes. The set E_F contains all of the edges that touch any two nodes within the set N_F. Like the original sets N and E, the sets N_F and E_F jointly define a forest. Axiom 4-2, Axiom 4-3, and Axiom 4-4 restrict the original hierarchy (defined by the sets N and E) to a forest by enforcing that all nodes have at least one edge and no more than one incoming edge, and preventing any node from being an ancestor of itself. As defined by graph theory, any sub-graph of a forest is also a forest. Thus the output of `GetProperPrefix` is guaranteed to also be a forest.

In order for (N_F, E_F) to be a proper prefix of (N, E), all root nodes in (N_F, E_F) must also be root nodes in (N, E). By definition, the set N_F includes all of the input nodes, as well as all ancestors of each input node. Every node in set N is either a root or has an ancestor that is a root. Therefore, N_F includes all of the root nodes from the set N that are either input nodes themselves, or the ancestor of one or more input nodes. Since the set E_F contains no edges that do not exist in E, a node cannot be a root in (N, E) but a non-root in (N_F, E_F). Given that all roots in (N_F, E_F) are also roots in (N, E), it follows that the rooted paths (paths starting at a root node and following zero or more edges) in (N_F, E_F) are also rooted paths in (N, E), since by definition $N_F \subseteq N$ and $E_F \subseteq E$.

4.3 Related Work

Many knowledge-based information retrieval systems use either thesauri or ontologies for knowledge organization. This section compares the Metadata++ conceptual model to thesaurus-based models and ontology-based models.

4.3.1 Thesaurus Models

Research spanning four decades focuses on the definition and review of thesaurus models in information retrieval [35,101]. Many modern systems adhere to the NISO standard for thesaurus construction [3]. These systems typically define their conceptual model in terms of RDF [119]. Authors at the World Wide Web Consortium reviewed [77] the thesaurus-based conceptual models [77] of several different systems [21,18,41,43,72,107]. This section will compare the Metadata++ to the same criteria used by Miles and Matthews [77] – to provide an overall review of how Metadata++ compares to the other systems described in that review.

Metadata++ uses a 'concept-based' approach – where nodes in the set N correspond to concepts. A node in Metadata++ is defined only by its path, there is no explicit definition. In the 'concept-based' approach, relations exist between concepts, and concepts are labeled by terms. The alternative is a 'term-based' approach – where terms are the only entities and participate in relationships directly. A 'term-based' approach would prevent important features, such as multiple occurrences (where multiple nodes map to the same term).

Some systems [18,107] allow the classification of terms according to top-level facets or categories – such as 'object', 'people', 'properties', and so forth. The Metadata++ model does not have any predefined categories for the classification of terms. However, the metadata expert may create whatever nodes are necessary to properly organize the various controlled vocabularies within the hierarchy. For example, in the natural resource domain, we created approximately twenty eight top-level nodes that correspond to the different subject areas [107]. Additional top-level nodes (similar to facets) can be created as necessary, and each of these nodes contains one or more controlled vocabularies.

The NISO thesaurus standard [3] defines a fixed list of relationships – including BT (broader term) and NT (narrower term). In addition to BT and NT, the standard further specifies more specific types of hierarchical relationships, such as BTP/NTP (broader/narrower term partitive) which indicate part-whole relationships. For example, the term 'central nervous system' would be related to the term 'nervous system' using the BTP relationship – because the central nervous system is one part of the whole nervous system. In Metadata++, the edges in the set E represent all of the standard NT (narrower term) relationships (NT, NTP, NTI, …). Metadata++ does not have any specific semantics when establishing narrower term relationships – other than representing the terms "as-is" in the controlled vocabularies.

Of the systems compared by Miles and Matthews [77], only a few support multi-lingual thesauri. Multi-lingual functionality is outside the scope of our project and is not currently accommodated by the Metadata++ model. However, Metadata++ does support inter-thesaurus mapping – where each of the source controlled vocabularies is considered a thesaurus and relationships may exist between nodes that map to terms in different vocabularies. So each language could have a thesaurus represented as a separate vocabulary in Metadata++. All but one of the systems compared by Miles and Matthews [77] use RDF to formally define the model. Metadata++ uses set theory and first order logic to formally define the conceptual model.

Polyhierarchies, as defined by the thesaurus standard, occur when a single concept is logically related to two (or more) different broader concepts. For example, 'piano' is both a 'stringed instrument' and a 'percussion instrument' – so it would be a child of both terms. In this case, it is still the same concept; but it is related to multiple broader concepts. When two different concepts have the same label (i.e., homonyms), the thesaurus standard recommends using parenthetical qualifiers. For example, the standard might suggest using the terms 'Riparian (aquatic biology)' and 'Riparian (watershed management)'. Metadata++ unifies polyhierarchies and parenthetical qualifiers by using the entire path to distinguish between the different meanings or connotations of the same term (see Section 2.1.2.1.1).

4.3.2 Ontology Models

Some systems use class-based structures or ontologies to store and utilize knowledge. Using an ontology definition language, such as OWL [121], supports the definition of specific constraints on the conceptual model – such as forcing each concept to have exactly one preferred term [77]. This section compares Metadata++ to several ontology-based systems.

Staab et al. [103] present semantic community web portals based on the Ontobroker [23] system. Their approach focuses on a single ontology that represents the shared knowledge of the community. Concepts are explicitly represented in the ontology and documents are related to concepts. The ontology is defined in F-Logic [64]. The query capabilities of the semantic portal include predefined queries, an ontology browser, and explicit F-Logic queries. Instead of mapping labels to pre-defined concepts, Metadata++ represents existing controlled vocabularies "as-is". The nodes and edges in Metadata++ imply no specific semantic meaning – other than to represent the hierarchical structure of the controlled vocabulary. Metadata++ emphasizes path-based terms, where the entire path is used to distinguish connotation instead of a single concept.

Ambite et al. [2] use an ontology-based approach in the Digital Government Research Center (DGRC) Energy Data Collection (EDC) project. Multiple domains are accommodated by mapping each domain to an existing reference ontology – as opposed to mapping directly between domains. Metadata++ accommodates controlled vocabularies from different sub-domains by integrating all vocabularies into the same hierarchy. Multiple occurrences (the same term in different locations) are implicitly related by the system – so an explicit relationship is unnecessary. Domain experts may create relationships between any nodes in the hierarchy – which eliminates the need to create and maintain a reference ontology.

Weinstein [117] uses an ontology focused on bibliographic concepts to generate and search metadata from Machine Readable Cataloging (MARC) records. These records contain bibliographic information – often redundant in similar works. For example, if Parley Pratt wrote many books, the string value 'Parley Pratt' in each record would be redundant. By making the values explicit, redundancy is eliminated. Bibliographic relations become explicit and can be utilized in intelligent searches.

Weinstein's approach uses a predefined ontology designed specifically for bibliographic data. The concepts are related with a predefined set of relationships with specific semantic meaning. Metadata++ is similar to Weinstein's approach in that keywords (terms associated with documents) are represented explicitly, which eliminates redundancy and improves performance when viewing documents while browsing the hierarchy. Although the current Metadata++ system is targeted towards the natural resource domain, the model could be applied to other domains.

Motta et al. [79] take a holistic approach to document enrichment based on an ontology. They focus on carefully defining the ontology to meet the needs of the users. Instead of annotating documents with metadata, they try to populate the ontology with documents. While it is important to represent controlled vocabularies "as-is", Metadata++ does not require that the hierarchy be static. Instead of focusing on designing the ontology completely and correctly the first time, Metadata++ allows terms to be created and related as you go along. When a new term is created and added to the hierarchy, it can be related to existing terms – reducing the need to re-create metadata for existing documents in reference to the new term.

The Simple HTML Ontological Extensions (SHOE) project [48] allows users to annotate web pages with metadata based on one or more ontologies. The project includes an editor that facilitates annotating exist web pages. SHOE uses metadata that is stored within web pages. The metadata is read by a crawling agent and used to answer queries. Because SHOE is an extension to HTML, it is focused primarily on HTML documents. Metadata++ makes no stipulations about what type of documents can be used in the system. The relationships between documents and terms (i.e. keywords) are stored explicitly outside of the document itself.

Chung et al. [21] apply sophisticated statistical algorithms to infer relationships between terms automatically extracted from an existing domain. Their focus is implementing the algorithms to efficiently process very large domains. Metadata++ is not designed to automatically infer relationships between terms. Instead, Metadata++ represents existing controlled vocabularies "as-is" – preserving the exact structure of the vocabulary as defined by the domain. Admittedly, our approach focuses on vocabularies

within a particular domain (e.g. natural resource management) and may not be applicable for a generic web-based search engine.

4.4 Summary

This chapter describes alternative model features that we considered during the early stages of our research and formally defines the current Metadata++ model using set theory and recursive first-order logic. The formalization consists of sets, predicates, axioms, and functions that define the hierarchy as a forest as described in graph theory. The formalization prescribes an exact definition of the functionality of the Metadata++ model and provides a formal specification for any implementation.

5 Building an efficient path-based storage and retrieval system

A key component of the Metadata++ digital library system is an efficient path-based storage and retrieval mechanism that supports the functions formalized in the previous chapter. This chapter describes four different approaches that we implemented and evaluated during the course of our research [113]. Clearly defined functional requirements enabled us to select the implementation method that most efficiently supports the Metadata++ application. For simplicity, the illustrations and explanations in this chapter will use a small portion of the Metadata++ term hierarchy as illustrated in Figure 5-1.

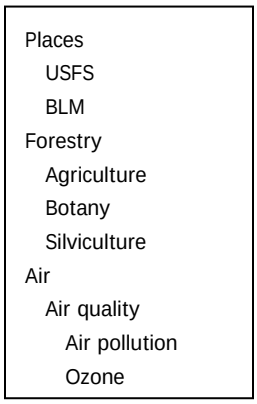


Figure 5-1: Example Hierarchy

5.1 Functional Requirements

The overall objectives for the path-based storage and retrieval system within Metadata++ are scalability and "real time" interaction as perceived by the user. The system is expected to support many thousands (10^5) of terms, and thousands (10^4) of documents (a moderate requirement when compared to the 10^9 documents indexed by modern web search engines). When the user interacts with the system, he or she expects to receive a timely response (within a couple of seconds). Lengthy delays make the system difficult to use and users quickly lose interest. This section describes the three challenges in building a responsive system based on a large hierarchy of path-based data.

5.1.1 FindTerms

The system uses the `FindTerms` function (Function 4-3) to find one or more terms given a specified string value. Instead of manually browsing the entire hierarchy to find the desired term, the user may type in a string – such as "Ozone". The `FindTerms`

function will return all terms that contain the specified string. The `FindTerms` function must efficiently complete two potentially time-consuming tasks: (a) traverse the entire set of nodes, and (b) perform a regular-expression-based string comparison for each node. Because this function must traverse the entire set of nodes (unlike most of the functions defined in Section 4.2.4), its performance is directly related to the number of nodes – and its execution time is a key component in the overall efficiency of the system.

5.1.2 GetDescendants

While searching for documents, the user may choose to expand a search by automatically including all documents related to any descendant of the current search term. For example, a search for *Mammals* might include documents related to beavers because the term *Beavers* is a descendant of the term *Mammals*. The `GetDescendants` function (Function 4-10) finds all of the descendants of a given term. This function may potentially traverse a large number of nodes – depending on the starting node and the breadth and width of the hierarchy. An efficient implementation of the `GetDescendants` function is essential to the overall efficiency of the system.

5.1.3 Concurrent Modifications

In addition to efficiently implementing the functions described above, the system must also support concurrent modifications. These modifications are write operations that include:

- authors or librarians adding new documents to the system
- authors or librarians creating metadata (including explicit keywords) for documents
- librarians managing the hierarchy of vocabularies (including adding and removing terms)

Because of the distributed nature of the application, two (or more) users may attempt modifications concurrently. For example, a botanist may be adding new species to the vegetation taxonomy at the same time that a wildlife biologist may be adding new species of fish. As with a relational database, the system must correctly, and responsively, receive and store all modifications.

5.2 Multiple Implementations

This section explains four different approaches for implementing a path-based hierarchy that supports the functional requirements explained in the previous section. The implementations described here include various combinations of relational database tables, XML, and file system technology. The following section (Section 5.3) outlines the performance for each alternative.

5.2.1 Parent-Child Binary Relation

The simplest approach uses a relational database table that represents a binary relationship between nodes – one row in the table for each parent-child relationship. This approach (illustrated in Figure 5-2) is similar to the edge relation described by Deutsch et al. [28] and Florescu and Kossmann [39]. It is important to note that this approach (as well as the next two approaches) actually uses nodes that map to a separate table of terms. The figure shows the actual text of the term (instead of the node identifier) to improve readability.

This approach is functional, but fails in terms of scalability and performance. Because each parent-child relationship is a separate row in the table, executing the `GetDescendants` function (e.g., to do a search or display the hierarchy to the user) requires an additional query for each level of the hierarchy (or a recursive query if supported by the relational database system). Even with relatively few terms (e.g., 10^2 terms), the performance of the `GetDescendants` function is noticeably slow. The `FindTerms` function also performs slowly. It does not take long to find the matching terms (exact match or wildcard) – the majority of the time is spent finding the ancestors in order to return the precise location in the hierarchy. For example, suppose the user searched for the string ‘ozone’. An index lookup and string

PARENT	CHILD
Places	USFS
Places	BLM
Forestry	Agriculture
Forestry	Botany
Forestry	Silviculture
Air	Air quality
Air quality	Air pollution
Air quality	Ozone
Air	Weather
Weather	Air pressure
Weather	Evaporation

Figure 5-2: **Parent-Child Binary Relation**

ID	PATH
1	Places
2	Forestry
3	Air
4	Places\USFS
5	Places\BLM
6	Forestry\Agriculture
7	Forestry\Botany
8	Forestry\Silviculture
9	Air\Air quality
10	Air\Weather
11	Air\Air quality\Air pollution
12	Air\Air quality\Ozone
13	Air\Weather\Air pressure
14	Air\Weather\Evaporation

Figure 5-3: **Breadth-first Path Relation**

comparison would quickly find the row containing the term *Ozone*; but three successive queries (or a single recursive query) would be required to compute the full path (*Air\Air quality\Ozone*).

Despite poor query performance, adding or deleting leaf nodes is very efficient – simply add rows or delete rows from the relation. For example, if we wanted to add *Acid Rain* as a child of *Air quality*, we would simply add a new row to the table with the corresponding values. Modifying internal (non-leaf) nodes requires more time to correctly process the affected sub-trees. For example, deleting the branch rooted at Air quality would require recursive queries that delete multiple rows. The underlying database management system supports concurrent updates to the hierarchy.

5.2.2 Breadth-first Path Relation

Our second implementation also uses a relational database, but includes a more novel approach for storing the hierarchy (similar to Dietz's numbering scheme [60]). Instead of storing the hierarchy as a set of parent-child relationships, a structure is used

based on the breadth-first ordering of the tree. Each node of the tree is stored – in breadth-first order – with its full path in the hierarchy (as illustrated in Figure 5-3). (As noted previously, the path is actually stored using term identifiers, but the figure shows the term text for improved readability). The hierarchy can be constructed using a single SQL query ("select PATH from HIERARCHY order by ID") The results of the query will return each path in the tree in breadth-first order – so the user interface can easily construct the appropriate tree structure based on the paths returned.

