

Utilizing Open Source and Object Oriented Paradigms in Instructional Technology

Terence C. Ahern

Institute for Communications Science and Technology, California State University, Monterey Bay
Seaside, CA 93955, USA

and

Nancy K. Van Cleave

Department of Mathematics and Computer Science, Eastern Illinois University
Charleston, IL 61920, USA

ABSTRACT

A *learning space* is defined by the intersection of *content*, *learner*, and *instructional designer*. Each of these components provide critical contributions to the learning process. Computer mediated instruction can model the concept of a learning space by separating the content from the delivery mechanism, and recognizing software as an extension of the instructional designer. In this model, content is composed of *learning objects*, broadly defined as reusable digital resources, and the instructional designer is responsible for determining the most effective delivery model and presentation sequence of the learning objects chosen to compose the lesson. Thus, computer solutions to facilitate learning should 1.) be based on learning objects 2.) give instructors the ability to choose model(s) of instruction, 3.) create and follow the sequencing determined most efficient by the designer, 4.) support the growing trend toward standards and open source solutions, and 5.) be robust and flexible.

Keywords: instructional design, instructional technology, learning objects, lesson sequencing, learning space, learning technology, open source, object oriented paradigm, web-based education

INTRODUCTION

The proliferation of digital resources continues to grow at an exponential rate [1] providing unprecedented access to information. This deluge threatens to overwhelm us in sheer volume. Ninety-eight percent of U.S. K–12 schools have Internet access (as of Fall 2000), with a ratio of students to instructional computers standing at 5 to 1, according to recent government statistics [2]. As late as 1999, however, only 52% of public school teachers reported instructional use of computers or the Internet during class time [3]. Student enrollment in higher education online courses more than doubled during the years from 1995 to 1998 [4], as the creation and dissemination of content also kept pace. However the completion rate of online courses is significantly lower compared to traditional courses, with perhaps 10 to 20 percentage points difference [5]. Globally, organizations lose approximately \$750 billion annually due to workers wasting time trying to find and capture information necessary for them

to do their jobs [6]. All of this adds up to an expensive under-utilization of computers. Obviously, “if you present it, they will learn” is not necessarily true: there are still obstacles to utilizing fully the power of computer mediated instruction.

Thus a major conundrum for instructional technology is how to transform the continuously expanding content into *effective* instruction. Two critical problems stemming from this challenge are:

1. How to quantify, classify, and establish relationships between units of information, and
2. How to utilize instructional design to best present these packets of information to the learner for the highest rate of understanding and retention.

A strong movement toward establishing standards and open source solutions to utilize computers in education exists. Among the current projects encompassing these goals are the Advanced Distributed Learning (ADL) initiative (a collaboration of the U.S. government, industry, and academia) [7]; MIT’s OpenCourseWare (OCW) project [8]; the IMS Global Learning Consortium Inc. [9]; and the IEEE Learning Technology Standards Committee (LTSC) [10]. This latter group was formed in 1998 specifically to develop the guidelines necessary to support the implementation of the *learning objects* approach, which was intended to be a solution to the first problem as noted above. Their stated purpose was to establish instructional technology standards with the intent to create environments that “automatically and dynamically compose personalized lessons” [11]. However, there is within the proposed standards a general lack of direction on just how this is to be accomplished, since “no one had considered the role of instructional design in composing and personalizing lessons” [12].

The second problem is directly related to this omission—how to incorporate instructional design into the development of courseware and other computer facilitated instruction. Robert Gagne, an educational researcher, asserts that learning is a form of information processing—it is progressive (i.e., sequential) and builds upon previous knowledge [13]. IMS, as one example, is currently involved in creating “a set of specifications for content sequencing—how one item of course content might follow another... how to

arrange various learning objects” [14]. Learning objects must be combined “in a way that [makes] instructional sense, or in instructional design terminology, ‘sequencing’ the learning object” [11, p. 11] if computer managed instruction is to make progress. Educational authoring software must allow the designer to determine the ordering of the content making up a lesson or course. This ordering may not be unique—there may be several paths through the material—and every object in the lesson may not be necessary to the learner’s understanding (for example, if the learner doesn’t need any review).

