**Closeout Notice Date**: 03-DEC-1998

**Project Number**: C-50-651  
**DoCh Id**: 45989

**Center Number**: 10/24-6-R8529-0A0

**Project Director**: STASKO, JOHN

**Project Unit**: COMPUTING

**Sponsor**: NAVY/OFC OF NAVAL RESEARCH

**Division Id**: 3314

**Contract Number**: N00014-95-1-0790  
**Contract Entity**: GTRC

**Title**: USING COGNITIVE PRINCIPLES TO DESIGN MULTIMEDIA TRAINING ENVIRONMENTS...

**Effective Completion Date**: 30-SEP-1998 (Performance) 30-SEP-1998 (Reports)

**Closeout Action**:  

<table>
<thead>
<tr>
<th>Closeout Action</th>
<th>Y/N</th>
<th>Date Submitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Invoice or Copy of Final Invoice</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Final Report of Inventions and/or Subcontracts</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Government Property Inventory and Related Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Classified Material Certificate</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Release and Assignment</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

**Comments**

**Distribution Required**:

<table>
<thead>
<tr>
<th>Distribution Required</th>
<th>Y/N</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Director/Principal Investigator</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Research Administrative Network</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Accounting</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Research Security Department</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Reports Coordinator</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Research Property Team</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Supply Services Department/Procurement</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Georgia Tech Research Corporation</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Project File</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**: Final Patent Questionnaire sent to PDPI
Document Header Id #: 45989
Project #: C-50-651
Center #: 10/24-6-R8529-0A0
OCA file #: Project type: RES
Contract #: N00014-95-1-0790
Prime #: Cost share #: : ADMINISTRATIVE
Cost share #: Center shr #: Project shr #: 
Center shr #: Contract shr #: 
CFDA: 12.300
Prime #: 
Project unit: COMPUTING
Project director(s): PDPI- STASKO COPDPI- GUZDIAL COPDPI- RAM
(404)894-5617 (404)894-5618 (404)894-4995

Sponsor amount New this change Total to date
Contract value: 0.00 382,046.00
Funded: 0.00 320,000.00
Cost sharing amount: 0.00 0.00

Title: USING COGNITIVE PRINCIPLES TO DESIGN MULTIMEDIA TRAINING ENVIRONMENTS...

Sponsor technical contact: HELEN M. GIGLEY, ONR 342 OFFICE OF NAVAL RESEARCH BALLSTON TOWER ONE 800 NORTH QUINCY STREET
Phone: 7036964502 Fax: ARLINGTON, VIRGINIA 22217-5660
Email: Security class (U,C,S,TS): U Defense priority rating : OCN

Sponsor issuing office: ADMINISTRATIVE GRANTS OFFICER ONR
Sponsor technical contact: HELEN M. GIGLEY, ONR 342 OFFICE OF NAVAL RESEARCH BALLSTON TOWER ONE 800 NORTH QUINCY STREET
Phone: 7036964502 Fax: ARLINGTON, VIRGINIA 22217-5660
Email: Security class (U,C,S,TS): U Defense priority rating : OCN

Supplemental sheet: ONR resident rep is ACO (Y/N): Y
Supplemental sheet: ONR resident rep is ACO (Y/N): Y