This approach offers a great improvement over the previous approach when executing the `GetDescendants` function. The nodes in the `GetDescendants` can be determined by matching the prefix of each path – and the structure of `GetDescendants` can be created by re-building the tree in breadth-first order. This implementation requires only a single SQL query – instead of several SQL queries or expensive joins. For example, suppose we want to get all descendants of the term *Air*. This task could be done using the following SQL: select PATH from HIERARCHY where PATH like 'Air\%' order by ID. The query will return just the paths that are rooted at the term *Air* – and they will be returned in breadth-first order for that sub-tree. The `FindTerms` function is also efficient – doing a string comparison on the path. Because the relation stores the full path, the `FindTerms` function does not need to issue additional recursive queries to determine the path (as with the Parent-Child approach).

As the number of terms increases (e.g., 10^3 terms), modifications to the hierarchy take unreasonable amounts of time. Whenever a new term or vocabulary is added to the hierarchy – or an existing term is moved to a new location – the entire hierarchy must be saved because the breadth-first traversal order may change. For example, suppose we want to add a new *Fire Management* vocabulary underneath the *Forestry* term (between *Botany* and *Silviculture*). A new row would need to be inserted (as shown in Figure 5-3) between row 7 and row 8. In order to preserve the breadth-first traversal order, all rows after row 7 would need to be modified – by updating the ID with the new breadth-first order of each row. This process requires computing the breadth-first order and full path for each node – and updating all of the corresponding rows in the table (which also triggers index maintenance). With a large number of terms, this process takes several minutes to complete – an obvious failure in performance.

This solution does not easily support concurrent modifications because arbitrarily large portions of the hierarchy must be updated after computing the breadth-first order. Since the bread-first order is computed within the application then updated to the database, modifications cannot be concurrent (or they may overwrite each other during the update process). Deleting existing terms is significantly faster (and reduces concurrency problems) – because removing rows does not invalidate the breadth-first ordering (so paths do not need to be re-ordered). When a node is deleted from the hierarchy – it simply requires deleting the appropriate row(s) from the table.

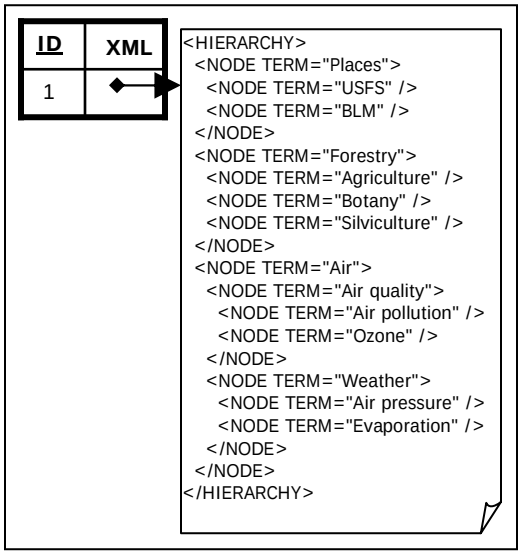


Figure 5-4: **XML BLOB**

5.2.3 XML BLOB

The structure of XML provides an ideal representation for the hierarchy of terms. Our third approach for managing the hierarchy in Metadata++ uses a single XML document to represent the hierarchy. The XML document is stored as text in a relational database binary large object (BLOB) field. This approach is illustrated in Figure 5-4. (As with the previous approaches, the XML in the figure shows the text of the terms, but the actual implementation uses term identifiers). Other data (synonyms, associations, document metadata) are also stored in relational tables. Metadata++ uses the XML to build (and save) the hierarchy in main memory data structures and uses the relational tables (and SQL) to query the related information.

Because the hierarchy is in XML, it is easy to execute both the `GetDescendants` and `FindTerms` functions using XPath (an XML query language)

executing against the XML in main memory. One drawback of this approach is that the entire XML document must be retrieved from the database and parsed into main memory. This process adds significant initial processing overhead. Additionally, this solution does not support modifications from concurrent users. Suppose one user is adding terms to one portion of the hierarchy, while a second user is adding terms to a different portion of the hierarchy. Because the entire XML string is treated as a single value – and the updates to that string are serialized by the relational database system – only the latest modifications are retained in the database.

5.2.4 NTFS and Microsoft® Index Server

Allowing concurrent access to a native XML database would be a suitable solution – assuming the database provided efficient execution of all of the necessary functions and supported modifications from multiple users. At the time of this research, some native XML databases were described in publications [54,115], but few were actually available for use. We found none that included all of the important features found in mature relational databases (such as concurrency). In order to achieve the concurrency needed to support multiple users and obtain the necessary performance requirements, we implemented a simple solution based on the file system (Microsoft NTFS®) and Microsoft® Index Server [77]. Microsoft Index Server is a highly scalable full-text indexing engine included as part of the Microsoft Windows Server® 2003 operating system. The indexing engine runs as a background process and can be configured to watch specific directories and automatically index new (or modified) documents within those directories. The indexing engine can index directory names (in addition to documents) and can be optimized based on typical queries. The server administrator can adjust the priority of the indexing engine process based on the desired response time and resource consumption compared to other processes running on the server.

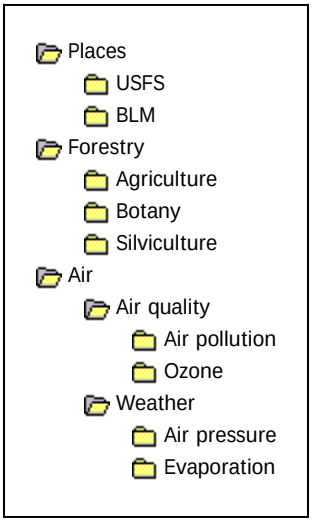


Figure 5-5: **NTFS/IS**

The hierarchy is stored and modified as folders in the file system as illustrated in Figure 5-5. It should be noted that there is no concept of term identifier in this implementation – the folder names are the actual terms. The file system is naturally hierarchical – and already has familiar tools for creating, editing, and deleting folders. It should be noted that scalability limitations with user interface tools – such as Windows Explorer® – are not necessarily limitations of the file system itself. For example, Windows Explorer® became unresponsive when trying to browse a directory that had approximately one thousand immediate sub-directories. However, the same directory caused no problems during browsing with other tools – such as the DOS prompt or through code.

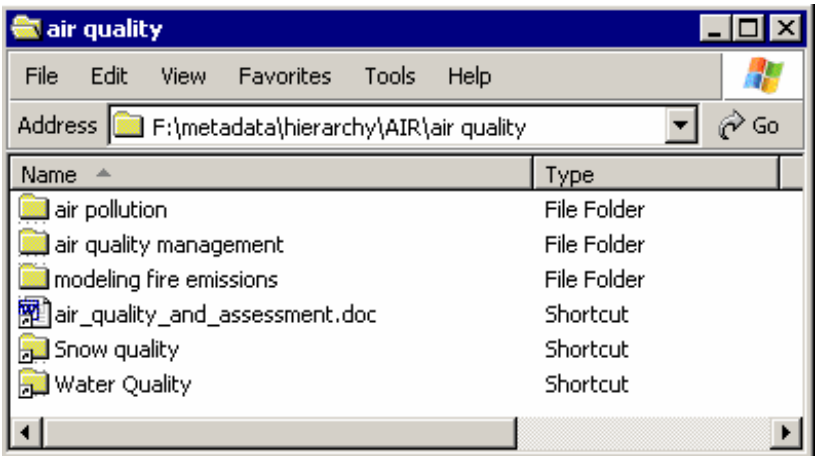


Figure 5-6: Screenshot of Microsoft Windows® Explorer showing shortcuts

In addition to storing terms as directories in the file system, we also use file system features for implementing other features of the model. Relationships among terms are represented as shortcuts (i.e., *.lnk files) that point from within one directory to the related directory. For example, suppose that *Air\Air quality* is related to *AQUATIC\Water\Water Quality*. The ".\Air\Air quality" folder would contain a shortcut (Water Quality.lnk, as shown in Figure 5-6) that points to the ".\AQUATIC\Water\Water Quality" folder. Shortcuts are also used to represent document keywords (i.e., nodes that map to documents as described in Section 4.2). Suppose that *Air\Air quality* is a keyword for the document 'Air Quality and Assessment.doc' (a Microsoft® Word document). The ".\Air\Air quality" folder would contain a shortcut (air_quality_and_assessment.doc.lnk, as shown in Figure 5-6) that points to the corresponding document file.

The NTFS file system also provides the concurrency necessary to allow multiple users to modify the hierarchy simultaneously. Different users can work in different parts of the hierarchy and all of the changes will persist without conflict. For example, a botanist may be adding new species to the vegetation taxonomy at the same time that a wildlife biologist may be adding new species of fish. Since each user is working in a different location in the directory structure, the changes do not conflict with each other and the modifications are successful. However, if both users try to modify the same directory, the file system will prevent the simultaneous changes. The first user will have to complete the modification – after which the second user will be permitted to make the modification. Preventing simultaneous modifications of the same directory is necessary – and expected – to maintain a consistency and prevent corruption in the file system.

The `GetDescendants` function looks at the directory structure to find all descendant terms. This function was originally implemented using the Microsoft Windows® SDK to recursively open each directory and browse its sub-directories. Explicitly browsing each directory – even through code – becomes noticeably slow for large hierarchies (such as the taxonomy of species and other large vocabularies). In order to improve performance, we configured Microsoft® Index Server to catalog all of the file system folders defined in the hierarchy. The index is searchable by folder name – providing support for the `FindTerms` function (both exact match and wildcard). Microsoft Index Server provides an efficient query interface for finding folders within the

hierarchy – as well as searching for shortcuts (to find related terms and document keywords). Microsoft Index Server also provides full-text indexing of all documents in Metadata++, thus enabling the identification of implicit documents, as described above.

	Max	Avg
Depth	14	6.4
Width	9896	*19.5

*does not include leaf nodes

Table 5-1: Hierarchy Statistics

5.3 Performance Results

We implemented each approach described in the previous section on an IBM xSeries 220 eServer® containing a single Intel Pentium III® (1.0 GHz) processor, 896 MB of RAM, and IBM ServeRAID® storage. The server is running Microsoft Windows Server® 2003 Standard Edition and Microsoft® SQL Server 2000 Enterprise Edition. The fourth solution uses Microsoft Index Server (included in the Microsoft Windows Server 2003 operating system). We conducted the experiments using code written and compiled with the Microsoft .NET Framework. The hierarchy consisted of numerous controlled vocabularies provided by various application domain experts – with a combined total of more than 70,000 terms. Table 5-1 summarizes the depth (number of levels) and width (number of children per node) of the hierarchy.

	Q1	Q2	Q3	Q4	Q5	Avg
ParentChild	2.3947	2.4259	0.3556	0.0040	0.2618	1.0884
Path	0.6407	0.9376	0.4493	0.2150	0.2657	0.5017
XmlBlob	0.4220	0.6603	0.7501	0.0040	0.0470	0.3767
NTFS/IS	0.3712	0.6642	0.3673	0.0001	0.0118	0.2829

Table 5-2: **Average GetDescendants Execution Times (in seconds) for 5 queries**

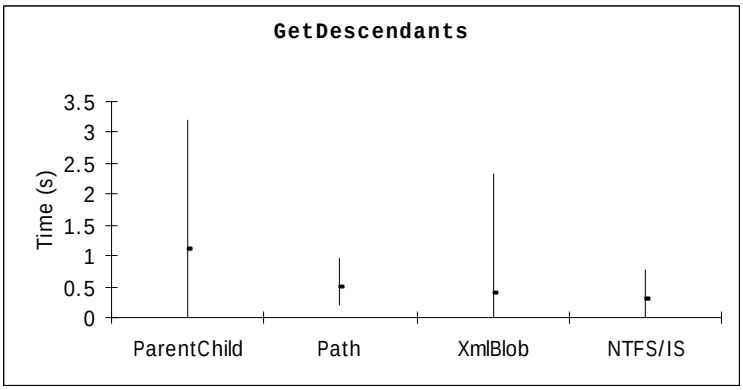


Figure 5-7: **GetDescendants Max-Avg-Min Comparison**

For each approach, we implemented and evaluated both the `GetDescendants` and `FindTerms` functions. We executed each function four times for each of five different queries (with inputs and outputs varying in size and complexity) and averaged the execution times. Table 5-2 shows the results of the `GetDescendants` function for five different queries (averaged for four executions of each of the five queries). Figure 5-7 compares the maximum, minimum, and average `GetDescendants` execution times for each approach. The ParentChild approach is relatively slow because of the numerous queries needed to find all descendants. XmlBlob has a high maximum because of the time it takes to initially parse the XML document into main memory. Both Path and NTFS/IS have relatively low maximum and average times.

Table 5-3 illustrates the four different queries that were used to evaluate the `FindTerms` function, and how many terms were found in each mode. Table 5-4 shows the results of the `FindTerms` function. We executed each query in both modes (exact match and wildcard) and averaged the results. The wildcard mode (any term containing the string) always found more terms – because the exact matches also count as wildcard matches. Figure 5-8 compares the maximum, minimum, and average execution times for each approach. The NTFS/IS approach is an order of magnitude faster (both maximum and average) because Index Server maintains an index of all of the directory names.

Table 5-3: Number of Terms Found for Each of Four FindTerms Queries

	Alpine	Clackamas	Douglas Fir	Soil
Exact Match	3	4	0	5
Wildcard Match	292	8	2	266

As described in the previous section, each approach performed differently with regards to concurrent updates. The Parent-Child approach efficiently handled updates at the leaf level (adding or deleting leaf nodes), but not when modifying entire sub-trees. The Path approach efficiently handles deletes – by simply deleting the appropriate rows in the table – but additions require saving the entire hierarchy. The XML is easily modifiable while in-memory, but does not support concurrent updates (and requires serialization with each modification). The NTFS/IS approach has the best overall performance (and is used in the current implementation). The file system natively supports concurrent updates to different parts of the directory structure, and Index Server watched for changes in the hierarchy and efficiently updated its index for each modification.

Table 5-4: Average FindTerms Execution Times (in seconds)

(average of exact match and wildcard searches for each of 4 queries)

	Alpine	Clackamas	Douglas Fir	Soil	Avg
ParentChild	88.4766	2.9297	0.7110	50.4219	35.6348
Path	16.7266	0.4454	0.2579	13.5782	7.7520
XmlBlob	54.0625	1.9688	0.5079	44.1016	25.1602
NTFS/IS	0.6250	1.7344	0.5625	0.5623	0.8711

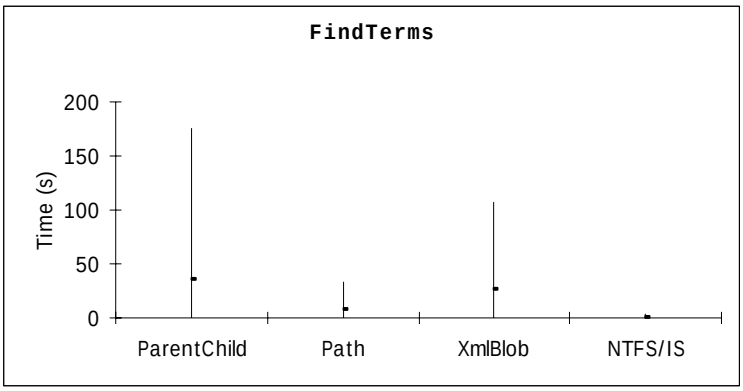


Figure 5-8: **FindTerms Max-Avg-Min Comparison**

5.4 Related Work

The natural choice for representing a hierarchy is XML. Recent languages such as XQuery and XPath make it easy and efficient to query hierarchical data stored in XML. However, the large and dynamic nature of the Metadata++ hierarchy is not adequately supported by current XML tools. A full-featured XML database would be an ideal solution, but the work in this area [54,115] has yet to produce a native XML database with all of the necessary features – including indexing, querying, and concurrency.

