A MATHEMATICIAN'S GUIDE TO IMS METADATA

The Overall Structure of IMS Metadata.

The goal of IMS metadata is to create a standardized set of descriptors for educational and, by extension, scientific resources. These descriptors must be both machine and human readable. At a basic level, the descriptors are stored as integers, strings, boolean data, dates, and so on – the same data types commonly found in databases and spreadsheets. Instead of using a database, however, IMS metadata is stored as a tree. This has both theoretical and practical advantages but is perhaps less familiar and therefore requires some acclimatization.

For machine readability, it is enough to produce a well-defined data structure whose consistency and validity can be checked by an algorithm. If we did not need to attach *meaning* to metadata, that would be the end of the story. But if we want to define and find resources using descriptors expressed in a natural language, then the meaning becomes essential. So does agreement upon meaning. The challenge is to find a way to translate natural language and discipline specific descriptions into a machine format that can be efficiently searched over the Internet and, at the other end, translate the results of a search back into natural language descriptions.

Whatever structure is created for describing resources, a key feature must be *extensibility*. If one thinks of the evolution of mathematical subject classifications, it is easy to see that any reasonable classification scheme must accept future modifications without altering the integrity of the current scheme.

IMS Metadata standards provide a structure for accomplishing all of the above. Very precise definitions and rules are required in order to make the definition work at the software level, but for the purposes of understanding how to create and interpret metadata, it suffices to understand the overall picture and a few specifics. The key is to place yourself in the role of a translator whose job it is to convert set of IMS metadata into an intelligible description of a resource. In other words, your job is to "read" the metadata.

Reading Databases. The metadata you are wish to read is given as numbers, words, and dates, so the problem is that of interpretation. You need to answer two questions:

1. *Given a descriptor, what it is describing?*

2. *Given what it is describing, what does it say?*

	Tables define context: Example: "Subject".		
Record Number	Columns further define context: e.g., "LCC" and "Sub-Classification"		Grade Level
1	QA	241	Undergraduate
2	QA	614	Graduate
Some entries, such as "LCC", depend on an external explicit coding scheme. Others, such "Record Number", do not. Still others, such as "Grade Level, are given by natural language descriptors that may or may not be chosen from a standardized vocabulary.			

Figure I.

In a natural language, perhaps especially in English, the meaning associated to a word depends on its *context*. This is also the way things work in a typical spreadsheet or database, and it is instructive to look at this more familiar example first.

Consider a database of that contains information on a list of books. One of the tables in this database might be labeled *Subject* and contain columns labeled *LCC, Sub-classification*, and *Grade Level* (see Figure I). *LCC* stands for the Library of Congress Classification. If we see a value of QA in the *LCC* column. we can look up this value and discover that the subject is mathematics. A 241 in the *Sub-classification* column means that the subject within mathematics is number theory.

In this way the meaning of each cell is determined by three things:

- The table and column in which the cell is found.
- Possibly an external classification scheme, such as the Library of Congress Classification.
- The value of the data in the cell.

The interpretation of the value of data in a cell is context-dependent. In another column in another table the number 241 might represent a price or the number of times a book has been checked out. Nonetheless, it is not hard to correctly "read" data stored in a well designed database. IMS metadata structures work in the same way, as we will now explain.

The IMS Metadata Tree.

IMS metadata are stored in trees. Each tree starts with a *root*. The root is implicit and represents the object being described. Next comes a set of *categories*. Each category represents a general type of property of the root, for example its property rights, how and by whom it was created, and its educational use. Each category has further descendents called *elements* which represent

specific properties within each category. We will see examples below. From there on, elements can have more elements as descendants until we finally reach the leaves of the tree. The leaves contain the actual data.