Administrative comments - Administrative revision issued to reflect correct budget distribution, error was caused during conversion. Budget now reflects actuals.
Our project focuses on developing multimedia systems to support human learning, based on cognitive principles and guidelines from cognitive science. The question is not so much whether multimedia makes a difference, but rather how can it best be deployed to make a difference? Specifically, what combinations of media and methods of interaction are most effective for learning, and why? Cognitive science has made significant advances in understanding human learning and training issues. Based on these advances, we can make strong hypotheses about how to construct effective interactive multimedia learning environments. We are using these hypotheses as the basis for a principled approach to the development of multimedia training systems and, in turn, further advance our understanding of learning and of media through careful assessment of the systems. In our initial investigations summarized here, our focus has been primarily on whether encouraging learners to interact in different ways with the system affects their learning, learning rate, and or transfer to new problems.
Our project focuses on developing multimedia systems to support human learning, based on cognitive principles and guidelines from cognitive science. The question is not so much whether multimedia makes a difference, but rather how can it best be deployed to make a difference? Specifically, what combinations of media and methods of interaction are most effective for learning, and why? Cognitive science has made significant advances in understanding human learning and training issues. Based on these advances, we can make strong hypotheses about how to construct effective interactive multimedia learning environments. We are using these hypotheses as the basis for a principled approach to the development of multimedia training systems and, in turn, further advance our understanding of learning and of media through careful assessment of the systems. In our initial investigations summarized here, our focus has been primarily on whether encouraging learners to interact in different ways with the system affects their learning, learning rate, and or transfer to new problems.

May 15, 1995 - November 14, 1995

Important Findings and Problems Encountered

We have conducted two initial studies to examine the effects of cognitive media types and self-explanations on learning in multimedia systems. One experiment was in the domain of chemistry (determining molecular shapes) and the other was in the domain of computer graph theory. While the experiments are described in more detail below, the major findings are that the type of learning orientation induced in students when interacting with different versions of the systems leads to different browsing (through the system) patterns but does not appear to lead to differences in test performance. However, these initial findings must be tempered by the observation that these studies were designed to debug the materials as much as to debug and learn about the consequences of various manipulations to the
multimedia learning environments.

Changes in Overall Plan and Personnel
None.

Experiments

Chemistry

In an attempt to test some of our ideas about cognitive media types and self-explanation, we collaborated with a Chemistry professor at Emory University in the design of instructional software for teaching students how to solve problems involving the determination of molecular shape. This topic is particularly difficult for introductory Chemistry students so the potential benefits in this domain are considerable.

The software, which we called ChemLab, is based on an outline of the problem solution procedure developed by our domain expert (the Chemistry professor). The steps in the procedure for determining molecular shape were laid out in a map that students could follow to arrive at a solution. Each step contained four cognitive media types that learners could utilize:

1. Definitions. Key concepts relevant to this step and the operations required at this step of the procedure.

2. Examples. Concrete examples of this step in the procedure (or the key concept in the step).

3. Worked problems. This was generally a "before and after" presentation in which the students saw a partial solution before the step was performed and then after it was performed. Explanations for the operations were also present.

4. Problem sets. This is similar to a worked problem but provides an opportunity to test one's knowledge. The screen presents a "before the step" situation and asks the learner to execute the step in their head or on scratch paper. The learner can request that the solution then be shown to verify their solution.

Because this particular domain is very visual, ChemLab makes extensive use of figures along with the text within all of the cognitive media types. However, use of other physical media types were limited. The semi-public computer cluster environment made it difficult for us to use sound, and time constraints made construction of animations impossible. Future versions of ChemLab will use animation to help illustrate the more dynamic content, though the use of sound is still an unresolved issue.

The participants in the ChemLab experiment were approximately 80 undergraduates who enrolled in an introductory Chemistry course at Emory university, who participated in the experiment for extra-credit. Participants were given paper "cheat sheets" to help them navigate the ChemLab software. The software itself was run on Macintosh Centris personal computers and developed in Apple's HyperCard (version 2.3).
All participants had access to the same instructional material in the software and were given the same post-test after using the software. The post-test was a short quiz designed by the regular chemistry professor for the course.

The ChemLab software was presented to participants in one of four conditions:

1. Passive watching. In this condition, participants were given access to the ChemLab instructional material and little guidance. They were instructed to "work through the materials until you are pretty sure you are prepared to solve some problems." This is the control condition to which others can be compared.