An alternative to building a native XML database is to map the XML to relations. The first two implementations described in this chapter (Parent-Child and Breadth-first Path) are attempts at mapping XML-like data into a relational database. The Parent-Child relation is similar to the edge relation described in work on XML representation [28,39]. The edge relation is a generic binary relation used to store node containment within the XML structure. For example, if element A contains element B, then a corresponding row is created in the edge relation. LegoDB [10] takes a novel approach to mapping XML into relations by doing a cost-based evaluation of different relational representations. The cost is computed based on the expected structure of the XML (as defined by a schema or DTD). Because the hierarchy in Metadata++ is a simple tree of arbitrary width and depth, the LegoDB algorithm would also result in a simple edge-based relation. Metadata++ requires functionality (e.g., `GetDescendants` and `FindTerms`) that is not efficiently supported by this type of representation.

Our Path approach is similar to XML indexing techniques described Kaushik et al. [60]. This technique works well for static data, but updates are more difficult. The authors propose leaving gaps in the ordering to handle updates. For example, when initially computing the breadth-first traversal index for each node, we could use a sparse sequence of integers (instead of sequential). Perhaps we would number the first row as 10, the second row as 20, the third row as 30, and so on. Using a sparse sequence initially would still maintain functionality (order would still be preserved). When additions occurred later on, the new nodes could be assigned unused indices that would preserve breadth-first order and eliminate the need to re-calculate the entire breadth-first index. Similarly, real numbers could be used as indices (instead of integers). A new

index value would be assigned as the value halfway between the successor and the predecessor. The entire breadth-first index would only need to be re-calculated when the new value exceeds the available precision. We did not perform any experiments based on these approaches, but they may or may not be suitable for frequent and arbitrary updates.

The Lightweight Directory Access Protocol (LDAP) defines basic directory services for a hierarchically structured set of directories [50,64]. LDAP allows a single conceptual directory hierarchy to be stored in multiple, distributed locations. The Metadata++ hierarchy differs from the directory abstraction supported by LDAP in that Metadata++ terms (in the hierarchy) consist only of the word or phrase that makes up the term (the name of the directory); there is no type structure for terms and, in particular, there is no need to have attributes for terms. Thus Metadata++ uses a much simpler structure for the term hierarchy with a rather narrow focus on the `GetDescendants` and `FindTerms` functions.

5.5 Summary

The Metadata++ Digital Library requires an efficient path-based storage and retrieval mechanism. We implemented and evaluated four different mechanisms: simple parent-child relation, breadth-first path relation, XML stored in a relational table, and the Microsoft Windows® NTFS file system along with Microsoft® Index Server (NTFS/IS). We evaluated each approach against three criteria: support for concurrent modifications, efficient execution of `FindTerms`, and efficient execution of `GetDescendants`. The last approach (NTFS/IS) provided the best overall performance with support for concurrent modifications and is used in the current Metadata++ implementation.

6 Designing the Software Architecture for Metadata++

One of the key components of the Metadata++ digital library system is the software application with which the user interacts with the system. Throughout the course of our research, we developed several iterations of the application – each time trying to improve the architecture with regard to the known objectives – also known as software quality attributes. Section 6.1 explains the architectural objectives for the Metadata++ application and Section 6.2 describes four different implementations we evaluated in trying to meet the objectives.

6.1 Architectural Objectives

Any software architecture is difficult to evaluate without a clear understanding of the desired quality attributes of the application. In collaboration with the USDA Forest Service, we defined a set of quality attributes that helped guide the design and implementation of Metadata++. These attributes are described in the following sections.

6.1.1 Accessibility

Users of the system will reside at various locations – including ranger stations, research laboratories, and regional offices. All of these users need to share information, so the system must be "online"; it must be accessible from different physical locations via the Internet or a wide area network. Many of the locations have some form of broadband Internet connection, while others (such as ranger stations) may be limited to dialup connections. All users will have access to a desktop computer running a Microsoft Windows® operating system. The hardware configuration (processor speed, RAM, disk space, etc.) of the average machine is typical of a computer originally purchased within the past few years.

6.1.2 Availability

Availability is important, but not mission critical. Users will use the system to retrieve information for various purposes (management decisions, FOIA requests, etc.). The majority of these requests will be during "normal business hours". Periodic system failure or offline maintenance is acceptable, assuming that the mean time to repair is reasonable (so as not to significantly disrupt the user's tasks).

6.1.3 Performance

As with availability, system performance is important but not critical. The system does not need to have hard real time performance guarantees. The desired performance is such that the user perceives no significant delays while interacting with the system during various user tasks. As long as the user perceives a "real time" response – i.e., the system responds immediately to a requested action – the system performance is adequate.

6.1.4 Multi-user scalability

Because the system will catalog a controlled collection of domain-specific documents (as opposed to the entire Internet), the number of users is proportionately limited. It is expected that only users within the domain (i.e., natural resource management) – and perhaps some interested public citizens – will make use of the system. The total number of users is on the order of 10^2 – significantly less than the millions of users served by modern web search engines. This objective affects not only the support of a sufficient number of concurrent users, but also the application maintenance and distribution for those users.

6.1.5 Usability

The system must be usable by people with minimal computer skills. It must be easy to use and not require extensive training. Our USDA Forest Service research partners explained the 15-minute rule: if providing metadata for a document takes more than fifteen minutes, it usually will not be provided. The system must simplify tasks for the user – and require minimal time investment for maximum benefit.

6.2 Thin-Client versus Thick-Client

One of the first decisions a software architect must make when building a user-oriented application is the technology for constructing the user interface. In software

development jargon, this issue is intimately tied to the decision to use a thin-client versus a thick-client, or equivalently, "web versus windows". The web (HTTP, HTML, scripting, etc.) is a popular platform for building scalable and accessible applications. Deployment of a web application is simple and efficient; the updated files are copied to the web server, and subsequent client sessions automatically use the updated version of the application. Users access a web application with a familiar, browser-based interface – often referred to as a thin-client. These benefits, however, do not come without a price. Web browsers are limited in the user interface functionality they provide. Some functionality can by achieved through advanced web development techniques (DHTML, client-side scripting, etc.), but this functionality can require significant development effort – and often reduces portability (different browsers expect different script syntax, etc.). Performance implications exist because of the potentially large number of "round trips" to the server in order to accommodate the simplicity of the client.

An alternative to a web application is a thick-client application that employs a rich graphical user interface. Such an application is written and developed using some development tool (such as Microsoft Visual Basic® or Java®) and then compiled into an executable file. The executable file and supporting libraries are deployed to the client machine. This type of application provides rich user interface functionality and rapid development. The deployment, however, is more difficult (each revision of the application requires a subsequent deployment of the new executable to each user's workstation).

The objectives described in Section 6.1 significantly impact this architectural decision. The accessibility requirement suggests a web application – making it easily accessible from any machine with a web browser and an Internet connection. The usability requirement suggests a windows application – providing a rich user interface that makes user tasks easy and intuitive. The following sections describe four different implementations of the Metadata++ application – with varying levels of success at satisfying the architectural objectives. These implementations also correspond to the four different storage mechanisms described in Section 5.2.

6.2.1 HTML/ASP Thin Client

Our first implementation was a traditional thin-client application based on Microsoft® Active Server Pages (ASP). Active Server Pages are scripts that are executed on the server in response to an HTTP request. These server-side scripts, hosted within Microsoft Internet Information Server (IIS) contain logic for obtaining data from Microsoft SQL Server 2000 and constructing a hierarchical display of terms using HTML. Client-side Javascript enables the user to select a search term by clicking on the text of the term within the hierarchy. Search results are displayed in a separate browser window using an HTML table. In this implementation, the server is completely stateless – i.e., no session or user information is retained on the server between requests.

Because of its thin-client architecture, this application sufficiently satisfies the accessibility, availability, and multi-user scalability objectives (limited only by the web server itself). The performance of this application is noticeably poor – due in part to the back-end storage mechanism described in Section 5.2.1 and also due to the numerous round-trips to the server (particularly while exploring the hierarchy). The biggest drawback to this implementation is lack of usability. Most of the user tasks (including exploring the hierarchy and expressing searches) are clunky and not intuitive. For example, browsing the hierarchy required double-clicking the node of interest – instead of the traditional expand and collapse icons in a typical tree-based user interface. Expressing a search involved clicking on the text of a term in the hierarchy, upon which a Javascript function would automatically copy the text of that term into a text box on the page. The user then clicked the 'Search' button to actually perform the search.

6.2.2 VB.NET Thick Client

In an effort to improve usability, we developed a second implementation using the thick-client architecture. We built this application using Microsoft Visual Basic.NET® and Windows Forms (the graphical user interface components within Microsoft .NET Framework). The application makes a direct TCP/IP connection (on port 1433) to Microsoft® SQL Server 2000 (which is running on a server) to retrieve the hierarchy and search for documents. The application stores all session information on the workstation (the server is stateless). The user interface provides familiar methods of interaction – including expanding and collapsing nodes (while exploring the hierarchy) and dragging

and dropping terms (when expressing searches). Expanding and collapsing nodes could have been improved in the previous HTML/ASP application, but dragging and dropping terms is very difficult (and error prone) within a thin-client application.

As is typical with a thick-client application, the increased usability comes along with decreased accessibility and multi-user scalability. A user cannot simply open a web browser and browse to a URL to run the application. Instead, he or she must first obtain and install the application on his or her own machine. This increases the cost and complexity of maintaining the application for multiple users because each user needs to be upgraded with new software updates.

Another drawback with this architecture, also related to accessibility, is firewall restriction on the TCP/IP database connection. The application will not run if a firewall is blocking communication on port 1433. A firewall is typically not a problem within a local area network (LAN) – but is quite often a problem within a wide area network (WAN) or over the Internet. One of our beta testers worked in a ranger district office that used a dialup connection for Internet access. Because of this connection, they were outside the agency firewall and could not communicate on port 1433. The limited bandwidth made VPN access unreasonable. A firewall is definitely a problem when trying to provide the application to public citizens (when the database resides within the agency firewall).

6.2.3 ASP.NET Portal Thin Client

As described in the previous section, the lack of accessibility caused by the firewall restriction is a significant issue. So our third implementation used a thin-client approach based on Microsoft® ASP.NET. We started with the ASP.NET Portal Starter Kit that is available for free public download [4] and added functionality using web controls produced by Infragistics [53]. This application (illustrated in Figure 6-1) uses the NTFS/IS back-end storage mechanism (as described in Section 5.2.4). In this implementation, the server is not completely stateless – minimal session information (including user authentication) is retained on the server between requests.

As with the first implementation (Section 6.2.1), this application sufficiently satisfies the accessibility, availability, and multi-user scalability objectives (limited only by the web server itself). The performance is also much better – mainly due to the

improved back-end storage mechanism. However, even with improved performance on the back-end, the thin-client architecture limits the performance in a couple of ways. The most significant issue is related to the client-server interaction of a web application. Each time the user interacts with the application, such as expanding a node in the hierarchy, the web browser contacts the server (over HTTP) and requests a new page. The server then constructs the new HTML page and sends it back to the web browser where it is parsed and displayed to the user. These page-based interactions, because of the delay associated with the round-trip to the server, are quite inefficient within the Metadata++ application. For example, suppose the user is browsing several levels deep within the hierarchy. The new page returned when the user expands a node must include the child nodes of the expanded node – but it must also include all of the other nodes that were already expanded. So even though expanding a single node may only add a few nodes to the visible hierarchical display, the entire display must be created, transmitted, parsed, and displayed with each interaction. The second performance issue relates to the content of the page itself, because HTML combines content and formatting. The formatting elements significantly increase the number of bytes that must be sent from the server to the client with each interaction.

The usability of this application is better than that of the first thin-client application – but it is still limited by the nature of the web browser. Exploring the hierarchy is intuitive – by expanding and collapsing nodes using the “plus” and “minus” symbols. This application also supports the ‘Find Term’ functionality using a popup window. The user may type in a search string, then click ‘Find’ and the popup window will display all of the terms that match the string. The user may then click on a term and have it automatically added to the search. Despite the improved usability, the application does not support other desired interactions – such as drag-and-drop for selecting terms and right-clicking for additional contextual information about a specific term. Each term has an extra “information” node (as shown by the blue ‘i’ icon in Figure 6-1). The user must expand this node to view explicitly referenced documents, implicitly referenced documents, and all related terms. This extra node is cumbersome and non-intuitive, especially for beginners.

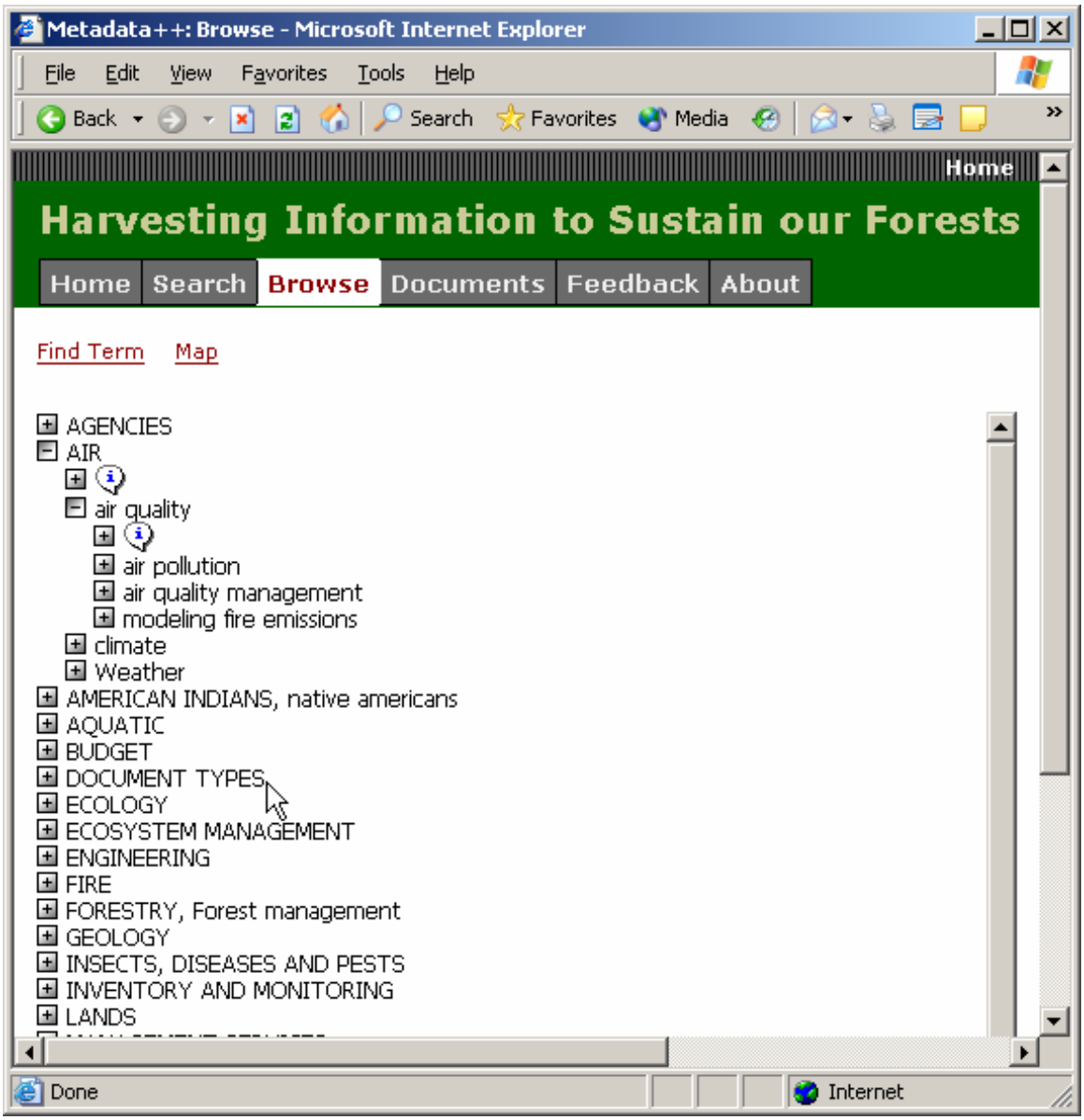


Figure 6-1: **Screenshot of ASP.NET Thin Client**

6.2.4 VB.NET Smart Client

Our final (and current) implementation of the Metadata++ application combines both thin-client and thick-client advantages using the smart client [100] technology made possible by the Microsoft® .NET Framework. We built this application (illustrated in Figure 6-2) using Microsoft Visual Basic.NET® and Windows Forms (including user interface components produce by Infragistics [53]). But we also included all of the deployment and accessibility advantages of a web application using techniques described by Glenwright [45] and Sells [97]. The application uses .NET Web Services [116] (as

described in Table 6-1) to communicate with the NTFS/IS back-end storage system (described in Section 5.2.4) on the server.