A basic principle is that each node is an attempt to further qualify the properties of its parent. As one descends through the tree, one progresses from the general to the specific. A path through the tree represents a narrowing of contexts until finally we reach some data. This data must be interpreted in the context defined by the path taken to reach it. To put it another way, each leaf in a tree determines a unique path from the root to that leaf. In reading a metadata tree we must first answer the question: *Given a descriptor, what it is describing?* The descriptors are data that reside in the leaves, and the answer to this question is that *the interpretation of a piece of data is its corresponding path through the tree.*

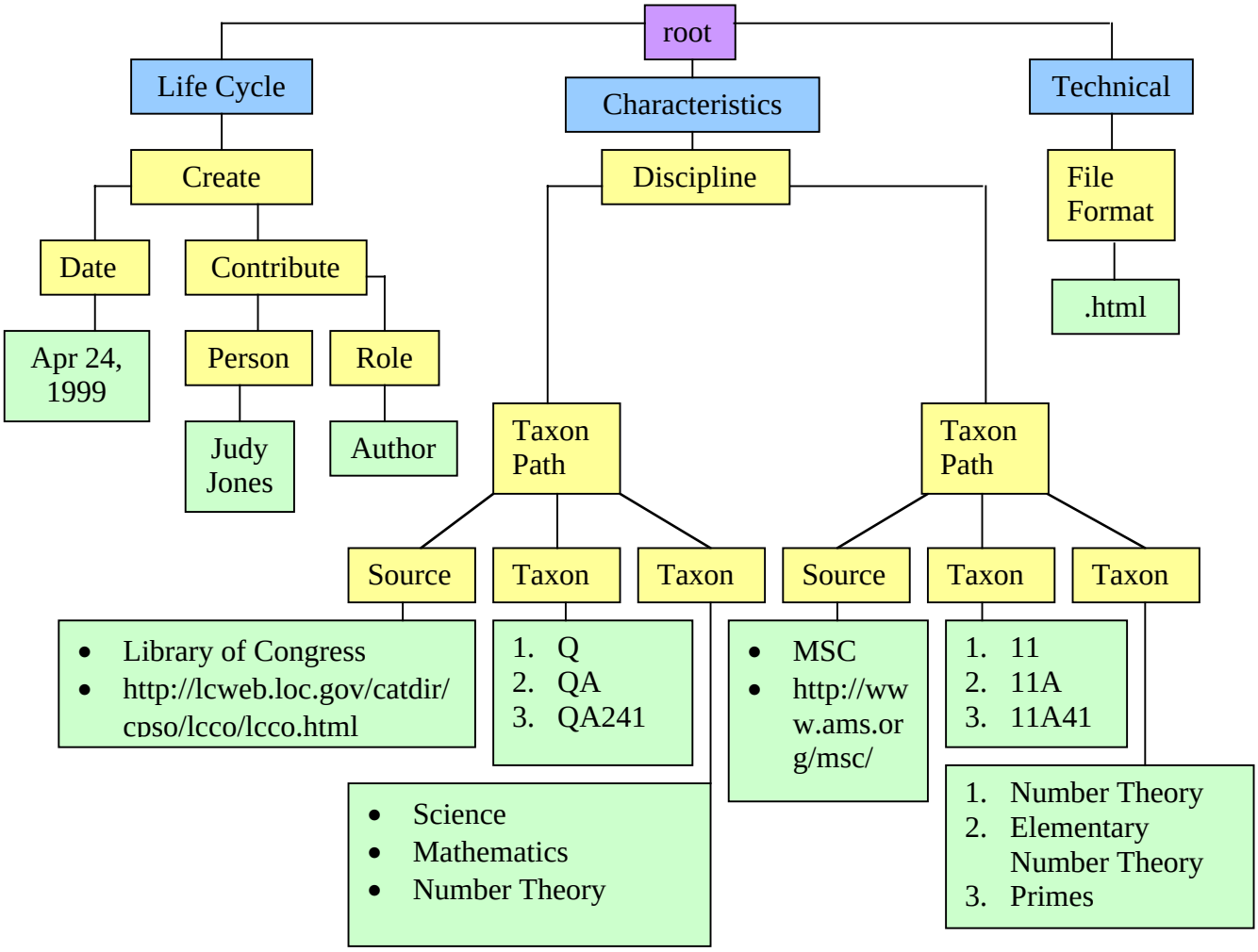

Figure II.