2. Directed watching. In this condition, we wanted to give the participants some learning goals while browsing through ChemLab. Thus, participants were given a series of questions to answer to help guide their browsing. These questions concerned different aspects: some were oriented towards the procedure (e.g. "how do you do X?" or "what do you do after step Y?"); some were oriented toward specific content (e.g. "what's the chemical formula for Hydrazine?"); some were definitions, and the rest were miscellaneous things that participants would have to find.

3. Problem solving. Participants in this condition could not only browse ChemLab, but were asked to solve two specific molecular shape problems. As they were doing so, they were allowed to use ChemLab as a resource to help them solve the problems (including a feature called "The Molecule Construction Kit"). This condition was included to encourage participants to spontaneously generate learning goals directly relevant to solving real problems.

4. Problem solving with prompted self-explanations. This condition was identical to condition 3 with one addition: at various points while participants solved the problems, they were prompted to explain what they were doing and why. It was hoped that this manipulation would increase reflection and self-explanation.

Participants were run over five consecutive school days spanning seven calendar days. They were run in a semi-public Macintosh cluster on the Emory campus in groups of up to six.

After filling out the consent form, participants logged into the ChemLab environment, which handled condition assignment and data collection. While browsing ChemLab, participants' mouse clicks were logged and time stamped. After participants finished working with ChemLab, they were given the post-test.

While all of the data have been collected, the log files have not yet been extensively analyzed, due to their complexity. However, some preliminary results are available. First, there were no reliable group differences in the overall post-test scores—analyses for individual questions on the post-test are still pending. On the other hand, there are clear differences in the amount of time participants in the different groups spent browsing ChemLab. Browsing patterns and use of the various cognitive media types also appear to differ across the experimental conditions. Details of these differences will be examined in the near future. Overall, it seems clear that the experimental manipulations did affect the usage of the ChemLab software, though it is not yet clear what the relationship is between usage patterns and post-test performance.
Computer Graph Theory

The Computer Graph Theory module taught students all about graph data structures and their use. It touched on topics such as connectivity, directed vs. undirected, completeness, and included related algorithms and data structures such as shortest path, minimum spanning tree, and searches.

Participants were 148 undergraduate Georgia Tech undergraduate students enrolled in an introductory computer programming course.

The Algonet2 software was installed over a network of personal computers spanning two undergraduate labs. The computers were IBM 486 compatibles running Windows 3.1 with 14" SVGA displays.

The students received a packet of written instructions and a problem to be solved requiring the application of graph theory. Students' interaction was logged to files for later analysis. After the students had completed the lab and shut down the Algonet2 program, they were asked to complete a post-lab questionnaire that asked them questions about their academic background, exposure to and usage of computers as well as their impressions about different parts of the Algonet2 system. A fourth source of data came from four students who volunteered to have their interactions with the system videotaped as well.

Interaction with the system was mouse-driven. The students clicked on buttons (e.g., back, quit, help, help-on-help), boxes in the topic tree, and lines of text in the question-asking interface.

The system contains groups of pages, each group focusing on one topic in graph theory. Preliminary analyses of the data are just beginning.

Publications

None

Papers Submitted for Publication

None

Papers in Refereed Conference Proceedings

None

Paper Presentations

We will be presenting the preliminary findings of our work at the workshop we are co-sponsoring in New Mexico in February.

Upcoming Research

We plan to rerun the chemistry experiment in a more controlled setting and perhaps with more challenging problems. These changes will accomplish two things: first, the increased
problem difficulty might allow us to detect learning differences among the experimental groups; second, the more controlled setting will allow us to more closely monitor and shape learners' interactions with the system in order to determine whether the manipulations can affect learning. While this increased control will make it more difficult for us to determine learners' "natural" interactions with the system, the control is necessary in order to for us to determine whether the manipulations can affect learning. Additional work can then look at how easily and naturally learners' make use of various features and whether they like them.