Table 6-1: Description of Metadata++ Web Service Methods

Method	Description
FindTerms	Returns all path-based terms that match the specified search string
GetChildTerms	Gets the child terms of the specified path-based term
GetDescendants	Gets all descendants (i.e., narrower terms) of the specified path-based term
GetDocumentKeywords	Returns the explicit keywords (structured as XML) for the specified document
GetDocumentProperties	Returns the summary information (URL, Title, Author, …) for the specified document
GetDocuments	Returns a list of all documents in the library
GetExplicitDocuments	Returns the explicitly referenced documents for the specified path-based term
GetExplicitDocumentsAdvanced	Returns the explicitly referenced documents for the specified path-based term that satisfy criteria based on summary information (Title, Author, …)
GetImplicitDocuments	Returns the implicitly referenced documents for the specified term (regardless of path)
GetImplicitDocumentsAdvanced	Returns the implicitly referenced documents for the specified term (regardless of path) that satisfy criteria based on summary information (Title, Author, …)
GetRelatedTerms	Returns path-based terms (structured as XML) that are related to the specified path-based term (including multiple occurrences, synonyms, and related terms)
RegisterDocument	Uploads and registers a document from the user's local machine
RegisterUrl	Registers the document found at the specified URL (and downloads a copy of the document to the server)
SetDocumentProperties	Updates the summary information (URL, Title, Author, …) for the specified document
UpdateDocumentKeywords	Updates the explicit keywords (specified in XML) for the specified document

The thick-client features of this application (including drag-and-drop, right-click, etc.) make it easy to use and intuitive. Unlike a traditional thick-client application, this application also satisfies the accessibility, availability, and multi-user scalability objectives. The application is easily deployed (by visiting a URL) and automatically updates itself. The application consists of a small wrapper executable and several library files (*.dll). The main functionality resides in the library files; the executable only contains logic for automatically updating the application. Whenever the user executes

the application, the executable will contact the server and ask if any new library files exist. If any files are newer on the server, they are automatically downloaded to the client machine. The number (and size) of updated files, as well as the connection bandwidth, determine how long it takes to complete the auto-update process. On a typical LAN connection, an auto-update would usually take less than one minute (often much less than a minute). After the auto-update is complete, the executable will load the library files and the user may begin using the application. In this way, the application is always current without requiring maintenance upgrades on each machine. The auto-update process and the web services communicate over the standard HTTP port (port 80) – which eliminates the problems caused by firewalls blocking the database connection.

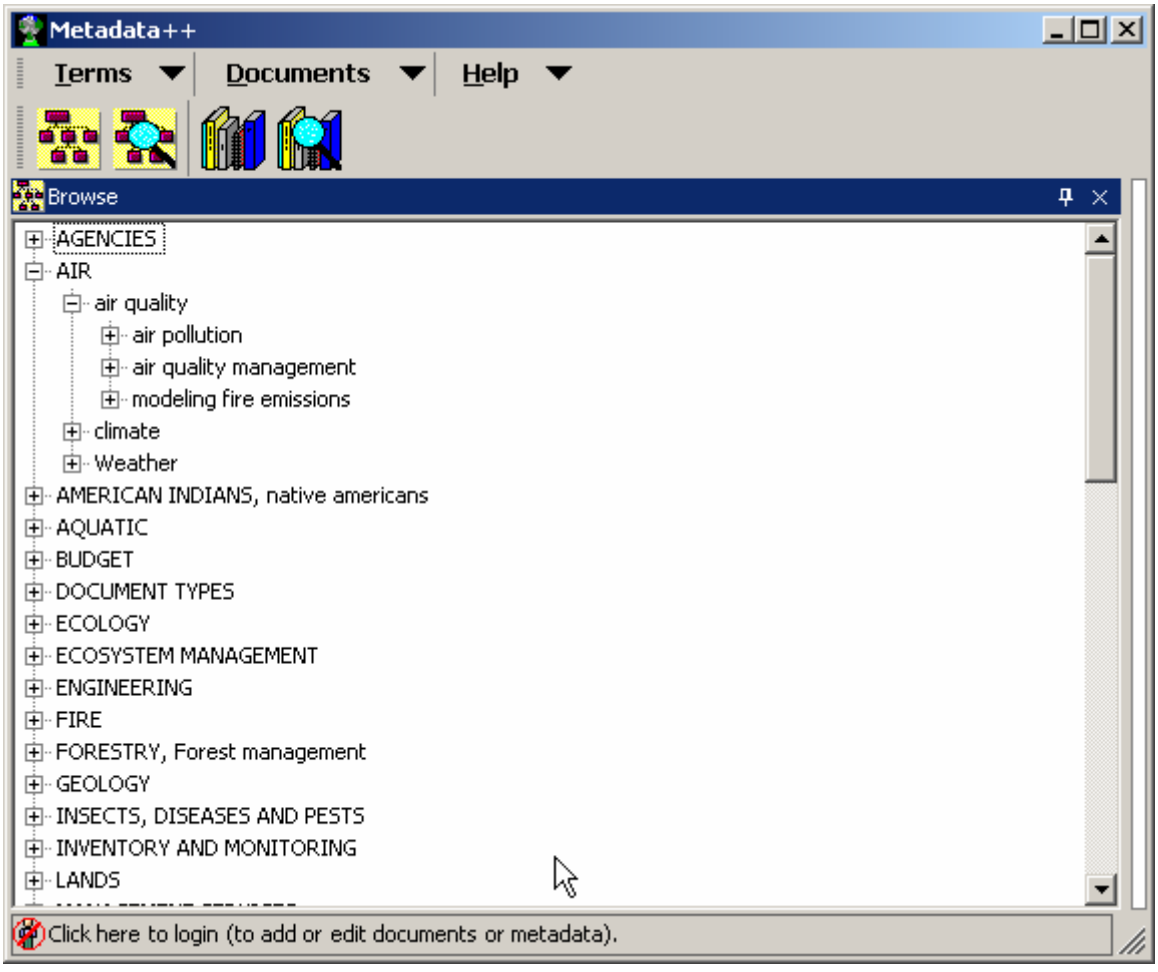


Figure 6-2: Screenshot of VB.NET Smart Client

This application also resolves the performance issues described in Section 6.2.3. Because the application is running as a standalone application on the client machine, it maintains its own process state (server is stateless) – which eliminates the need to transmit the entire hierarchy with each transaction. For example, when a user expands a node, the server only needs to send the children of that node – the rest of the hierarchy is already on the client. The client receives the new nodes (from the web service) and adds them to the hierarchical display. Additionally, the web service only returns content – it does not return formatting information. The formatting is controlled entirely by the client application – which significantly reduces the number of bytes transmitted from the server to the client.

6.3 Summary

Designing and implementing the software architecture for the Metadata++ application involved articulation of architectural objectives and multiple implementations to try and satisfy those objectives. Table 6-2 summarizes each implementation with regard to the objectives. Using a smart-client architecture allows Metadata++ to adequately satisfy each of the architectural objectives. However, a smart-client architecture may not be preferred in other applications with different objectives. For example, a graphically intensive application intended for a small number of users might use a thick-client architecture; whereas a simple application intended for a large number of users might use a thin-client architecture.

Table 6-2: Summary of Architectures with Regard to Objectives

	Accessibility	Availability	Performance	Multi-user Scalability	Usability
HTML/ASP Thin-Client	Good	Good	Poor	Good	Poor
VB.NET Thick-Client	Poor	Good	Good	Poor	Good
ASP.NET Thin-Client	Good	Good	Moderate	Good	Moderate
VB.NET Smart-Client	Good	Good	Good	Good	Good

7 Integrating with GIS

Effective natural resource management requires thorough knowledge of the physical location of the resources being managed. Most of the documents within Metadata++ relate to one or more geographic areas, e.g., indicating the place where a study was done or where a proposed project will take place. Locations are typically described using standard place names, such as administrative organizations (including national forests and ranger districts) or watersheds (such as the Hydrologic Unit Codes defined by the U.S. Geologic Survey [96,110]). Most users in the natural resource management community are quite familiar with geographic information systems (GIS) that display geographic information as a map. Geographic information systems support typical map functionality – including zoom, pan, and point or polygon selection. These systems also support other spatial datasets – such as rainfall, temperature, soil types, and so forth – and provide the ability to query these datasets to find areas of interest.

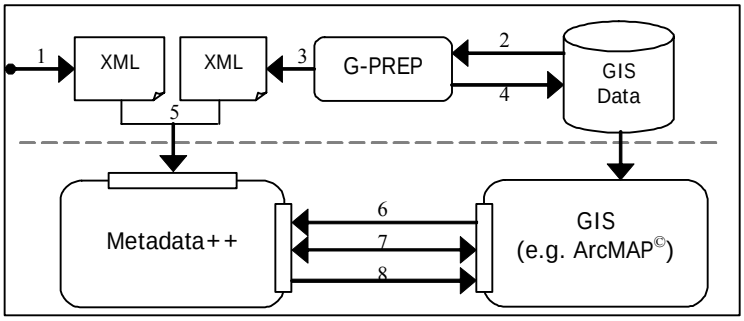
Metadata++ represents each set of place names as a controlled vocabulary and designates spatial containment using path-based terms within the hierarchy. For example, the Cle Elum Ranger District is a child term of Wenatchee National Forest because it is geographically within the forest. Metadata++ currently contains approximately a dozen such location vocabularies that are in common use in natural resource management.

While searching or indexing, the user frequently selects terms from both spatial and non-spatial vocabularies – which presents both challenges and opportunities. We want the benefit of the rich metadata structure and search capability of Metadata++ plus the benefits of spatial reasoning of a standard geographic information system. Our approach [98,112] uses Metadata++ and a standard GIS system running independently, with communication between both systems. Existing GIS datasets (produced by

specialists within the natural resource community) contain place names and are used to generate controlled vocabularies in Metadata++. The user may use the standard GIS system to browse and search various GIS datasets and, at any time, select a region of interest (in the GIS system) and send the corresponding location(s) to Metadata++ for inclusion in a document search. Additionally, when Metadata++ has a set of documents in a search result, the documents that are associated with locations can be sent to the GIS system for display on a map. Metadata++ may also ask the GIS system to compute broader and narrower terms (i.e., containing and contained places) and synonyms (i.e., significantly overlapping places) for any location term.

7.1 Integrated Architecture

This section explains our architecture for integrating Metadata++ and GIS to build a digital geo-library. The architecture enhances retrieval of geographic information (i.e., documents associated with one or more locations) by exploiting functionality of both systems, and communicating information between the two systems, as shown in Figure 7-1. Steps 1-5 (above the dashed line in Figure 7-1) occur during the initial setup phase. Steps 6-8 (below the dashed line in Figure 7-1) occur while the application is in use – and may happen in any order or at any time based on user interaction.



1. (Non-spatial) CVs are prepared in XML
2. GIS datasets are processed by G-PREP
3. Location CVs are prepared in XML
4. Place ID attribute added to GIS datasets
5. XML CV files are loaded into Metadata++
6. User-selected locations are passed to Metadata++
7. Metadata++ asks GIS to compute synonyms
8. Metadata++ passes documents (with locations) to GIS for display

Figure 7-1: Metadata++ Geo-library Architecture

7.1.1 Vocabulary Extraction

The first step in integrating Metadata++ with GIS is extracting controlled vocabularies of place names from GIS datasets. Geographical places are often naturally hierarchical based on spatial containment. Larger places, such as states or national forests, contain smaller places, such as counties or ranger districts, respectively. In GIS datasets, this hierarchy is represented implicitly by the fact that some spatial footprints are contained within others. For example, the polygon representing the State of Oregon geometrically contains the polygons that represent the counties within Oregon. A process called G-PREP implements vocabulary extraction by taking GIS datasets and generating hierarchical controlled vocabularies of place names expressed in XML (shown in Figure 7-1, Steps 2 & 3, and illustrated in Figure 7-2). The controlled vocabularies are then loaded (along with non-spatial vocabularies from other sources) into Metadata++.

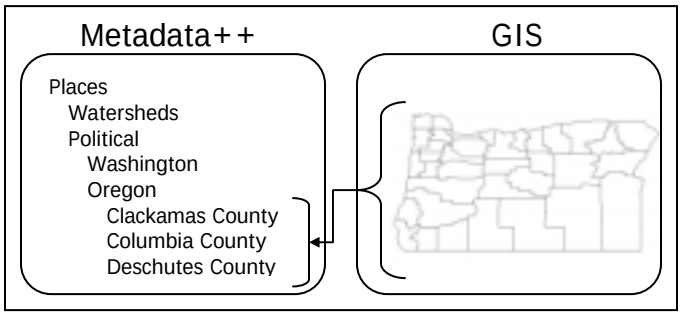


Figure 7-2: Vocabulary Extraction

While implementing G-PREP, we encountered a number of problems. GIS datasets are usually separated into layers (also called themes) based on similar feature type. For example, USDA Forest Service ranger districts are contained in one layer and national forests are contained in a separate layer. Because of this separation of layers, G-PREP cannot generate the complete hierarchy within the context of a single layer. Instead, the G-PREP needs to know which layers correspond to which levels in the hierarchy. With the datasets available to us, we could not easily automate this process. We had to manually specify the proper hierarchical order for processing layers – which limits the scalability of G-PREP.

The nature of geographic footprints presents another, more significant, obstacle to vocabulary extraction. A person might imagine a layer consisting of precise, disjoint polygons – such as political county boundaries. However, many geographic places

cannot be represented by a simple polygon. For example, a national forest may consist of many, non-contiguous regions of varying shapes and sizes – resulting in a complex geographic footprint. All of these polygons must be mapped to one conceptual place name (i.e., the name of the national forest). G-PREP implements this mapping by generating a conceptual place identifier – a globally unique identifier (GUID) – and adding the identifier as a new attribute for each polygon in the dataset. This place identifier attribute is added to the original GIS dataset (as shown in Figure 7-1, Step 4) and included in the XML hierarchy that is loaded into Metadata++. When the XML is loaded into Metadata++, the place identifier is stored with each path-based term. In the NTFS/IS implementation described in Section 5.2.4, the place identifier is stored as a filename within the directory that represents the corresponding term. Index Server is then used to lookup the path-based term for any place identifier – thus serving as an implicit gazetteer that connects spatial footprints with path-based textual place names. Because the place identifier is a separate attribute in the GIS dataset, updates to the dataset (such as refined footprints) have little or no effect on the identifiers.