The Mentor Project is designed to explore exactly the issues introduced above by mediating between learning objects and instructional sequencing. In order to better understand both the theoretical underpinnings of Mentor and its design, we will investigate the learning process in humans, an *Instructional System* model where we look at the role of an instructional designer, learning objects and metadata, and the application of the open source paradigm to CMI before describing the Mentor Project.

GAGNE’S NINE EVENTS OF INSTRUCTION

In order to more successfully automate the instruction process, we would do well to understand the process from an educational point of view. Gagne [13] breaks down the instructional design process into nine steps which are used to support the internal process of learning. These may additionally be grouped into three categories:

1. *Preparation*
 - (a) Gain attention
 - (b) Describe goal
 - (c) Stimulate recall of prerequisite material or knowledge
2. *Acquisition & Performance of Skills*
 - (a) Present lesson content
 - (b) Provide lesson guidance
 - (c) Elicit performance
 - (d) Provide feedback (reinforcement) to learner about performance
3. *Retrieval & Transfer of Skills*
 - (a) Assess performance
 - (b) Enhance retention and transfer

Most often this is an iterative process where satisfactory performance allows the student to proceed with the material, otherwise more practice or quite possibly review of foundational information is in order before the learner can achieve the goal. What is relevant to our project, however, is that this provides us with a well defined roadmap to the learning process. A learner is evaluated (for placement) and begins a lesson with, it may be assumed, certain initial knowledge, then proceeds through the content toward

one or more goals. The incorporation of content and attainment of its particular knowledge or skill goal is what we are calling a *learned state*. Before progressing from one learned state to the next, assessment takes place and the student’s understanding is (again) evaluated. The result of assessment determines, perhaps in conjunction with learner preferences, the next learning experience. These concepts can be modeled with finite state diagrams (or, equivalently, by a Finite State Machine or regular grammar), with *learned state nodes* linked together into a network, one of which is an initial *start state node* and one or more of which are final *goal state nodes*.

THE INSTRUCTIONAL SYSTEM MODEL

A *learning space* is defined by the intersection of *content*, *learner*, and *teacher* or *instructional designer*, as shown in Figure 1. Each of these components provides critical contributions to the learning process. In order to utilize computers to mediate instruction, we need to understand clearly the interactions between these elements.

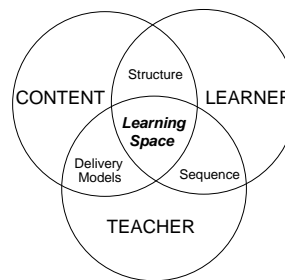


FIGURE 1. The Instructional System Model

Content organization is directly dependent on the student. The age and current knowledge status of the learner are important factors in determining the type of structure(s) to utilize in a lesson. An abstract structure that starts with an abstract definition might be very beneficial to an adult learner, but beyond the ability of a child to comprehend—a younger learner may require concrete structures.

The designer is responsible for selecting or creating content which is appropriate in terms of information, teaching method, and learner level. In addition, they must be able to design the most appropriate delivery model, given that content. The choice is not monolithic, but depends on the type and organizational structure of the content. Joyce and Weils [15] cataloged a variety of teaching models and grouped them into four major categories: (1) behavioral, (2) information processing, (3) social interaction, and (4) personal source. This provides developers with a very useful tool to categorize delivery methods.

The designer is also responsible for the creation of an effective sequence of learning experiences that matches the content with the developmental level of the learner. Here we define sequencing as “the or-

der in which elements of subject matter, including information, skills, and cognitive strategies are taught during instruction” [15, p.23] The teacher not only selects, but also must regulate the sequence of material by pacing the learning experiences most effectively for the learner. This is a critical factor, and the teacher must constantly evaluate the readiness of a student. Given specific content structure, the teacher must judge whether or not the student is ready to proceed to the next target within the learning sequence.

The individual elements of content, teacher, and student come together to form a learning space, which defines the lesson environment. Time is the primary constraint on the ability to deliver instruction within this framework. The teacher, which could be an automated system or a human, mediates between the learner and the content by managing the delivery of the material in an appropriate sequence. In real time systems, the teacher is free to adapt to changes in the environment (including feedback from the student) by speeding up, slowing down, or altering the delivery of material and exercises. Within automated environments, this is much more difficult to accomplish.

We conclude this section by drawing attention to the separation of content from its delivery mechanism within the learning system model. This is an important concept, with similarities to and the same benefits as the object oriented paradigm of computer programming languages. Extending this separation into a computer implementation of the learning system model will reduce costs while enhancing reusability and flexibility [17].