We also plan to do additional studies with the Algonet2 software that will match the manipulations done in chemistry. The importance of this approach is to allow us to demonstrate the generality of our findings and to see if the instructional approaches we are developing can be applied to multiple domains.
Our project focuses on researching and defining architecture guidelines for developing multimedia systems to support human learning. We base these guidelines on principles from cognitive science. The question is not so much whether multimedia makes a difference, but rather how can it best be deployed to make a difference? Specifically, what combinations of media and methods of interaction are most effective for learning, and why? Cognitive science has made significant advances in understanding human learning and training issues. Based on these advances, we can make strong hypotheses about how to construct effective interactive multimedia learning environments. We are using these hypotheses as the basis for a principled approach to the development of multimedia training systems and, in turn, further advance our understanding of learning and of media through careful assessment of the systems. In the work summarized for this report, our focus has been primarily on architecture development for multi-media systems and on empirical results that address whether encouraging learners to interact in different ways with a system affects their learning, learning rate, and/or transfer to new problems.

May 15, 1996 - November 14, 1996

Important Findings and Problems Encountered

We adapted the software developed to examine the effects of cognitive media types and self-explanations on learning in a multimedia tutoring/training system. The content domain is chemistry and the software is called ChemLab. Our earlier results with ChemLab, described in last report, suggested that learners’ browsing patterns were affected by the "orientation" given to learners. That is, some students were required to answer specific "fact" questions during training and could search through the software to get answers. Other students were required to solve practice problems during training while still others were allowed to browse through ChemLab with the goal of "learning" it. We expected students’ browsing patterns to differ as a function of their orientation. While we found differences in browsing and the use of cognitive media types among the conditions, these differences did not seem to affect performance on the problem solving post-test.
We conducted a new study in chemistry during the Fall of 1996 to address limitations in the earlier studies that may have made it difficult to find group differences on the post-test. One limitation in the earlier studies was that a "paper only" control group was not used; such a group might perform differently compared to groups that used the computer-based training system even if the computer-based groups do not differ from each other. In addition, the post-test problems used in the earlier studies needed to be refined in order to have a more systematic relationship with the training materials and thus allow us to look for expected group differences on certain aspects of the problems. We are just now analyzing the browsing data and post-test problem solving data from the new study.

We have realized that we need to explore a variety of cognitive media roles and a variety of navigation systems over these roles. Continuing to create new systems each time we wish to adapt the cognitive media roles displayed or the style of navigation is a waste of resources. A great deal of flexibility is needed in the multimedia system in order to conduct the experiments we wish to run. We want to be able to easily alter factors such as available navigation techniques, order and style of materials presented, and types of media available. For example, one future exploration is to provide learners with different initial goals as well as different versions of the system that we expect to affect learners' browsing patterns. Thus, in parallel with the chemistry research, we are developing an architecture for a computer-based training and tutoring system that will allow us to easily change the task domain, navigation style, cognitive media roles, and materials of study. We have chosen the domain of home repair as our initial testbed for the architecture since this domain presents interesting challenges for people in terms of learning mental models and specific troubleshooting and repair procedures.

Our current effort in the architecture is focused on a production rule system with a variation of a semantic net to store knowledge and media (associated with the knowledge through cognitive media roles). The production rule system will encode the kinds of tasks that students may wish to undertake, the kind of navigation to use, and the kind of cognitive (or physical, depending on the experiment) media to present. The knowledge structure will reflect both procedural and systemic knowledge (e.g., how to repair a leaky faucet, how to install a new sink, as well as how the hot water system works), with connections to a wide range of physical media through cognitive media roles, such as definitions, justifications, process visualizations, and the like. The description below provides more detail on how we are currently designing our semantic nets.

Types of nodes

A knowledge node represents a concept that the system knows about. Information about the concept is represented using media nodes, which can be presented to the user, and relationships between concepts are represented using knowledge links.