The place identifier generated by G-PREP is also used to disambiguate place names. For example, the State of Oregon contains twenty six places that are all officially named "Salmon Creek" – three of which are not even creeks! A simple keyword search for "Salmon Creek" would likely yield many irrelevant documents. Using a place identifier in both the GIS and Metadata++ allows the user to precisely select a location of interest – whether from a map or from a controlled vocabulary.

Another significant problem arises because various footprints are generated at different times, using different instruments, for different purposes – and quite often have different precision and accuracy. Because of complex and imprecise footprints, calculating spatial containment is non-trivial. For example, the footprint of a ranger district may slightly extend beyond the footprint of the national forest – even though the ranger district is under the administrative jurisdiction of (and contained within) the national forest. G-PREP uses heuristics to handle these calculations but occasionally requires user interaction.

Representing places as terms in Metadata++ supports the use of unofficial place names – places that are commonly referred to but do not have an official geographic

footprint. Searching for unofficial places names is defined as a necessary feature for digital geo-libraries [82]. For example, most people in the State of Washington know about Snoqualmie Pass – many people could take you there without any difficulty. However, Snoqualmie Pass does not have an official geographic footprint – and may not even appear on some maps. By representing Snoqualmie Pass as a term in Metadata++, the user may still use it for document retrieval – even though it may or may not appear in a GIS dataset. Furthermore, if an unofficial place does have a footprint, we can easily assign a place identifier – as with official places – and include the place name in a controlled vocabulary within Metadata++.

7.1.2 Place Selection

During indexing or searching, the user may select places within the GIS and communicate those selections to Metadata++ (as shown in Figure 7-1, Step 6 and illustrated in Figure 7-3). Place selection may be as simple as pointing to a region on the map and clicking the mouse to select the polygon. However, one advantage of using the GIS is the ability to do more advanced spatial analysis. For example, the user may issue a query to find all counties with geographic area less than two million acres. The GIS will answer this query by automatically selecting those regions within the active dataset that satisfy the specified query, and the selected locations can then be transmitted to Metadata++. In addition to maps, the user may choose to use other types of GIS data to assist in finding relevant regions – such as a dataset of annual precipitation to find those counties that receive more than one hundred inches of rain per year. The GIS will answer this query by intersecting the qualifying regions of the annual rainfall dataset with the map of counties. This query could be combined with the previous example to find all counties with area less than two million acres that receive over one hundred inches of rain annually.

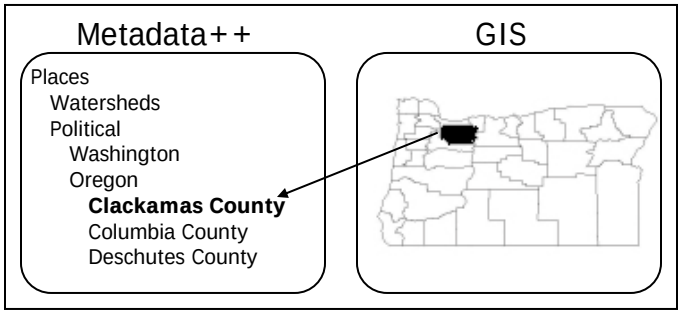


Figure 7-3: Place Selection

After using GIS functionality to select the desired place, the place identifier is sent to Metadata++, which finds the corresponding path-based place name. Selecting a place in GIS is equivalent to selecting the corresponding place name in the Metadata++ hierarchy. The selected place may be used for any Metadata++ function – such as document search or metadata attachment. Users have the flexibility of using either Metadata++ or the GIS – or a combination of both systems – for selecting places. Because all terms are presented to the users in Metadata++, they can easily mix geographic terms (e.g. place names) with any other (non-geographic) terms.

Our work includes three different implementations of G-MAP – the tool that integrates GIS with Metadata++. The first implementation of G-MAP uses an embeddable map component from ESRI® called MapObjects [33]. This implementation is a standalone application that displays GIS datasets (shapefiles [34]) and provides simple map manipulation functionality (zoom, pan, point selection). This implementation limited the user to simple map manipulation – it did not support advanced GIS functionality such as spatial attribute queries (e.g., find all regions with more than 20 inches of annual rainfall).

In order to support advanced GIS functionality, we built a second implementation using ArcMAP® (a popular GIS application produced by ESRI). This implementation adds a new button to the existing toolbar within ArcMAP, which is a familiar tool that many natural resource managers use on a daily basis. The managers may use all of the functionality in ArcMAP to analyze any geographic datasets (including shapefiles). At any point in that process, the user may select one or more locations (within a dataset that has been pre-processed by G-PREP), using any of the selection mechanisms in ArcMAP, and click the button to send the selections to Metadata++.

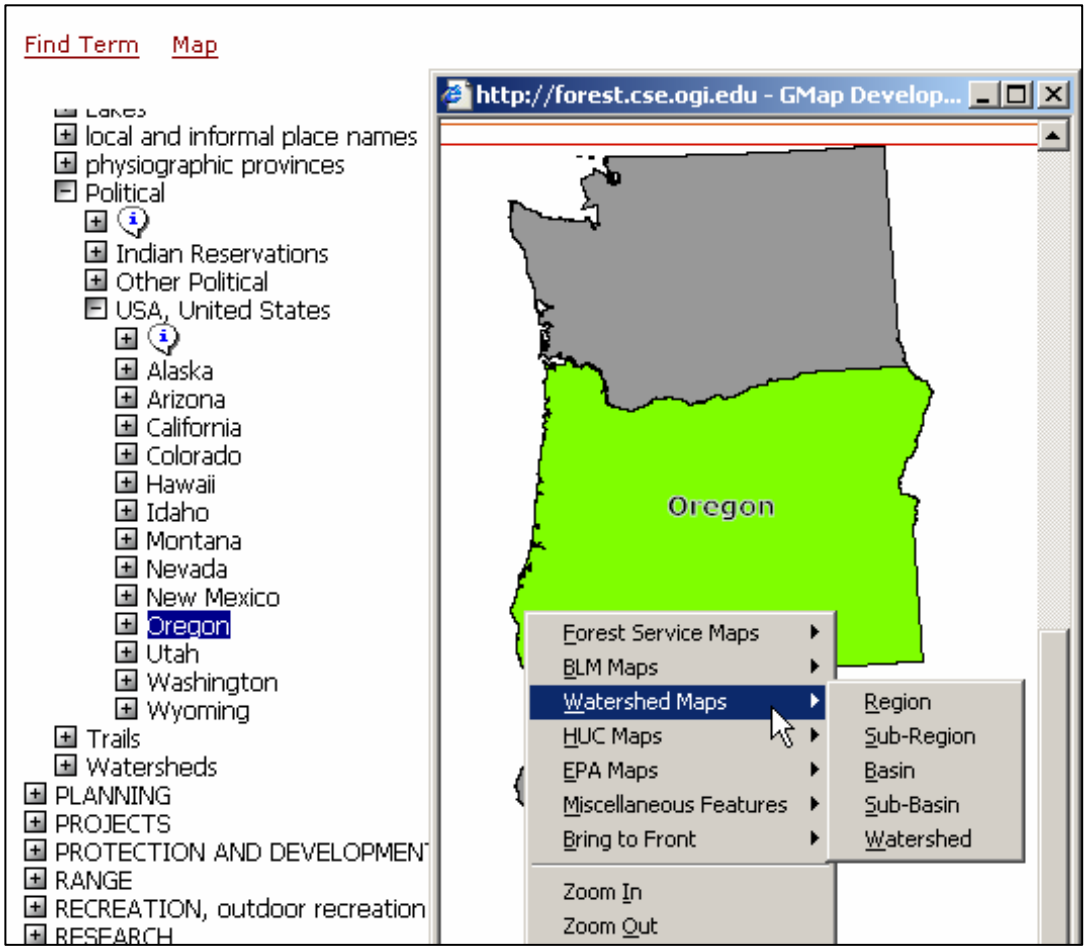


Figure 7-4: **Screenshot of SVG implementation of G-MAP**

Our extension to ArcMAP worked well with the thick-client implementation of Metadata++, but it did not integrate with the thin-client implementation of Metadata++ (as described in Section 6.2.3). Our third implementation of G-MAP (illustrated in Figure 7-4) is a thin-client application that uses scalable vector graphics (SVG) [120] and Javascript to integrate with the thin-client Metadata++ application. As shown in Figure 7-4, the user selected the State of Oregon on the map and the corresponding term in the hierarchy is automatically selected. The illustration also shows the various types of maps that are available to the user while browsing the data spatially.

In this implementation, the G-PREP process produces not only a hierarchy of place names, but also a corresponding SVG representation of the GIS dataset (including place identifiers). The SVG is rendered in the browser window – where the user can turn

different maps on and off using the right-click context menu. When the user selects a place on the map, a Javascript function sends the place identifier to the Metadata++ window – which will contact the server to determine the path-based term associated with that place identifier and select the corresponding term in the hierarchy.

7.1.3 Synonym Discovery

Metadata++ users may choose to use synonyms for query expansion during document retrieval. For non-geographic controlled vocabularies – such as wildlife species or climate terms – synonymous terms are represented explicitly in Metadata++. For geographic controlled vocabularies of place names, Metadata++ uses the GIS to find synonyms (as shown in Figure 7-1, Step 7). We could explicitly represent synonymous places in Metadata++, but we chose not to do so for two reasons. First, a large number of synonyms exist among places. Every place within every location vocabulary may be considered a synonym with one or more places in many other location vocabularies. For example, the land within Clackamas County is also within the Lower Willamette River basin. All of the land in the State of Oregon belongs to some county and also belongs to some watershed and may also be part of a ranger district – resulting in a large number of possible synonyms.

The second, more significant, reason is the ambiguous semantics of spatial synonymy. If the spatial footprints of two different places exactly coincide, then those places would likely be defined as synonyms. However, that rarely – if ever – occurs in real geography. Clackamas County and the Lower Willamette River Basin are in the same geographic place, but their spatial footprints do not coincide. Some points within Clackamas County are not within the Lower Willamette River basin and vice versa. This type of situation makes it difficult to explicitly represent spatial synonyms as related terms in Metadata++.

Spatial synonyms are determined in the GIS by computing a percentage of overlap between the selected polygon and other polygons in different spatial datasets (i.e., vocabularies) based on a user-specified threshold. Because the GIS computes synonyms (instead of representing them explicitly in Metadata++), the user may dynamically adjust the threshold to achieve the desired results. In addition to percentage of overlap, the user may also wish to include other GIS computations (such as adjacency

or proximity) while discovering synonyms. These computations are performed on a default set of GIS datasets – or on one or more specific datasets chosen by the user.

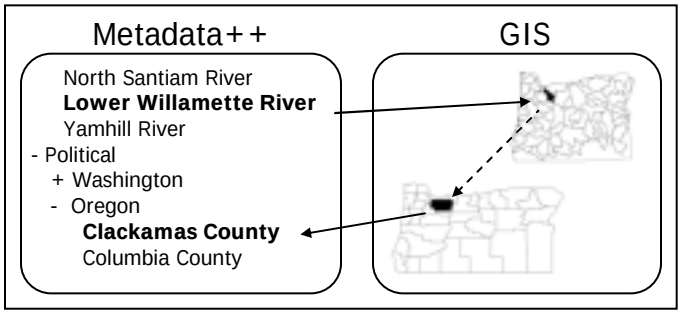


Figure 7-5: Synonym Discovery

Figure 7-5 shows an example of synonym discovery within the GIS. The user selects Lower Willamette River from the Watershed controlled vocabulary. Metadata++ then sends that selection to the GIS. Using the established threshold, the GIS determines which place(s) in other vocabularies (such as political regions) overlap with the selected region. In the example, the GIS determines that the Lower Willamette River basin is a spatial synonym of Clackamas County (because of overlap), and returns that place to Metadata++. In addition to finding synonyms for places selected in Metadata++, synonym discovery may be combined with place selection. If a user selects an area on a map, the GIS can compute the spatial synonyms and send all of the corresponding places (the selected place and its synonyms) to Metadata++ for processing.

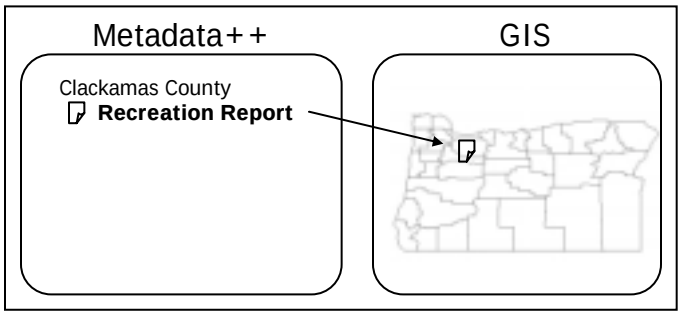


Figure 7-6: Document Display

7.1.4 Document Display

Because many natural resource documents are attached to one or more place names, it is useful to display the documents on a map (as shown in Figure 7-1, Step 6 and illustrated in Figure 7-6). When such a document is part of a query result, Metadata++ can send the appropriate place name(s) to the GIS. The GIS will then display the

documents on the appropriate map. For example, consider a document about hazardous tree removal in Clackamas County. Perhaps the user searched for documents about tree removal in the State of Oregon. Metadata++ would show the document in the context of the hierarchy and would notify the GIS that the document was attached to Clackamas County. The GIS would then locate the polygon corresponding to Clackamas County and display an icon representing the document in or near the polygon. Clicking on the icon will display the corresponding document in Metadata++.

7.2 Related Work

Geographic information retrieval (GIR) focuses on document retrieval based on geographic references within documents. Each document can be uniquely classified as georeferenced, georeferenceable, or non-georeferenceable. Georeferenced documents (such as a digital map) contain explicit reference to geography in the form of a spatial footprint (coordinate, polygon, etc.). Georeferenceable documents (such as an Environmental Impact Statement) do not contain an explicit spatial footprint, but do contain implicit reference to geography in the form of one or more textual place names. Non-georeferenceable documents (such as a scientific report describing the feeding process of spotted owls) are not related – implicitly nor explicitly – to any geographic location. Each type of document must be considered in a GIR system.

Some GIR systems use only spatial queries for document retrieval. Spatial queries execute over geographic footprints – so georeferenced documents are easily retrieved. However, before a georeferenceable document can be considered in a spatial search, the system must somehow associate a footprint with the document [118] – at which point it becomes georeferenced. Automatically associating footprints with georeferenceable documents is non-trivial. Georeferenceable documents contain place names – but place names are often ambiguous. For example, a document about the first president of the United States may be mistakenly associated with the town of George in the State of Washington (which is a real town). Similarly, the city of Portland exists in Oregon and Maine. A related problem is the use of alternate or informal names for places. For example, an older or more informal document may refer to Boston, Massachusetts as “Beantown” or Portland, Oregon as “Stumptown”. Metadata++ searches for documents within the context of the term hierarchy – not in context of the

spatial footprints of documents. Our architecture uses the GIS to execute spatial queries to extract vocabularies of place names and discover spatial synonyms, but document retrieval occurs in Metadata++ (which does not require that documents be associated with spatial footprints).

The Federal Geographic Data Committee (FGDC) [36] is an inter-agency organization that defined a metadata standard for geographic data. The FGDC standard focuses on geo-referenced information, particularly scientific data, with an emphasis on the spatial attributes of the data. FGDC extensions add content-related attributes, but only some enforce specific controlled vocabularies. Metadata++ puts equal emphasis on spatial and non-spatial keywords, because all keywords are path-based terms in the hierarchy. Neither FGDC nor Metadata++ would prevent a document or dataset from having metadata in both systems.