META-METADATA

Learning objects are typically viewed as reusable digital instructional resources (examples: a PDF document, a URL, or a java applet), but are very widely defined (Wiley, pp. 2-7). One popular design describes a learning object as containing both *content* and *delivery method* (Figure 2). According to Koper [18], this view of a learning object is consistent with the IEEE proposed definition.

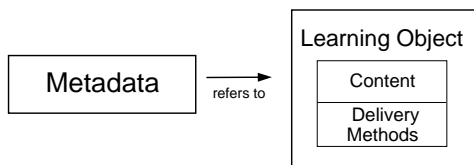


FIGURE 2. Meta-Learning Object

Along with the learning objects themselves, additional information, called *metadata*, is required in order to automate utilization of the object. Basic characteristics such as a description, human language(s) used, keywords, structural form, and requirements (e.g., application software needed for viewing), along with other sorts of related information should be included in this metadata as indicated by the Draft

Standard for Learning Object Metadata [1]. In order to parse and process this metadata, its actual form and structure must itself be described—in other words, data about the metadata, or *meta-metadata* is necessary. Learning object meta-metadata describes the metadata, rather than the learning object the metadata describes.

Why is this necessary? It is a key element in separating lesson content from the delivery mechanism, allowing a public interface to be associated with the learning object to facilitate its reusability. The meta-metadata describes the *meaning* of the metadata associated with the learning object.

OPEN SOURCE / OPEN STANDARDS SOLUTIONS

To integrate seamlessly with other Internet applications, educational software must take advantage of the openness of the Internet and utilize the public interfaces of learning objects. These protocols (definitions of relevant data types and how they are stored) are crucial to the open interchange between various computers and users. Published standards encourage third party add-ons and free exchange of content. A prime example of how openness encourages system extensions is Linux. Many Linux components come out of Richard Stallman’s GNU Project [19] and the Free Software Foundation [20], which are open forums that encourage additional components for Linux.

Extensible Markup Language (XML), an open standard for hypertext markup, allows users to define their own hierarchy of data, which can be published by using Document Type Definitions (DTDs). According to Forester Research [21], “XML-defined metadata... can embed structured information within data streams that can be used to discern the underlying meaning of the searched material... [It] will facilitate more efficient tagging of content.” Once a DTD is created, it is possible to define custom sets of data that can work in the same way as other XML files already being used by an application. Conversely, given a DTD and a data file which conforms to it, any application which can parse the DTD has access to the meaning of the data.

THE MENTOR PROJECT

The Mentor Project incorporates instructional design into a course authoring and management system while utilizing an object oriented paradigm within an open source approach. Conceptually, the designer mediates within a learning space between the content (learning objects) and the learner. The name Mentor was chosen to represent this process, and the interface for the sequence editor was developed to maintain the notion of the learning space with mediation as the fundamental metaphor.

The complete system will be comprised of several components, minimally:

- Content Editor—to create meta-learning object group sequences from digital resources
- Learning Space Sequence Editor—to create *learned state nodes* (from meta-learning object groups) and sequence them via a *learning space map*
- Mentor—to manage student progress through lessons consisting of learning space maps and learning objects

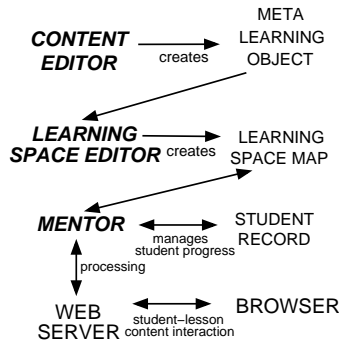


FIGURE 3. The Mentor Project

The Content Editor will allow a designer to select learning objects, such as URLs, electronic documents, or java applets, and combine them into a learning object group. The Learning Space Sequence Editor will then use such groups and learning objects to create learned state nodes, which may be linked together to sequence the content, resulting in a learning space map. Finally, Mentor will use the map to present the material to the learner. The relationships between these components are shown in Figure 3. An XML DTD for a meta-learning object group will link the content editor output with the sequence editor. A second XML DTD for the lesson sequence map then links the output of the sequence editor with the Mentor application.