Figure II shows part of an IMS metadata tree meant to describe an entry in a digital mathematical library. The boxes are the nodes in the tree. The immediate descendents of the root are the

categories. These are shaded blue. The yellow boxes are elements and the leaves are green. The text inside the leaves is data. The text inside the categories and elements are their names. We will go now through this example in detail.

To start, look at the right-hand most leaf containing the data ".html". The category in which this appears is *Technical*, which means that we are describing technical (as in technological) properties of the root object. The element *File Format* tells us which technical property we are describing. The value ".html" can now be understood as meaning that the root object is written in HTML, ergo is a Web Page.

Before continuing our exploration of this example, we need to have a convenient way of writing paths through the tree. We will do this by listing the antecedent nodes in order, separated by dots. Thus the path to the leaf containing ".html" is written as *Technical.FileFormat* and the path to the leaf containing "Judy Jones" is written as *LifeCycle.Create.Contribute.Person*. These paths are called the *contexts* and are precisely what tell us how to interpret the data in the leaves.

We can also use this notation for nodes that are not leaves. The path *LifeCycle.Create,* for example, refers to the second node down in the left-most path through the tree. Let us turn our attention to this node for a moment.

The node *LifeCycle.Create* has two immediate descendants, *LifeCycle.Create.Contribute* and *LifeCycle.Create.Date*. The path name *LifeCycle.Create* tells us that this node represents information about the manner in which the root object was created. Therefore the leaf *LifeCycle.Create.Date* is the root object's creation date and the node *LifeCycle.Create.Contribute* contains information about a contribution to the creation of the root object. This node is not a leaf but itself has two descendants. Both descendents must qualify the same contribution. The two leaves *LifeCycle.Create.Contribute.Person* and *LifeCycle.Create.Contribute.Role* therefore combine to tell us that the contribution in question was made by a person whose role was that of an author. The value of the data in the leaf *LifeCycle.Create.Contribute.Person* tells us the author's name was Judy Jones.

We mentioned earlier that the interpretation of the data in a leaf is defined by its path. The path answers the question, "what does the data describe?" and the values of the data answer the question, "what does it say?" Our analysis of the node *LifeCycle.Create* illustrates that this generalizes to an arbitrary node in the tree. Its context tells us what the node describes and the sub-tree rooted at the node tells us the actual description.

The context of a node is the path to the node. This answers the question, "what does the node describe?" The value of a node is the sub-tree rooted at that node. This, in its entirety, gives the actual description.

A consequence of this is that certain elements *must* be included in the sub-tree rooted at a particular node. It would not make sense to specify a person as a contributor without specifying a role. In the middle part of the tree, which we will examine next, it would not make sense to specify a subject classification without saying what classification we are using.

We now turn to the two nodes whose contexts are *Characteristics.Discipline.TaxonPath.* A word of explanation about this vocabulary is in order. In the view of IMS metadata, a *taxonomy* is an external classification scheme. Examples include the Library of Congress Classification, the MSC Subject Classification (the one used by the AMS and Zentralblatt), and Bloom's Taxonomy (familiar to those who study education). Taxonomies may have different parts corresponding to different points of view or properties. The Library of Congress Classification has two parts, a code and a keyword description.

The IMS metadata term *TaxonPath* refers to a description of the root object coming from a single taxonomy. The value of a *TaxonPath* node, which is to say the sub-tree rooted at that node, must include all the information necessary to understand what this description says. In our example, this consists of two pieces of information. First, we need to say which taxonomy we are using. This is done by the leaf *Characteristics.Discipline.TaxonPath.Source.* Notice that the data contained in this leaf is an unordered list. The elements of this list are a human readable name for the taxonomy (LCC or AMS) and a URL. The URL could at least theoretically be used by software to retrieve the taxonomy. Second, we need to use the taxonomy to describe the root object. Each piece of this description is called a *Taxon*. In our example, each *TaxonPath* has two *Taxons*, an ordered list of codes and an ordered list of keywords.