The ADEPT [55] Digital Library Architecture, part of the Alexandria project at the University of California, Santa Barbara, provides support for geographic information retrieval using search buckets. A search bucket is an abstract metadata category with defined search semantics. Collections may provide metadata for items (e.g., documents) using various buckets. For example, the "Geographic location" bucket contains coordinates describing a document's spatial footprint and supports three spatial search operators: contains, overlaps, and is-contained-in. The "Assigned term" bucket contains subject-related terms from controlled vocabularies and supports three text-based search operators: contains-all-words, contains-any-words, and contains-phrase. The ADEPT architecture searches multiple heterogeneous collections by specifying queries using the various bucket search operators. By treating place names as terms in controlled vocabularies, Metadata++ uses a single, unified search operation that exploits hierarchical relationships and synonyms. Synonyms discovered by the GIS are handled uniformly with synonyms explicitly represented in Metadata++.

Recent GIR research [40,58,59] relies on ontologies to facilitate information retrieval. SPIRIT [59] attempts to address proximity relationships as well as alternate and informal place names by developing a geographical ontology that "models both the vocabulary and the spatial structure of places". OASIS [40] represents places as explicit objects in an ontology with specific attributes (latitude, longitude, standard name) and

relationships (meets, overlap, partOf) to other place objects. These relationships are used to explicitly represent and query spatial relationships between places. Instead of building an explicit ontology, Metadata++ focuses on faithfully representing the controlled vocabularies (both spatial and non-spatial) that are commonly used in the application domain. The hierarchy among place names in Metadata++ is similar to the partOf relationship, but Metadata++ relies on the GIS to discover spatial synonyms instead of representing those relationships (e.g., meets, overlap) explicitly.

Our work is similar in spirit and proposes a similar architecture to that of GeoVSM [46]. The authors argue convincingly that GIR systems must support two kinds of description (keyword as well as spatial) as well as two kinds of search, although they assume that both sides of the system are providing search capability over the same set of documents. They also recommend that the user interface available in a GIR system include two different user interfaces for the two components because they correspond to distinct ways of representing and organizing information. Our architecture follows the same philosophy, with a separate interface for Metadata++ and the (standard) GIS system. Key differences in our work compared to GeoVSM are that: (1) our system explicitly accommodates non-georeferenceable documents; (2) our GIS component is a standard GIS system that is used to browse various kinds of maps and layers and to select locations (but is not explicitly used to search for documents); (3) our document system, Metadata++, does not use a spatial metaphor to display non-spatial keywords, rather we provide a hierarchical display of terms; and (4) because of the explicit use in this domain of controlled vocabularies to describe places, we are able to easily combine place names with any other (non-spatial) terms in our description and search of documents in Metadata++.

G-Portal [67] is a map-based digital library architecture for georeferenced resources. Like Metadata++, G-Portal provides a map-based interface and a classification interface (to support non-georeferenceable documents). The authors emphasize synchronization between the interfaces – documents selected in one interface will be automatically selected in the other interface. Our work differs from G-Portal in the primary purpose of the map-based interface. In Metadata++, the map-based interface (i.e., the GIS) is not used to specify searches for documents. Rather, it is intended to

search for geographic places so that the place names can be combined with non-geographic search terms in Metadata++ – as well as displaying georeferenceable documents from search results.

7.3 Summary

Geography is an important part of natural resource management – and users within the natural resource community commonly use geographic information systems (GIS). Combining the spatial functionality of a standard geographic information system (such as selecting places on a map) with Metadata++ produces a powerful and intuitive geographic information retrieval system. Our approach includes pre-processing existing GIS datasets to extract vocabularies of hierarchical place names. We encountered several difficulties while implementing vocabulary extraction – including imprecise spatial footprints and the organization of places across different datasets – that required manual review and intervention. Following vocabulary extraction, communication between Metadata++ and the GIS supports place selection, synonym discovery, and document display. We implemented place selection three separate ways, including an ArcMAP extension and a web-based SVG implementation.

8 Related Work, Future Work, and Conclusions

The previous chapters describe various aspects of the Metadata++ digital library system. This chapter compares Metadata++ to other similar systems and presents our conclusions. The final section discusses further research related to Metadata++.

8.1 Comparison of Metadata++ with Similar Systems

Our research extends into several different areas of related work. Related work specific to a particular area is discussed within the appropriate chapter above. This section provides a system-level comparison of Metadata++ to other thesaurus-based applications.

8.1.1 WordNet

WordNet [37] is a popular linguistic thesaurus designed for natural language processing. WordNet includes *synsets* – a set of words that are interchangeable in some context. Synsets are quite similar to polyterms in Metadata++ – although the intention is somewhat different. In WordNet, synsets contain different spellings and inflections of the same (or similar) words. Metadata++ uses polyterms to combine terms that describe the same thing within one context, but also may exist as individual terms in different contexts. The hierarchical relationships in WordNet are of specific types, including holonym (includes), meronym (is part of), hypernym (broader), and hyponym (narrower). In Metadata++, hierarchical relationships are more general – and are used to organize the terms as they exist in the domain terminology.

8.1.2 Library of Congress Subject Headings (LCSH)

The Library of Congress Subject Headings is a popular classification scheme in library science. LCSH defines a set of rules for creating, organizing, and using subject headings [19]. These rules include the USE relationship (and the reciprocal USED FOR

relationship) to relate unauthorized or non-preferred term with the corresponding valid subject heading. In Metadata++, any term in the system may be used for indexing and searching – all terms are first-class. Both LCSH and Metadata++ support polyhierarchies – though Metadata++ is more general in allowing any multiple occurrence of the same term. LCSH includes multiple-concept main headers (e.g., *Electricity in art*) that – unlike Metadata++ polyterms – combine unrelated subject headings for a particular purpose. If appropriate for a particular controlled vocabulary, Metadata++ could represent the same concept using a path-based term: *Art\Electricity*.

As described by Bates [9], the LCSH uses the rule of specific entry – meaning that each book should be classified under the heading that is most specific to the scope and content of the book. The subject headings were then alphabetized to produce a list of subject headings with corresponding books. Because the books were associated with a single subject heading with a certain scope, it was difficult for users to find material other than for the exact scope they were seeking. For example, a searcher looking for 'Cognition' would not find books about 'Memory' (which is one aspect of cognition), because the books about memory were related to the 'Memory' subject heading. Most modern systems, including Metadata++, resolve this issue using some form of thesaurus-based query expansion.

8.1.3 Medical Subject Headings (MeSH)

The MeSH [83,84] system uses generic hierarchical relationships like those in Metadata++. As described in [83], the MeSH hierarchical structure was designed to reflect a view of the literature for a user – without being confined to specific types of hierarchical relationships. The Metadata++ hierarchical structure was designed to reflect the discourses of the various domains. Both MeSH and Metadata++ support polyhierarchies – though MeSH defines polyhierarchies as the same concept within multiple categories (e.g. *Nose* within *Sense Organs* versus *Nose* within *Face*). MeSH does differentiate between 'entry terms' and 'preferred terms', whereas Metadata++ gives each term equal status – without any notion for preferred term.

8.1.4 Art and Architecture Thesaurus (AAT)

Compiled and maintained by the Getty Research Institute [43], the AAT thesaurus follows the NISO thesaurus guidelines [3] for thesaurus construction. The AAT currently

contains approximately 34,000 concepts described by 128,000 different terms. The terms in the AAT may be used to describe such things as art, architecture, and culture. The AAT puts into practice most of the features of a standard thesaurus – including concepts, preferred terms, facets, and polyhierarchies. Besides differences in the application domain, Metadata++ differs from the standard thesaurus model as explained in Chapter 2.

8.1.5 AGROVOC

AGROVOC [41], developed and maintained by the Food and Agriculture Organization within the United Nations, serves users around the world – particularly within agricultural information retrieval systems. Though not specifically based on the thesaurus standard, AGROVOC is a multi-lingual system that includes many of the features of a typical thesaurus – including concepts, descriptors, broader and narrower terms, and scope notes [102]. AGROVOC also includes many relationships types, referred to as *linktypes*, that are very specific to agriculture. These linktypes include relationships such as *pest of*, *grows in*, and *beneficial for*. More relationships, such as *is caused by*, are under consideration. Due to some specific implementation issues, AGROVOC has been limited to a maximum term length of 35 characters [89]. The AGROVOC thesaurus may be downloaded in a variety of formats, including Microsoft® Access, MySQL®, and tagged text. Work is underway to establish an ontology for AGROVOC in both OWL [121] and SKOS [77]. Metadata++ and AGROVOC have some overlap in application domain – but are quite different in scope and structure.

8.1.6 Logic and Language Links (LoLaLi)

LoLaLi [16,68] is an ongoing project a the University of Amsterdam supported by Elsevier Science. The goal of the project is to provide electronic access to scientific handbooks with a particular focus on the scientific literature about logic, language, and linguistics. LoLaLi uses a thesaurus structure very similar to WordNet – including synsets. The hierarchical relationships in LoLaLi are specific to the application domain – including *is a*, *part of*, *technical notion*, and *mathematical result*. A concept is a subtopic of another concept if exactly one of those relationships applies. Metadata++ uses a generic hierarchical relationship that implies various semantics depending on the location in the hierarchy (see Section 2.1.1).

8.1.7 Lundbeck Thesaurus

Nielsen [85] describes the Lundbeck pharmaceutical thesaurus designed for corporate information retrieval tasks. The Lundbeck thesaurus focuses on pharmaceutical concepts and implements several relationship types specific to this industry. These relationship types include equivalence relationships specific for acronyms (ACR), chemical numbers (GN), and trade marks (TM). The thesaurus contains entries for specific drugs along with relationships to other drugs, chemicals, etc. The specific issues in the pharmaceutical industry warrant these types of specific relationships – so the more general Metadata++ thesaurus model may not apply in this industry.

8.2 Future Work

Building the Metadata++ Digital Library generated several interesting ideas that would be interesting to pursue. This section describes different areas of future work that stem from our research thus far and our implementation of the Metadata++ Digital Library.

8.2.1 Exploit path-based terms during automated index

The cost of human indexing is well-known in the information retrieval literature [5,6]. One obvious area of future search is the development of automated (or semi-automated) tools for indexing. We do not believe that an automated indexing system could adequately replace the human indexer, particularly within natural resource management. However, a high-quality automated system could provide valuable assistance to the human indexer and expedite the indexing process. In an earlier version of the system, we built an automated indexing system that would compare documents to the terms in Metadata++ and suggest possible keywords to the indexer [114]. This process presents some interesting research problems. Our implemented system performed quite poorly, even on relatively small documents. During processing, the system would iterate through every term in the hierarchy and then search for that term in the document. With approximately one hundred thousand terms, this process took a long time (several minutes or more) and took much longer on large documents.

We considered a couple of optimizations that would be interesting to pursue. One would be to iterate through the words in the document and look for each word in the

hierarchy. We would expect this approach to improve performance because the average document has far fewer words than there are terms in the hierarchy. This process would be non-trivial, however, because the terms in the hierarchy are rarely a single word – most often they are phrases. We would need to parse the document content into words, then search for each word in the hierarchy. Upon finding matching terms, we would then need to compare the entire term to the word (and its neighboring words) in the document to see if there is a match.

Another optimization would be to semi-automate the process by letting the user initiate the process for specific portions of the hierarchy. For example, while indexing a particular document, the indexer may ask the system to suggest possible *PLACE* keywords. The system would then suggest only terms that are descendants of *PLACE* (that occur in the document). This domain-specific indexing [69] would improve the performance by reducing the number of terms processed at any one time.

Regardless of performance, our implemented system did not benefit from the structure of the hierarchy – it simply iterated through the individual terms in the hierarchy. If the term only occurs once in the hierarchy, then it would be easy to suggest that path as the appropriate keyword. However, when a term has multiple occurrences, we would like to review the path of each occurrence and determine which of the path(s) is most related to the document and suggest the most relevant path(s). For example, consider the document excerpts in Section 2.1.2.1.1. We would like to determine, based only on document content and the hierarchy, which of the multiple occurrences is most relevant for the document.

8.2.2 Digital "workspace"

Most information retrieval systems focus primarily on finding information – but stop when the information is found. Recent research [61,90] focuses on extending the digital library into a digital workspace. At a physical library, patrons often find various books and then sit down at a desk or table to sort through, organize, and understand the information they found. This second step in the information seeking process is just as important, if not more important, then just finding the books. During the user study of Metadata++, we heard several comments from users indicating the need for a digital "workspace". They wanted to be able to save searches for their own future use and to

share with others. They also wanted to be able to select documents and "cache them in a separate window" so they could continue their current search and go back later to read the documents. In addition to saving documents in the workspace, users wanted to be able to explicitly eliminate particular documents from the search result. For example, one searcher saw a document in the search result that he knew was not relevant to his search. He wanted to hide that document and prevent it from showing up in his search results. Buchanan et al. [14] and another one of our projects [24] describe systems that allow the user to visually group and organize information. It would be interesting to take the current Metadata++ digital library application and add digital "workspace" functionality.

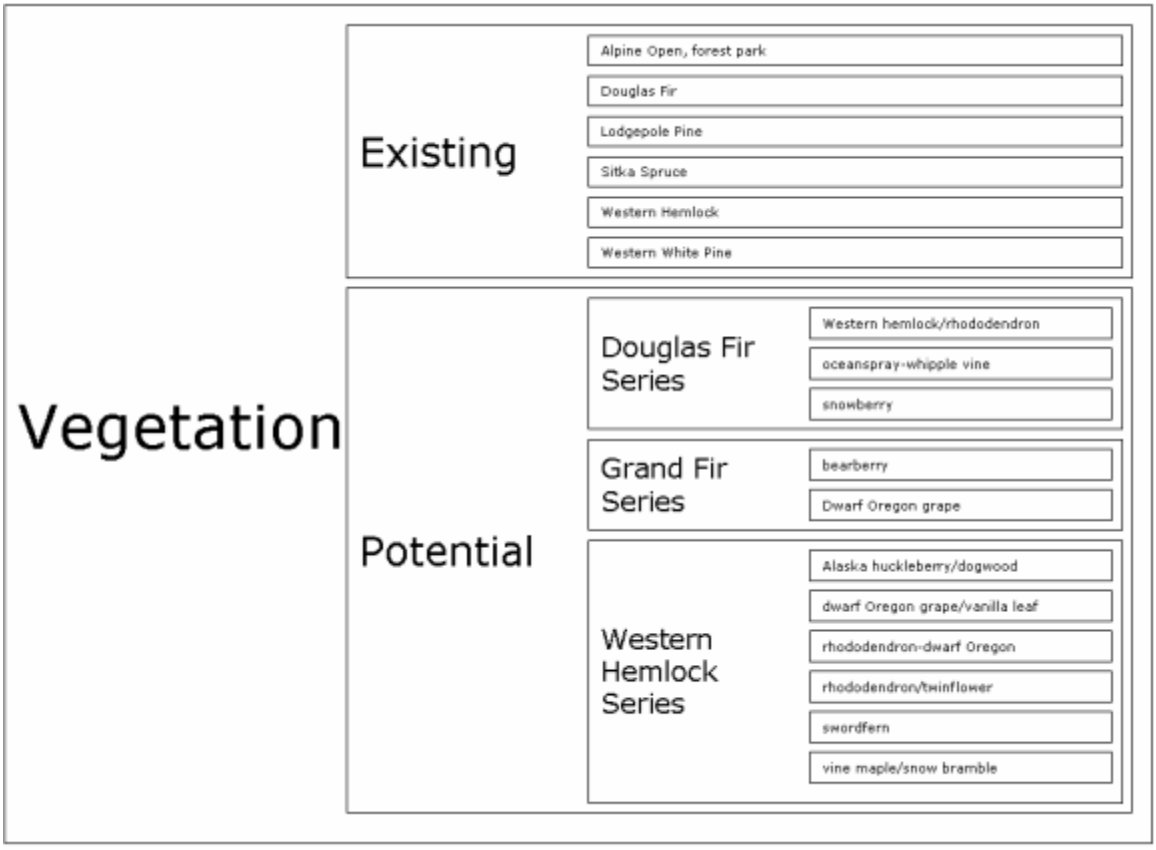


Figure 8-1: Controlled Vocabulary displayed spatially

8.2.3 Spatial Metaphor

As described in Chapter 7, most natural resource managers are very familiar with geographic information systems – and most people in general are familiar with maps. If given a map of their country, most people could quickly point to the approximate location

of their home, birthplace, etc. We explored the possibility of using a map to display controlled vocabularies (as shown in Figure 8-1). In this example, narrower terms are spatially contained within broader terms. This approach might make it easier for users to browse large hierarchies and quickly find terms of interested. Other similar approaches exist and it would be interesting to pursue something like this for Metadata++.