Meta-Metadata for Content

The Meta-learning object group DTD includes the format for information about the type of instructional method (direct or cognitive) embodied in each content component for the group, as well as the type, name, and location of each element. An individual developer can quickly identify different learning objects from a variety of vendors, and insert them in the group sequence in an appropriate order. This expedites the inclusion of these objects within the design space.

```

<!ELEMENT source (content)>
<!ELEMENT content (name,models)>
<!ELEMENT models (direct,cognitive)*>
<!ELEMENT direct (method)*>
<!ELEMENT cognitive (method)*>
<!ELEMENT method (type,name,data)*>
<!ELEMENT type (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT data (#PCDATA)>
  
```

FIGURE 4. Meta-Learning Object Group DTD

By utilizing this definition for meta-learning object groups, we enable instructional designers to incor-

porate reusable learning objects from within a content discipline, then create an index file from the Grouped Content DTD (produced by the Content Editor). This facilitates searching groups for particular models, methods, names, etc. Figure 5 illustrates an example output file resulting from a content editor session. This file conforms to the DTD shown in Figure 4 and contains specific learning objects which the designer wished to group together to include in a particular lesson. It can be used by the Sequence Editor as a building block in a larger lesson.

```

<!DOCTYPE source SYSTEM "content.dtd">
<source>
<content>
<name>Sentence Structure</name>
<models>
<direct>
<method>
<type>nominal</type>
<name>Noun definitions</name>
<data>http://faculty.csumb.edu/noun_defs.html
</data>
</method>
<method>
<type> nominal</type>
<name>Verb definitions</name>
<data>http://faculty.csumb.edu/verb_defs.html
</data>
</method>
</direct>
</models>
</content>
</source>
  
```

FIGURE 5. Meta-Learning Object Group Data File

The example demonstrates the content for an English lesson on sentences, and contains the description of two basic elements within a sentence—nouns and verbs. Notice the content description describes the delivery method and the links to the location of the learning objects. Mentor needs this information in order to create dynamic delivery information within an appropriate learning sequence. In this example, both of the objects were simple web sites, but the sequence editor is capable of utilizing any type of learning object as long as it is accessible.

Sequence Editor Output

Using the instructional system as our guide, we designed the lesson sequence DTD that defines the resulting output from the sequence editor (Figure 6). The resulting output file associated with this document type definition will in turn be used as input to a course management system which will use the inherent sequencing within the document to automate presentation of the lesson to students.

The sequence editor output DTD defines the lesson sequence map for engaging the student. A major feature of the DTD is a *learned state node* which defines some general metadata available only to other instructional developers. This metadata includes such items as the node's name, a general description of the content, and the goal for this particular state node. In essence, a node describes all the necessary learning

experiences required for a learner within its specific content sequence.

The state node also provides a mechanism for linking the specific learning objects or groups together in a particular order. These objects can be web sites, local documents, or other types of learning objects. Finally, the state node also identifies its own exit criteria, which is the required student performance established by the teacher/designer for the particular lesson. The sequence editor allows multiple target or goal nodes with different performance requirements.

```

<!ELEMENT StateMap (lesson)>
<!ELEMENT lesson (name, content)>
<!ELEMENT content (node)*>
<!ELEMENT node (ID,location,name,description,
               stategoal,methods,targetStates,
               criteria)*>
<!ELEMENT location (x,y)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT stategoal (#PCDATA)>
<!ELEMENT methods (line)*>
<!ELEMENT line (#PCDATA)>
<!ELEMENT targetStates (stateNode?)*>
<!ELEMENT stateNode ((source,sHndl,target,tHndl))>
<!ELEMENT source (#PCDATA)>
<!ELEMENT target (#PCDATA)>
<!ELEMENT sHndl (#PCDATA)>
<!ELEMENT tHndl (#PCDATA)>
<!ELEMENT criteria (line?)*>

```

FIGURE 6. Lesson Sequence DTD

The Sequence Editor Prototype

Educational authoring software developed to facilitate the creation of electronic lessons should be based on the following simple design principles [17]:

1. Utilize a cross-platform, open system model, combining minimum cost with maximum flexibility
2. Provide an easy to learn and use graphical interface
3. Follow the instructional model system discussed in a previous section

To incorporate these principles in the lesson sequencing component of the *Mentor* system, we selected *Java* as our programming language. Current versions of the *Java* virtual machine and development environments are stable and run on multiple platforms with little or no need for revision.