Again, some further explanation is needed. Taxonomies themselves are generally tree structures. Think of the LCC scheme as an example. If we specify an entry, such as QA241, this single code implies all of its parent entries. In other words, if classify the subject of a book as QA241, we automatically know that the book can be found in the library on the QA241 shelf (number theory) of the QA section (mathematics) on the floor containing Q (science). The IMS project calls this the progression from the general to the specific a *taxonomic stairway*. IMS specifications require the explicit inclusion of the entire stairway rather than just the last step. This builds in redundancy needed to facilitate searches. It is more practical, especially given the latency in the Internet, to be able to search directly for the keyword *mathematics* than to be forced to first download enough information to determine all of the keywords which imply the keyword *mathematics* and then to search on them.

Putting this all together, we can now interpret the nodes *Characteristics.Discipline.TaxonPath.* These nodes describe what we call the *subject* of the root object in North American English. The IMS standards are international and consequently use the word *discipline* instead. In our example, two independent taxonomies are used, the Library of Congress Classification and the AMS Subject Classification scheme. In each case the description of the subject matter includes

- a URL that points to the definition of the taxonomy,
- an ordered set of codes giving increasingly specific descriptions of the root object, and
- an ordered set of keywords organized in the same fashion.

The Library of Congress Classification tells us that the Web page being described is in the area of science called mathematics and in the area of mathematics called number theory. The AMS Subject Classification tells us that the page is in the area of mathematics called number theory, in the area of number theory called elementary number theory, and in the area of elementary number theory that deals with primes.

Taxonomic Services and Document Encoding

Our goal in writing this exposition is to aid those who must translate existing digital library classification schemes into IMS compliant metadata and/or who must define new discipline-specific taxonomies. All of this can take place on the human side of the equation without any knowledge of how the results will be implemented. But it is still good to understand a little about how this all works in practice.

The IMS metadata standards contain some implicit taxonomies that were put together from a number of other metadata projects. These are more or less self-explanatory. Thus we should have no trouble interpreting the meaning of *author* in the context of *Characteristics.Create.* But the guts of IMS metadata refers to external taxonomies, many of which do not currently exist. For example, there are no existing universally accepted taxonomies for the educational approach taken by a piece of pedagogic hypermedia or for the level and style of a mathematical proof. Yet IMS metadata anticipates that taxonomies such as these will (and indeed must)

1. be defined,
2. accepted, and
3. *made available to software using IMS metadata.*

What will the software need? At the very least, it will need ways to

- look up definitions of codes and/or keywords and
- check that a taxonomic stairway is valid and consistent.

These are called *taxonomic services*. The *Gateway to Educational Materials* (http://www.thegateway.org) is planning to provide extensive taxonomic services associated for taxonomies written by and for member organizations, but as of this writing very little exists in the way of concrete examples. The Library of Congress Classification scheme exists in books and on Web pages, but to my knowledge it has not been put into a form that can interface with programs seeking to interpret codes or check that codes are valid.

The notion of a taxonomic service is not all that relevant to the defining and acceptance of a taxonomy. We can leave the programming to someone else. The same applies to the exact manner in which IMS metadata is represented in a document, but we should at least understand how a tree can be encoded into a document. At some time, after all, we may be called upon to troubleshoot a document in which that has been done. It turns out that there are two standards, RDF and XML, that can be used for this purpose. We will take a look at XML. A good on-line technical reference is http://www.xml.com/axml/axml.html.

XML stands for "extensible markup language". XML is in many respects like HTML with some very important differences. HTML is a quick and dirty language that specifically tells software how to display Hypertext. It really isn't designed for anything else, although a long set of ad-hoc extensions have added functionality over the pat few years. The advantage of HTML is that browsers can interpret it directly and its disadvantage is that this is the only way it can be used. It also is not "grammatically correct", but that is another story.

XML is a formal grammar that can be used to describe "documents" of any type. One of its intended applications is to describe metadata and one of its underlying principles is that it can be easily interpreted by software. To render an XML document it is necessary to have a dictionary that translates between the XML instructions and instructions to a browser, printer, or other display environment. MathML, for example, is an XML standard that cannot be displayed by current browsers but that can be displayed by the WebEQ Java applet.