8.2.4 Extend to other application domains

The current Metadata++ system focuses on natural resource management. However, we believe many of the same principles may be useful in other application domains – especially domains with well-known controlled vocabularies. We plan to extend Metadata++ to the health care domain in a new project in collaboration with the Danish Healthcare Ministry. This domain is particularly interesting because medicine and health-related topics have a large amount of well-established, domain-specific terminology. This project allows us to evaluate whether our path-based representation works in a new domain. The Danish Healthcare Ministry is building a health information portal intended for clinicians, practitioners, and the public. Our research thus far has focused on domain experts (natural resource managers). It will be interesting to discover if non-experts (e.g., public citizens) also benefit from path-based terminology.

8.3 Conclusions

Several specific aspects of natural resource management present interesting challenges with regard to information retrieval. We worked closely with Region 6 of the USDA Forest Service (as well as other agencies) to understand the specific information management needs. Our research team included several individuals with a wide range of expertise including computer science, natural resource management, environmental science, marketing and information sharing, and library science. Several years of collaboration helped us learn more about the specific needs of natural resource management – and become familiar with their nomenclature and information processes. The success of our research is largely attributed to the diversity and extensive collaboration of our research team.

As a team, we discovered intricate and interesting relationships within natural resource controlled vocabularies such as multiple occurrences, non-transitive synonyms, and polyterms. We gathered and evaluated several controlled vocabularies that span a

large range of natural resource topics. We built a domain-specific digital library system that uses path-based metadata to represent these vocabularies "as-is" and to utilize the intricate relationships among terms. Path-based terms allow the user to easily understand terms without explicit definitions or scope notes. The Metadata++ software application uses relationships among terms to support user-driven, interactive query expansion.

In addition to understanding controlled vocabularies, we worked together to define architectural objectives for the application. These objectives established a metric that we used to evaluate the backend storage and retrieval mechanism and the user interface. We implemented a path-based storage and retrieval mechanism using four different approaches. After three different implementations utilizing a relational database, we learned that a relational database is not necessarily the best tool for managing path-based data. Our performance evaluation (and user feedback) showed that using the file system, and Microsoft® Index Server, more adequately satisfies all of the architectural objectives.

We also implemented the user interface several times. After attempting two different thin-client applications and a thick-client application, we determined that the best architecture for this particular application is a smart-client. Our smart-client implementation combines a highly-interactive user interface with the deployment benefits of a web application. The client application uses web services to communicate with the server. Overall, the smart-client application satisfied all of the architectural objectives.

Our user study shows that the application is intuitive and usable, and we received considerable positive feedback about the functionality. Several users expressed their desire to see Metadata++ deployed for actual use within the USDA Forest Service. However, budget cuts and other administrative factors have prevented the deployment as of yet. We hope that these factors may, at some point, be overcome so the application may be put into actual use to benefit natural resource managers.

References

[1] Jean Aitchison, Alan Gilchrist, David Bawden.
Thesaurus construction and use: a practical manual.
London : Aslib. (2000).

[2] Jose Luis Ambite, Yigal Arens, Eduard Hovy, Andrew Philpot, Luis Gravano, Vasileios Hatzivassiloglou, Judith Klavans.
"Simplifying Data Access: The Energy Data Collection Project"
IEEE Computer
Volume 34, Issue 2, Pages 47-54, February 2001.

[3] American National Standards Institute – ANSI/NISO Z39.19
"Guidelines for the Construction, Format, and Management of Monolingual Thesauri"
NISO Press (2003).

[4] ASP.NET Portal Starter Kit
http://www.asp.net/StarterKits/DownloadPortal.aspx
Date Viewed: February 2003.

[5] James D. Anderson, José Pérez-Carballo.
"The nature of indexing: how humans and machines analyze messages and texts for retrieval. Part I: Research, and the nature of human indexing"
Information Processing & Management
Volume 37, Issue 2, Pages 231-254, March 2001.

[6] James D. Anderson, José Pérez-Carballo.
"The nature of indexing: how humans and machines analyze messages and texts for retrieval. Part II: Machine indexing, and the allocation of human versus machine effort"
Information Processing & Management
Volume 37, Issue 2, Pages 255-277, March 2001.

[7] Ricardo Baeza-Yates, Berthier Ribeiro-Neto.
Modern Information Retrieval
Addison-Wesley (1999).

[8] Balbinder Banga, Eric Landis, Timothy Tolle, Lois Delcambre, Fred Phillips.
"User needs assessment for the adaptive management portal"
National Conference on Digital Government Research
Los Angeles, California, Pages 329-334, May 2002.

[9] Marcia J. Bates.
"Subject Access in Online Catalogs: A Design Model"
Journal of the American Society for Information Science
Volume 37, Issue 6, Pages 357-376, 1986.

[10] Lena Blomgren, Helena Vallo, Katriina Byström
"Evaluation of an Information System in an Information Seeking Process"
European Conference on Digital Libraries
Bath, UK, Pages 57-68, September 2004.

[11] Philip Bohannon, Juliana Freire, Prasan Roy, Jerome Simeon.
"From XML Schema to Relations: A Cost-Based Approach to XML Storage"
International Conference on Data Engineering
San Jose, California, Page 64, March 2002.

[12] Pia Borlund.
Evaluation of interactive information retrieval systems
Åbo: Åbo Akademi University Press (2000).

[13] Sergey Brin, Lawrence Page.
"The Anatomy of a Large Scale Hypertextual Web Search Engine"
Computer Networks and ISDN Systems
Volume 30, Issues 1-7, Pages 107-117, April 1998.

[14] George R. Buchanan, Ann Blandford, Harold Thimbleby, Matt Jones.
"Supporting Information Structuring in a Digital Library"
European Conference on Digital Libraries
Bath, UK, Pages 464-475, September 2004.

[15] Barbara Buttenfield.
"Usability Evaluation of Digital Libraries"
Science & Technology Libraries
Volume 17, Issues 3-4, Pages 39-60, 1999.

[16] M. Teresa Cabré.
Terminology. Theory, methods and applications
John Benjamins (1999).

[17] Caterina Caracciolo.
"Towards Modular Access to Electronic Handbooks"
Journal of Digital Information
Volume 3, Issue 4, Article No. 157, February 2003.

[18] The CERES/NBII Thesaurus Partnership Project
"Thesaurus::RDF -- The RDF Thesaurus descriptor standard"
http://ceres.ca.gov/thesaurus/RDF.html
Date Viewed: May 2005.

[19] Lois Mai Chan.
"Library of Congress Subject Headings: Principles of Structure and Policies for Application"
Library of Congress, Cataloging Distribution Service (1990).

[20] Hsinchun Chen, Tobun D. Ng, Joanne Martinez, Bruce R. Shatz.
"A Concept Space Approach to Addressing the Vocabulary Problem in Scientific Information Retrieval: An Experiment on the Worm Community System"
Journal of the American Society for Information Science
Volume 48, Number 1, Pages 17-31, January 1997.

[21] Yi-Ming Chung, Qin He, Kevin Powell, Bruce Schatz.
"Semantic indexing for a complete subject discipline"
ACM Conference on Digital Libraries
Berkeley, California, Pages 39-48, 1999.

[22] Phil Cross, Dan Brickley, Traugott Koch.
"RDF Thesaurus Specification (draft)"
Institute for Learning & Research Technology. Technical Report Number: 1011
http://www.ilrt.bristol.ac.uk/publications/researchreport/rr1011/report_html?ilrtyear=00
Date Viewed: October 2000.

[23] Stefan Decker, Michael Erdmann, Dieter Fensel, Rudi Studer.
"Ontobroker: Ontology based access to distributed and semi-structured information"
DS-8: Semantic Issues in Multimedia Systems. R. Meersman et al., editor.
Kluwer Academic Publisher, Pages 351-369, 1999.

[24] Lois Delcambre, David Maier, Shawn Bowers, Mathew Weaver, Longxing Deng, Paul Gorman, Joan Ash, Mary Lavelle, Jason A. Lyman.
"Bundles in Captivity: An Application of Superimposed Information"
International Conference on Data Engineering
Heidelberg, Germany, Pages 111-120, April 2001.

[25] Lois Delcambre, David Maier, Mathew Weaver, Leonard Shapiro, Judith Bayard Cushing.
"Superimposing Spatial Enrichments in Traditional Information"
International Workshop on the Next Generation Geospatial Information
Cambridge, Massachusetts, October 2003.

[26] Lois Delcambre, Timothy Tolle.
"Harvesting information to Sustain Forests"
Communications of the ACM
Volume 46, Issue 1, Pages 38-39, January 2003.

[27] Lois Delcambre, Mathew Weaver, Timothy Tolle, David Maier, Eric Landis, Shawn Bowers, Patty Toccalino, Fred Phillips, Nicole Steckler, Craig Palmer, Julia Norman , Rupini Tummala , Sonali Varde.
"Similarity Search for Harvesting Information to Sustain our Forests"
National Conference for Digital Government Research
Los Angeles, California, Pages 155-158, May 2001.

[28] Alin Deutsch, Mary F. Fernandez, Dan Suciu.
"*Storing Semistructured Data with STORED*"
ACM SIGMOD International Conference on Management of Data
Philadelphia, Pennsylvania, Pages 431-442, June 1999.

[29] Dewey Decimal Classification.
http://www.oclc.org/dewey/index.htm
Date Viewed: January 2005.

[30] Dublin Core Metadata Initiative.
http://www.dublincore.org/
Date Viewed: March 2001.

[31] Susan Dumais, Edward Cutrell.
"Optimizing Search by Showing Results in Context"
ACM SIGCHI Conference on Human Factors in Computing Systems
Seattle, Washington, Pages 277-284, April 2001.

[32] Peter W. Eklund, Philippe Martin.
"WWW Indexation and Document Navigation using Conceptual Structure"
IEEE International Conference on Intelligent Processing Systems
Gold Coast, Australia, Pages 217-221, August 1998.

[33] Environmental Systems Research Institute (ESRI)
MapObjects LT
http://www.esri.com/software/mapobjectslt/index.html
Date Viewed: September 2000.

[34] Environmental Systems Research Institute (ESRI)
"ESRI Shapefile Technical Description"
http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf
Date Viewed: September 2000.

[35] Martha Evens.
"Thesaural Relations in Information Retrieval"
The Semantics of Relationships: An Interdisciplinary Perspective
Information Science and Knowledge Management
Volume 3, Pages 143-160
Kluwer Academic Publishers (2002).

[36] Federal Geographic Data Committee.
http://www.fgdc.gov/
Date Viewed: February 2001.

[37] Christiane Fellbaum (ed).
"WordNet: An Electronic Lexical Database"
MIT Press (1998).

[38] Raya Fidel.
"Searchers' Selection of Search Keys: II. Controlled Vocabulary or Free-Text Searching"
Journal of the American Society for Information Science
Volume 42, Number 7, Pages 501-514, 1991.

[39] Daniela Florescu, Donald Kossmann.
"Storing and Querying XML Data using an RDMBS"
IEEE Data Engineering Bulletin
Volume 22, Number 3, Pages 27-34, September 1999.

[40] F. Fonseca, M. Egenhofer, P. Agouris, C. Câmara.
"Using Ontologies for Integrated Geographic Information Systems"
Transactions in Geographic Information Systems
Volume 6, Number 3, Pages 231-257, 2002.

[41] Food and Agriculture Organization of the United Nations.
"AGROVOC Thesaurus"
http://www.fao.org/agrovoc/
Date Viewed: January 2005.

[42] G. W. Furnas, T. K. Landauer, L. M. Gomez, S. T. Dumais.
"The vocabulary problem in human-system communication"
Communications of the ACM
Volume 30, Issue 11, Pages 964-971, November 1987.

[43] The Gateway to Educational Materials Consortium.
"Monolingual Thesauri Vocabulary"
http://gemstar.ischool.washington.edu/schema/2002/08/15/NISO-Z3919#
Date Viewed: May 2005.

[44] Getty Research Institute.
"About the AAT"
http://www.getty.edu/research/conducting_research/vocabularies/aat/about.html
Date Viewed: May 2005.

[45] Anthony Glenwright.
"Automatically Upgrade Your .NET Applications On-The-Fly"
http://www.devx.com/dotnet/Article/10045/0/page/1
Date Viewed: May 2004.

[46] Cai Guoray.
"GeoVSM: An Integrated Retrieval Model for Geographical Information"
International Conference on Geographic Information Science
Boulder, Colorado, Pages 65-79, September 2002.

[47] Cai Guoray.
"GeoVIBE: A Visual Interface for Geographic Digital Libraries"
Workshop on Visual Interfaces to Digital Libraries
ACM/IEEE-CS Joint Conference on Digital Libraries
Portland, Oregon, Pages 171-187, July 2002.

[48] J. Heflin, J. Hendler, S. Luke.
"Reading Between the Lines: Using SHOE to Discover Implicit Knowledge from the Web"
Workshop on AI and Information Integration
National Conference on Artificial Intelligence
Madison, Wisconsin, July 1998.

[49] Morten Hertzum, Niels Ebbe Jacobsen.
"The Evaluator Effect: A Chilling Fact About Usability Evaluation Methods"
International Journal of Human-Computer Interaction
Volume 15, Number 1, Pages 183-204, 2003.

[50] Timothy A. Howes, Mark C. Smith.
LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol
Macmillan Technical Publishing (1997).

[51] Jane Hunter, Katya Falkovych, Suzanne Little.
"Next Generation Search Interfaces – Interactive Data Exploration and Hypothesis Formulation"
European Conference on Digital Libraries
Bath, UK, Pages 86-98, September 2004.

[52] Mirja Iivonen.
"Consistency in the Selection of Search Concepts and Search Terms"
Information Processing & Management
Volume 31, Number 2, Pages 173-190, 1995.

[53] Infragistics
http://www.infragistics.com/
Date Viewed: May 2005.

[54] H.V. Jagadish, Shurug Al-Khalifa, Adriane Chapman, Laks V.S. Lakshmanan, Andrew Nierman, Stelios Paparizos, Jignesh M. Patel, Divesh Srivastava, Nuwee Wiwatwattana, Y.Wu, C.Yu.
"TIMBER: A Native XML Database"
The VLDB Journal
Volume 11, Number 4, Pages 274-291, December 2002.

[55] Greg Janée, James Frew.
"The ADEPT digital library architecture"
ACM/IEEE-CS Joint Conference on Digital Libraries
Portland, Oregon, Pages 342-350, July 2002.