The prototype was designed with the underlying instructional model in mind. A developer begins a new lesson in the editor by naming it. The next step is to register content specific learning objects and learning object groups. The editor will read the file and parse it according to the name and delivery type, and additional information can be entered and associated with this node. Once an object is registered, it may be viewed (in a new window) by the designer.

Now the user can proceed to develop the lesson sequence and tailor it to individual learner performance. Learned state nodes can be Added to the lesson, and appear as labeled circles positioned wherever the designer wishes within the lesson window.

The user enters more information, termed a **State Description**, about the node—such as vendor or content for the specific learning objects, and the intended general outcome(s). This information may be used by other lesson developers or teachers when this lesson is made accessible or shared. Nodes can be linked together in a sequence determined entirely by the designer.

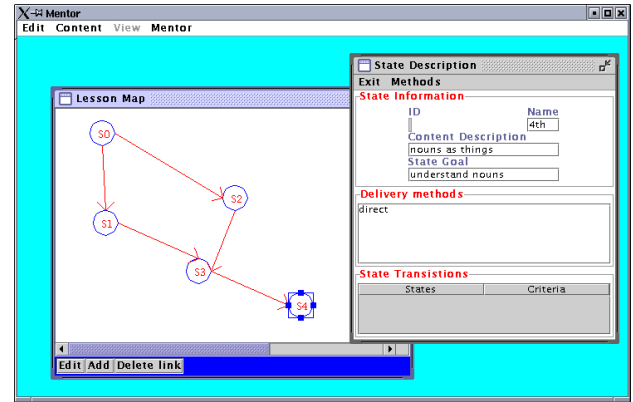


FIGURE 7. Sequence Editor and Map

The developer provides a sequence of nodes built around individual student performance within each learned state node. As illustrated in Figure 7, the designer has created five nodes with various links connecting the **source** or **start** node S_0 with the **target** or **goal** node S_4 .

The learning state Sequence Editor allows the designer to easily develop and provide multiple learning paths through the specific content structure of a lesson. The system tracks the number of state criteria required for each of the target links within the learning space. These alternative paths can provide the learner with additional content or exercises, if warranted by the individual student's performance. Thus, if the student's performance in a state is less than satisfactory, the designer can provide automatic alternatives for the learner. A specific student would then be directed to any one of possibly several **Review** states based upon their level of achievement. These **Review** states are intended to be remedial in nature, providing a transition for the student to the desired level of understanding. In this way, the system can accommodate multiple learners and multiple learning styles yet still remain based upon learner performance.

The result is a state diagram, where nodes represent one or more learning objectives and include one or more learning objects to help the student attain the particular objective. One or more nodes will be designated as **target** or **goal** states which contain the criteria for exit from the lesson with satisfactory achievement. There may be multiple paths from the **start** node to **goal** nodes, allowing the learner some flexibility by selecting their own path through the diagram. If at any node the student fails to meet that

node's criteria (as set by the lesson designer), they can be routed through alternative nodes in order to review or reinforce their understanding of background material. This provides for a transition in the student from unsatisfactory to satisfactory performance.

FUTURE REFINEMENTS

The learning state Sequence Editor represents a single application within the **Mentor** Project. One component yet to be developed is the Content Editor, which will allow content vendors to categorize specific learning objects by the intended delivery method. Along these same lines, issues of middleware and course management need to be addressed. **Mentor** itself, the presentation and management application also has not been implemented at this time. Further partitioning of **Mentor** to facilitate the assessment, management, and feedback mechanisms is expected.

Formal usability studies are planned for the **Mentor** Project. Anecdotal feedback from educators indicates that a simple mechanism for creating transition objects within the state descriptions appear to be very useful.

SUMMARY

The growth of content development and dissemination has outstripped our ability to design and implement computer managed instruction. The IEEE has recognized this and created the LTSC to address this new need. Learning objects have been suggested as a possible solution. However, specific implementation guidelines are lacking. The **Mentor** Project was conceived as one way to incorporate instructional design into instructional technology using the open source paradigm.

The goal of the learning state Sequence Editor prototype was to mimic as closely as possible the mediating aspect of lesson design. The user interface approaches this as a design metaphor. The design space allows great flexibility in how the designer will lay out the lesson in terms of each state description, as well as the global issue of learner state transitions and sequences. The designer has control over the sequencing of the lesson, but leaves the pace up to the individual learners to decide through lesson performance.