What is important for us is that XML has a well-defined *block* structure**.** Each block must be properly opened and closed. This creates a containment relationship among blocks with no loops, and hence a tree with an implied root. Figure III shows an XML version of the *LifeCycle* sub-tree of the example tree given in Figure II.

```
<LIFECYCLE>
     <CREATE>
          <CONTRIBUTE>
               <ROLE>
                    <A_VOCABULARY>
                         <![CDATA[Author]]>
                    </A_VOCABULARY>
               </ROLE>
               <PERSON>
                    <A_PERSON>
                         <A_NAME>
                              <A_FIRSTNAME>
                                   <![CDATA[Judy]]>
                              </A_FIRSTNAME>
                              <A_SURNAME>
                                   <![CDATA[Jones]]>
                              </A_SURNAME>
                         </A_NAME>
                    </A_PERSON>
               </PERSON>
          </CONTRIBUTE>
     </CREATE>
</LIFECYCLE>
```

Figure III

It is pretty easy to read off the tree structure from this listing. However, one might have expected something more simple, as is illustrated in below.

```
<LIFECYCLE>
     <CREATE>
          <CONTRIBUTE>
               <ROLE>
                    "Author"
               </ROLE>
               <PERSON>
                    "Judy Jones"
               </PERSON>
          </CONTRIBUTE>
     </CREATE>
</LIFECYCLE>
```

Figure IV

XML is a little more demanding than the code in this simplification, but what really is happening is that the human readable data that occupy the leaves of the IMS Metadata tree are, from the point of view of software, not really data at all. Before we can get to including real data, we need to define its type, and these types themselves consist of IMS metadata *sub-schemes.* For our purposes we can ignore this level of detail. We can think of trees as we did in Figure II and think of the block structure of trees as in Figure IV. We should just be prepared to read the more complete XML versions if and when the need arises.

IMS Specifications From My Point of View

A major reason for writing this document is to facilitate the translation of existing mathematical taxonomies into IMS-compliant metadata. To do that, we need to become familiar with the elements available to us in the IMS scheme. This section contains some sketchy information and makes some editorial comments about the IMS specifications in relation to this task.

The current working document for IMS metadata specifications is charmingly called DID188 and is available at

http://www.imsproject.org/technical/metadata/library/DID188.html.

It is probably a good idea to go through that document, especially the tables at the end that lay out the IMS metadata schema in table form. We will be using the entire scheme (the *master* scheme) as a starting point but may want (or need) to define our own restriction and/or extension. Here are a few of the specifications from the master scheme that seem important to our work.

- **Categories**

  The IMS master scheme contains nine categories.

1. Annotation.
2. Characteristics.
3. Educational Use Dependent.
4. General.
5. Life Cycle.
6. Meta-Meta-Data
7. Relation.
8. Rights Management.
9. Technical.

The ones that we will have to work with the most are *Characteristics*, *Educational Use Dependent, Relation,* and *Technical.* Three of these contain pointers to external taxonomies that we will need to define and for which we will need to seek acceptance. The fourth, *Relation,* is supposed to describe relations among resources. This will presumably be used to describe the relation between a theorem and an example. Since these relations are essential to mathematical exposition, we might be the ones who are forced to come to grips with them.

Other categories will come into play, but probably in ways that are common to many disciplines and applications.

- **Elements**

  The IMS scheme defines a long list of elements (technically, *semantic elements,* or *data elements.*) Not all categories contain all elements and the meaning of an element is dependent on its context. To illustrate the type of taxonomies we are faced with defining, here is a list of some of the elements that appear in *Characteristics*

  - *Characteristics.Discipline.TaxonPath.* The AMS classification will work very well for higher level mathematics, but we need to address school and college level mathematics as well. Existing taxonomies exist but must be standardized. Moreover, it is desirable to have *international* standards, which means that we cannot assume that all educational systems are the same as those in North America.

  - *Characteristics.Concept* According to the vocabulary suggested by the IMS Project, this is where taxonomies describing proofs, examples, explanations, etc. would go. In thinking about some things I would like to find on the Internet, I came very quickly to the need for a taxonomy for proofs. For example, I would like to be able to find proofs of the fundamental theorem of arithmetic pitched at the undergraduate level. I would also like to know that these proofs are induction arguments. WOW!