[56] Bernard J. Jansen.
"Seeking and implementing automated assistance during the search process"
Information Processing & Management
Volume 41, Number 4, Pages 909-928, July 2005.

[57] Bernard J. Jansen, Udo Pooch.
"A Review of Web Searching Studies and a Framework for Future Research"
Journal of American Society for Information Science and Technology
Volume 52, Issue 3, Pages 235-246, 2001.

[58] Christopher B. Jones, Harith Alani, Douglas Tudhope.
"Geographical Information Retrieval with Ontologies of Place"
International Conference on Spatial Information Theory Foundations of Geographic Information Science
Morro Bay, California, Pages 322-335, September 2001.

[59] Christopher B. Jones, R. Purves, A. Ruas, M. Sanderson, M. Sester, M. van Kreveld, R. Weibel.
"Spatial Information Retrieval and Geographical Ontologies: An Overview of the SPIRIT project"
ACM SIGIR Conference on Research and Development in Information Retrieval
Tampere, Finland, Pages 387-388, August 2002.

[60] Raghav Kaushik, Philip Bohannon, Jeffrey F. Naughton, Henry F. Korth.
"Covering indexes for branching path queries"
ACM SIGMOD International Conference on Management of Data
Madison, Wisconsin, Pages 133-144, June 2002.

[61] Claus-Peter Klas, Norbert Fuhr, André Schaefer
"Evaluating Strategic Support for Information Access in the DAFFODIL System"
European Conference on Digital Libraries
Bath, UK, Pages 476-487, September 2004.

[62] Marianne Koch, Lois Delcambre, Patricia Toccalino, Eric Landis, Fred Phillips, Tim Tolle, Len Shapiro, Nicole Steckler, David Maier, Mathew Weaver, Shawn Bowers, Balbinder Banga, Jason Brewster, Afrem Gutema, Sudarshan Murthy, Bill Howe, Rupa Tummala, Julia Norman, Kirsten Zillman, David Drake, Craig Palmer, Ashley Burt.
"The Forest Portal: A Multidisciplinary Project"
National Conference for Digital Government Research
Boston, Massachusetts, Pages 351-354, May 2003.

[63] Jaana Kristensen.
"Expanding end-users' query statements for free text searching with a search-aid thesaurus"
Information Processing & Management
Volume 29, Number 6, Pages 733-744, 1993.

[64] Thomas K. Landauer
The Trouble With Computers: Usefulness, Usability, and Productivity
The MIT Press (1995).

[65] Michael Kifer, Georg Lausen, and James Wu.
"Logical Foundations of Object-Oriented and Frame-Based Languages"
Journal of the ACM
Volume 42, Issue 4, Pages 741-843, July 1995.

[66] Open LDAP Project.
http://www.openldap.org/
Date Viewed: January 2005.

[67] Ee-Peng Lim, Dion Hoe-Lian Goh, Zehua Liu, Wee-Keong Ng, Christopher Soo-Guan Khoo, Susan Ellen Higgins.
"G-Portal: a map-based digital library for distributed geospatial and georeferenced resources"
ACM/IEEE-CS Joint Conference on Digital libraries
Portland, Oregon, Pages 351-358, July 2002.

[68] LoLaLi
http://lolali.net
Date Viewed: October 2004.

[69] Malika Mahoui, Sally Jo Cunningham.
"Search Behavior in a Research-Oriented Digital Library"
European Conference on Digital Libraries
Darmstadt, Germany, Pages 13-24, September 2001.

[70] Jens-Erik Mai.
"Analysis in indexing: document and domain centered approaches"
Information Processing & Management
Volume 41, Number 3, Pages 599-611, May 2005.

[71] Gary Marchionini.
"Evaluating Digital Libraries: A Longitudinal and Multifaceted View"
Library Trends
Volume 49, Number 2, Pages 304-333, 2000.

[72] Brian M. Matthews, Ken Miller, Michael D. Wilson.
"A Thesaurus Interchange Format in RDF"
http://www.limber.rl.ac.uk/External/SW_conf_thes_paper.htm
Date Viewed: January 2002.

[73] Joseph R. Matthews, Gary S. Lawrence.
"Further analysis of the CLR online catalog project"
Information Technology and Libraries
Volume 3, Number 4, Pages 354-376, December 1984.

[74] Robert E. McGrath, Joe Futrelle, Ray Plante, Damien Guillaume.
"Digital Library Technology for Locating and Accessing Scientific Data"
ACM Conference on Digital Libraries
Berkeley, California, Pages 188-194, August 1999.

[75] Microsoft Developer Network (MSDN)
"Dsofile.dll lets you edit Office document properties without Office in Visual Basic .NET 2003 and in Visual Basic .NET 2002"
http://support.microsoft.com/kb/q224351/
Date Viewed: May 2004.

[76] Microsoft .NET Framework Developer Center.
"Smart Client Application Model and the .NET Framework 1.1"
http://msdn.microsoft.com/netframework/programming/winforms/smartclient.aspx
Date Viewed: October 2004.

[77] Alistair Miles, Brian Matthews.
"Review of RDF Thesaurus Work"
http://www.w3c.rl.ac.uk/SWAD/deliverables/8.2.html
Date Viewed: January 2005.

[78] Alistair Miles (ed.), Dan Brickley (ed.)
"SKOS Core Guide"
http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20050510/
Date Viewed: October 2004.

[79] Enrico Motta, Simon Buckingham Shum, John Domingue.
"Ontology-Driven Document Enrichment: Principles and Case Studies"
Workshop on Knowledge Acquisition
Modeling and Management
Banff, Alberta, Canada, October 1999.

[80] Krishna Nareddy.
Indexing with Microsoft Index Server
Microsoft Corporation (1998).

[81] National Marine Fisheries Service, Northwest Region.
"Biological Opinion: Implementation of Interim Strategies for Managing Anadromous Fish-producing Watersheds in Eastern Oregon and Washington, Idaho, and Portions of California (PACFISH)"
National Oceanic and Atmospheric Administration, United States Department of Commerce (1995).

[82] National Research Council.
"Distributed Geolibraries: Spatial Information Resources, Summary of a Workshop"
Panel on Distributed Geolibraries, (1999).

[83] Stuart J. Nelson, Douglas Johnston, Betsy L. Humphreys.
"Relationships in Medical Subject Headings"
Relationships in the organization of knowledge
Kluwer Academic Publishers (2001).

[84] Stuart J. Nelson, Tammy Powell, Betsy L. Humphreys.
"The Unified Medical Language System (UMLS) Project"
Encyclopedia of Library and Information Science
Marcel Dekker, Inc. (2002).

[85] Marianne Lykke Nielsen.
"A framework for work task based thesaurus design."
Journal of Documentation
Volume 57, Issue 6, Pages 774 – 797, 2001.

[86] Marianne Lykke Nielsen.
"Task-based evaluation of associative thesaurus in real-life environment"
American Society for Information Science and Technology
Providence, Rhode Island, November 2004.

[87] Marianne Lykke Nielsen.
The word association method: a gateway to work-task based retrieval.
Doctoral dissertation. Åbo Akademi University Press (2002).

[88] Chris Nowak, Peter W. Eklund.
"Issues in the Efficient use of a Relation Ontology for Conceptual Reasoning"
International Symposium on Knowledge Retrieval, Use, and Storage for Efficiency
Santa Cruz, California, Pages 194-198, 1995.

[89] Gordon W. Paynter, Ian H. Witten.
"A Combined Phrase and Thesaurus Browser for Large Document Collections"
European Conference on Digital Libraries
Darmstadt, Germany, Pages 25-36, September 2001.

[90] Daniela Petrelli, Preben Hansen, Micheline Beaulieu, Mark Sanderson, George Demetriou, Patrick Herring.
"Observing Users – Designing Clarity: A Case Study on the User-Centred Design of a Cross-Language Information Retrieval System"
Journal of the American Society for Information Science and Technology
Volume 55, Issue 10, Pages 923-934, 2004.

[91] Gordon H. Reeves, David B. Hohler, David P. Larsen, David E. Busch, Kim Kratz, Keith Reynolds, Karl F. Stein, Thomas Atzet, Polly Hays, Michael Tehan.
"Aquatic and Riparian Effectiveness Monitoring Plan for the Northwest Forest Plan."
Regional Ecosystem Office, Portland, Oregon, USA (2002).

[92] Tefko Saracevic.
"Digital Library Evaluation: Toward Evolution of Concepts"
Library Trends
Volume 49, Number 2, Pages 350-369, 2000.

[93] Tefko Saracevic, Paul Kantor.
"A Study of Information Seeking and Retrieving. II. Users, Questions, and Effectiveness"
Journal of American Society for Information Science
Volume 39, Number 3, Pages 177-196, 1988.

[94] Tefko Saracevic, Paul Kantor.
"A Study of Information Seeking and Retrieving. III. Searchers, Searches, and Overlap"
Journal of American Society for Information Science
Volume 39, Number 3, Pages 197-216, 1988.

[95] Tefko Saracevic, Paul Kantor, Alice Y. Chamis, Donna Trivison.
"A Study of Information Seeking and Retrieving. I. Background and Methodology"
Journal of American Society for Information Science
Volume 39, Number 3, Pages 161-176, 1988.

[96] P. R. Seaber, F. P. Kapinos, G. L. Knapp.
"Hydrologic Unit Maps"
U.S. Geological Survey Water-Supply Paper
Number 2294, 1987.

[97] Chris Sells.
".NET Zero Deployment: Security and Versioning Models in the Windows Forms Engine Help You Create and Deploy Smart Clients"
MSDN Magazine
http://msdn.microsoft.com/msdnmag/issues/02/07/NetSmartClients/default.aspx
Date Viewed: August 2002.

[98] Leonard Shapiro, Lois Delcambre, Timothy Tolle, Mathew Weaver, David Maier, Dale Guenther, Jason Brewster, Afrem Gutema.
"G-Metadata++: Rich Keyword Search Enhanced with a GIS"
International Conference on Geographic Information Science
Boulder, Colorado, September 2002.

[99] Marios Sintichakis, Panos Constantopoulos.
"A Method for Monolingual Thesauri Merging"
ACM SIGIR Conference on Research and Development in Information Retrieval
Philadelphia, Pennsylvania, Pages 129-138, July 1997.

[100] Smart Client Developer Center
http://msdn.microsoft.com/smartclient/
Date Viewed: October 2004.

[101] Dagobert Soergel.
Indexing Languages and Thesauri: Construction and Maintenance
Wiley-Becker & Hayes Series Book. Melville Publishing (1974).

[102] Dagobert Soergel, Boris Lauser, Anita Liang, Frehiwot Fisseha, Johannes Keizer, Stephen Katiz.
"Reengineering Thesauri for New Applications: the AGROVOC Example"
Journal of Digital Information
Volume 4, Issue 4, Article No. 257, March 2004.

[103] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure.
"Semantic community web portals"
Proceedings of the 9th International World Wide Web Conference
Amsterdam, The Netherlands, Pages 473-491, May 2000.

[104] StreamNet – Fish Data for the Northwest.
http://www.streamnet.org
Date Viewed: December 2002.

[105] Elaine Svenonius.
"Unanswered questions in the design of controlled vocabularies"
Journal of the American Society for Information Science
Volume 37, Issue 5, Pages 331-340, September 1986.

[106] Tamino XML Server.
http://www1.softwareag.com/Corporate/products/tamino/default.asp
Date Viewed: May 2005.

[107] Timothy Tolle, Patricia Toccalino, Lois Delcambre, Julie Norman, Fred Phillips, David Maier.
"Using controlled vocabularies as a knowledge base for natural resource managers"
National Conference on Digital Government Research
Los Angeles, California, Pages 355-362, May 2002.

[108] Marisa Trigari.
"ETB Thesaurus Description and Comments"
http://www.eun.org/eun.org2/eun/en/etb/content.cfm?ov=12233&lang=en
Date Viewed: June 2005.

[109] Howard Turtle, W. Bruce Croft.
"Inference Networks for Document Retrieval"
ACM SIGIR conference on Research and development in information retrieval
Brussels, Belgium, Pages 1-24, September 1990.

[110] United States Geologic Survey.
"Hydrologic Unit Maps"
http://water.usgs.gov/GIS/huc.html
Date Viewed: June 2005.

[111] Mathew Weaver, Lois Delcambre, David Maier.
"A Superimposed Architecture for Enhanced Metadata"
DELOS Workshop on Interoperability in Digital Libraries
European Conference on Digital Libraries
Darmstadt, Germany, September 2001.

[112] Mathew Weaver, Lois Delcambre, Leonard Shapiro, Jason Brewster, Afrem Gutema, Timothy Tolle.
"A Digital GeoLibrary: Integrating Keywords And Place Names"
European Conference on Digital Libraries
Trondheim, Norway, Pages 422-433, August 2003.

[113] Mathew Weaver, Lois Delcambre, Timothy Tolle.
"Metadata++: A Scalable Hierarchical Framework for Digital Libraries"
International Symposium on Digital Libraries and Knowledge Communities in Networked Information Society
Tsukuba, Japan, March 2004.

[114] Mathew Weaver, Bill Howe, Lois Delcambre, Timothy Tolle, David Maier.
"Representing, Exploiting, and Extracting Metadata using Thesaurus++"
National Conference for Digital Government Research
Los Angeles, California, Pages 257-264, May 2002.

[115] XML:DB Initiative for XML Databases.
http://www.xmldb.org/
Date Viewed: January 2003.

[116] Web Services Developer Center.
Microsoft Developer Network (MSDN)
http://msdn.microsoft.com/webservices/
Date Viewed: May 2005.

[117] Peter C. Weinstein.
"Ontology-Based Metadata: Transforming the MARC Legacy"
International ACM Conference on Digital Libraries
Pittsburgh, Pennsylvania, Pages 254-263, June 1998.

[118] A. G. Woodruff, C. Plaunt.
"GIPSY: Geo-referenced Information Processing System"
Journal of the American Society for Information Science
Volume 45, Issue 9, Pages 645-655, 1994.

[119] World Wide Web Consortium (W3C).
"Resource Description Framework (RDF)"
http://www.w3.org/Graphics/RDF/
Date Viewed: June 2005.

[120] World Wide Web Consortium (W3C).
"Scalable Vector Graphics (SVG)"
http://www.w3.org/Graphics/SVG/
Date Viewed: June 2005.

[121] World Wide Web Consortium (W3C).
"Web Ontology Language (OWL)"
http://www.w3.org/Graphics/OWL/
Date Viewed: March 2001.

[122] X-Hive/DB.
http://www.x-hive.com/products/db/index.html
Date Viewed: January 2003.

[123] Xindice.
http://www.apache.org/xindice
Date Viewed: January 2003.

Biographical Sketch

Mathew Jon Weaver was born in June of 1976 in Salt Lake City, Utah. He received a Bachelor of Science degree in Computer Science from Brigham Young University (Provo, Utah) in April of 1999. He received a Master of Science degree in Computer Science and Engineering from the OGI School of Science and Technology at Oregon Health and Science University (Portland, Oregon) in September of 2003. Mathew received a National Merit Scholarship (1993), a Brigham Young University Trustees Scholarship (1993), a National Defense Science and Engineering Graduate Fellowship (2001), and was an honorable mention for the National Science Foundation Graduate Fellowship (2001). His research interests include a variety of aspects of database technology, conceptual models, metadata, and digital libraries. His professional activities include membership in the ACM and the IEEE Computer Society, and he is a Microsoft Certified Professional. Mathew currently serves as Chief Technical Officer for EarthSoft, Inc., a manufacturer of environmental data management software.