The editor presented in this paper investigated the feasibility of automating the sequencing of automated instruction. This editor followed a specific instructional system and attempted to mediate the development of an automated lesson composed of learning objects and their relationship to one another within the lesson.

REFERENCES

[1] IEEE LTSC P1484.12 LOM Working Group, Wayne Hodgins, Chair, (18 April 2001), "Draft Standards for Learning

Object Metadata," Version 6.1, [Online]. Available: ltsc.ieee.org/doc/wg12/LOM_WD6-1_1_without_tracking.pdf

[2] Cattagni, A. and E. Farris Westat May 2001), "Internet Access in U.S. Public Schools and Classrooms: 1994-2000," National Center for Education Statistics, Office of Educational Research & Improvement, U.S. Department of Education, Washington DC [Online]. Available: nces.ed.gov/pubs2001/2001071.pdf

[3] Williams, C. (February 2000), "Stats in Brief: Internet Access in U.S. Public Schools and Classrooms: 1994-99," National Center for Education Statistics, Office of Educational Research & Improvement, U.S. Department of Education, Washington DC [Online]. Available: necs.ed.gov/pubs2000/2000086.pdf

[4] Lewis, L., E. Farris, K. Snow, and D. Levin (1999), "Distance Education at Postsecondary Education Institutions: 1997-98," National Center for Education Statistics, Office of Educational Research & Improvement, U.S. Department of Education, Washington DC [Online]. Available: nces.ed.gov/pubs2000/2000013.pdf

[5] Carr, S. (February 2000), "As Distance Education Comes of Age, the Challenge Is Keeping the Students," Chronicle of Higher Education, Information Technology, [Online]. Available: chronicle.com/free/v46/i23/23a00101.htm

[6] Kearney, A.T. (2001), "Network Publishing: Creating Value through Digital Content," A.T.Kearney: Santa Clara, CA [Online]. Available: atkearney.com/pdf/eng/Network_Publishing_Study_S.pdf

[7] Advanced Distributed Learning (ADT) web site, www.adlnet.org

[8] OpenCourseWare Project, MIT, web site, web.mit.edu/ocw

[9] IMS Global Learning Consortium Inc., web site, www.imsglobal.org

[10] LTSC, Learning Technology Standards Committee (2001, April), web site, [Online]. Available: ltsc.ieee.org/

[11] Learning Technology Standards Committee (2001, April), IEEE Standards Board: Project Authorization Request (PAR) form, [Online]. Available: ltsc.ieee.org/par-1o.htm

[12] Wiley, D.A., (2001) "Connecting learning objects to instructional design theory: a definition, a metaphor, and a taxonomy," Utah State University, Digital Learning Environments Research Group, The Edumetrics Institute. [Online]. Available: reusability.org/read/chapters/wiley.doc

[13] Gagne, R., L. Briggs, and W. Wager, "Principles of Instructional Design," Holt, Rinehart, and Winston, 1988.

[14] Syllabus Magazine, "Standards Update: An Interview with Ed Walker," (April 2002), SyllabusWeb [Online]. Available: www.syllabus.com/syllabusmagazine/article.asp?id=6239.

[15] Joyce, B., M. Weils, with E. Calhoun, *Models of Teaching*, Allyn & Bacon Publishing Company, 2000.

[16] English, R.E., and C.M. Reigeluth, "Formative research on sequencing instruction with elaboration theory," *Educational Technology Research and Development* (ETRD), **44**(1), 1996, p. 23-42.

[17] Ahern, T.C., N. Van Cleave, and B. York, "Open Protocols for Web-based Educational Materials," Frontiers in Education, Reno, NV, Oct. 2001.

[18] Koper, R., (2001, June) "Modeling units of study from a pedagogical perspective: the pedagogical meta-model behind EML," Educational Technology Expertise Centre, Open University of the Netherlands. [Online]. Available: eml.ou.nl/introduction/articles.htm

[19] GNU's Not Unix! web site, www.gnu.org

[20] The Free Software Foundation web site, www.gnu.org/fsf/fsf.html

[21] Forester Research (1999, January), "Guided Search for eCommerce," referenced in [6].