  - *Characteristics.Coverage* This is a term from the Dublin Core. The definition (according to the Dublin Core) is

    > *The spatial and/or temporal characteristics of the resource. Formal specification of COVERAGE is currently under development. Users and developers should understand that use of this element is currently considered to be experimental.*

See http://purl.org/DC/about/element_set.htm#. I don't really understand this one!

- *Characteristics.Type* This is an ordered list describing the type of the document. Suggested vocabulary includes tutorial, exercise, example, etc. Can you imagine what it will take to *agree* on the definition of a definition?

- *Characteristics.Approach* This is defined as being the pedagogy. We will need to define a pedagogical taxonomy. The problem is actually not defining one but choosing among a large number of existing ones. Do we try to use them all? Just some? Which ones?

The entire category of *Educational Use Dependent* seems to need a lot of definition. In mathematics we will have some special needs for *Technical* since we use discipline specific software (computer algebra systems) and formats (such as $\TeX$, MathML, or DVI). These will not present much of a problem although they will require some thought.

Design Principles, No Examples, and Going Off the Deep Mathematical End

There do not seem to be a plethora of examples from which we can draw wisdom about designing taxonomies. However, we have been given one word of advice: *don't overload any categories or elements.* The principle behind this is that of *orthogonality.* As far as I can tell (and I do not necessarily agree with this model) what this means is the following.

Let us define a space D of all on-line documents. Associated with D is another space M of metadata descriptions. The IMS specification tell us how to define (but do not themselves define) a map $\varphi: D \to M$. This map is not injective. The process of searching for documents using metadata is the process of computing $\varphi^{-1}(C)$ where $C \subset M$ is a set of search criteria.

We can go on to think of M as a vector space with the metadata elements as a basis. This is potentially an infinite-dimensional space, but we will only work with a projection onto the finite-dimensional subspace spanned by the metadata elements that have actually been defined. For our purposes, identify *M* with this projection. Also, for the sake of making the model work, think of all data values as real numbers, possibly after re-coding other forms of data.

The values of metadata are the coordinate functions on *M*. What we want to do is to endow M with an inner product in which the metadata elements become an *orthogonal* basis and in which the induced metric corresponds to a human notion of what it means for two documents to be similar. If that can be accomplished, searching would be greatly facilitated. This would mean, for example, that the set of documents containing a proof of the fundamental theorem of arithmetic pitched at the undergraduate level and written in a particular style would be $\varphi^{-1}(U)$ for a small open subset $U \subset M$. The smaller the open set, the more accurate the search.

To do this, we need as a precondition that metadata elements are independent in terms of our human interpretation of similarity. For otherwise the implicit metric defined by us and the theoretical metric defined on M cannot be equivalent. By the same token, if we put more

information in one element than in another, then φ will tend to have larger pre-images for smaller sets. This will make the results of a search less meaningful, very much as they are now.

Searching by Trees

If we think about searching using metadata trees, we encounter the same two elements discussed when learning how to read metadata trees: *context* and *description.*. There are two parts to the notion of what it means for documents to be "close". One notion of proximity is defined solely by the tree structure. The deeper the nodes on which we are searching, the finer the net we are casting. The other notion is defined by the descriptions (and ultimately the data) themselves. Thus once we have agreed that two documents will be "close" if they were published at nearly the same time, we then need to look at the publication dates.

There is, however, something very interesting about trees: *Any node of a tree can be interpreted as its "root", and if one starts comparing trees at different root nodes, one ends up with different notions of proximity.* This makes very good sense. As an example, suppose that a mathematics education researcher is interested in issues of gender equity in middle school. He starts by looking for material that is (1) about mathematics, (2) about mathematics education, (3) about mathematics education in middle school, (4) about teacher attitudes in middle school classrooms, and finally (5) about teacher attitudes along gender lines in middle school classrooms. His search produces a paper entitled

> "A Comparison of Mathematics Problems Assigned to Male and Female Students by Eighth Grade Teachers in Coeducation Classrooms."

At the same time, a feminist philosopher is interested in the same topic: gender equity in middle schools. She performs a search by looking for material about (1) gender equity, (2) gender equity in education, (3) gender equity in middle school classrooms. Her search produces the same paper. *However, papers that the first researcher would consider "near" to the given one are not the same as those that the second would consider "near."* If we change the word "mathematics" to "biology" in the title of the paper, it is probably no longer of any interest to the mathematics education researcher but is of equal interest to the feminist philosopher. Similarly, a paper on mathematics problems assigned to male students in single sex classrooms might be of interest to the mathematics education researcher but not to the feminist philosopher.

Browse Structures, Keyword Searches, and Metadata Searches.

Both the Math Forum and the Eisenhower National Clearinghouse have what are called *browse structures.* The same is true of a search site like *Yahoo*. A browse structure is a tree. One searches through the tree by refining categories. In that sense it is similar to a metadata structure with two important differences. First, if the browse structure is fixed, we cannot "pick the tree up by a different node" and thereby change the notion of when two documents are close. Secondly, and more substantially, this only uses the tree structure and not the actual data in the leaves of the tree. Once you get down to the last level of a browse structure, you are confronted with an undifferentiated list of documents.

The NEEDS project has discovered (and this may also apply to the other two projects) that *keyword searches* are more natural and effective. Rather than trying to define a browse structure into which documents are placed, the NEEDS project simply associates keywords to documents. The user then looks for documents containing a desired set of keywords. This is also a common search strategy used by major search engines. Keyword searches are the equivalent of using only the data in the leaves of a metadata tree, and then only the leaves which contain discrete data.

Some search engines and digital libraries combine these two strategies, allowing keyword searches within predefined structures, or sometimes returning nodes in a browse structure that contain documents matching the keyword search. This is much closer to the type of searching that metadata would allow. Restricting to a browsing categories is the equivalent of looking for metadata in just a few contexts. Looking at for keywords at this point is the equivalent of looking for specific values of the metadata attached to the given context.

All of this is great until we start picturing the user. From what we understand of studies on user behavior (no references cited!), very few users can effectively apply any common Web search strategy, let alone a more complicated one.

Patton's Suggestion. The ineffectiveness of user searching is a big problem in education. Picture a student studying in an on-line setting. Say the student is studying the quadratic formula and wants to find some examples of it. If there is a direct link to an example, the student will click it. If not, the student will do nothing. The reason is that even if the student wanted to browse for examples, she probably wouldn't know what terms to enter. The student may not even be aware that the formula is called the quadratic formula! And if she did enter "quadratic formula", she would not be able to usefully continue searching among the 2617 pages returned by Altavista, 857 returned by Infoseek, 940 returned by Yahoo!, and 550 returned by Lycos (I just checked for fun).

Our ultimate goal is attach metadata to every on-line resource but have the metadata remain invisible to the student. Students could initiate searches using it. One could envision a pull-down menu with "Examples", "Explanations", "Exercises", and "Applications" as choices. The student would make a choice and behind the scenes the software would

- Look up metadata associated with the particular object being studied,
- Look up metadata from a profile of the student using the software, and
- Combine this metadata to search for what the student needs.

Each search could include parameters (such as the grade level) that are specific to the student as well as to the educational resource. If the student is using the equivalent of an on-line text, the metadata-enabled search engine might also give more weight to resources from the same "text". Thus, when the student looks for exercises, the relevant exercises in the text would be displayed first, although others might also appear.

What metadata enables is a change in user behavior. Asking the question "show me something like the thing I am looking at" is much more natural than going through some combination of browse structure and key word search. Moreover, it is an approach that can work with a fuzzy

understanding of what is being sought. In a learning situation, we should *expect* the user to have a fuzzy understanding.

End of Document

This document is a work in progress, to be continued after various meetings of the mathematics metadata working group. Please feel free to send comments to me at any time!

Robby Robson, Oregon State University. Email: robby@orst.edu. Work Phone: 541-737-5171.

Shouldn't the metadata about this document go here?