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition/Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thing</td>
<td>Represents a system, physical object, part, or substance in its normal state. Things may be composed of other things.</td>
<td>Hot water system (system).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faucet (physical object).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Washer (part).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Water (substance).</td>
</tr>
<tr>
<td>State</td>
<td>Each thing has one or more states that it can be in. A thing's normal state is the usual operational state of that thing; other states represent problem conditions that may need repair. Usually only problem states are represented explicitly; the main representation of a thing is assumed to represent its normal state.</td>
<td>Pilot light off.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faucet leaky.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Washer worn out.</td>
</tr>
</tbody>
</table>
Procedure

Represents a sequence of actions carried out by the user that operate on a thing in some manner. The actions that comprise a procedure may themselves be procedures; ultimately, this bottoms out when the primitive action is operationalized and can be directly carried out by the user. The desired level of detail may be dependent on the expertise of the user.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Lighting a pilot lamp.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Replacing a leaky faucet.</td>
</tr>
<tr>
<td></td>
<td>Installing a shower.</td>
</tr>
</tbody>
</table>

**Media nodes**

A media node contains the actual information about a knowledge node that the system can display to the user. Media nodes are characterized by the content of the information that they contain, the physical media in which that knowledge is represented (text, video, etc.), the mode, and the modality of the interaction with the user. A media node may contain more than one type of physical media. A knowledge node may contain one or more media nodes; these are organized using cognitive media roles.

**Types of links**

**Cognitive media roles**

The media nodes within a knowledge node are organized in terms of the cognitive roles played by the information contained in the media node in the problem-solving task that the user is engaged in. For example, a particular mixed-media text-and-pictures description of a faucet may serve as an example of a single-lever faucet; in this case, the knowledge node for single-lever faucet will contain an example slot which contains the media node that represents that text-and-pictures description.

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition/Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title/Name</td>
<td>The name of the item being represented in the knowledge node. Usually one or a few words of text.</td>
<td>Pilot light.</td>
</tr>
<tr>
<td>Definition/Description</td>
<td>The definition of the concept being represented in the knowledge node or, more informally, its description. Usually a textual description accompanied by diagrams.</td>
<td>A pilot light is a small gas flame that is continually burning. It is used to light the furnace when ... ... [Schematic of pilot light]</td>
</tr>
<tr>
<td>Example</td>
<td>An example of the concept being represented in the knowledge node. Usually a textual description accompanied by diagrams.</td>
<td>[Photograph of an actual pilot light] This is an example of a pilot light. In this design, the small lever to the left [pointer to picture] is used to ...</td>
</tr>
<tr>
<td>Counter-example</td>
<td>An example of an alternative design, an alternative state, etc. Usually a textual description accompanied by diagrams.</td>
<td>[Photograph of a flint-based lighting system.] An alternative to the pilot light is the ...</td>
</tr>
<tr>
<td>Process simulation</td>
<td>A step-by-step simulated description of how something works. Usually an animation or movie accompanied by a narrative, which may be a voice-over or a textual commentary.</td>
<td>[Video showing how a pilot light lights the gas burners in a heating unit.] [Voice describing what is happening.] The pilot light causes... [short textual sentences accompanying the video]</td>
</tr>
<tr>
<td>Justification</td>
<td>An explanation of a step being carried out in a procedure.</td>
<td>You want to turn off the water at the supply first because ...</td>
</tr>
</tbody>
</table>
**Knowledge links**

Knowledge nodes may be linked to other knowledge nodes using knowledge links that represent conceptual relationships between those knowledge nodes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition/Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is-a</strong></td>
<td>Thing is-a Thing Represents the broader category of a thing, or equivalently the particular types of a thing.</td>
<td>Single-level faucet is-a Faucet Pilot light is-a Ignition device</td>
</tr>
<tr>
<td><strong>Has-a</strong></td>
<td>Thing has-a Thing Represents a subsystem of a system or a part of a physical object. Only things can have parts, which are other things.</td>
<td>Water heater has-a Pilot light Faucet has-a Washer</td>
</tr>
<tr>
<td><strong>Substeps</strong></td>
<td>Procedure substeps Procedure Represents the substeps of a procedural step, which would be a procedural steps that represent the same procedure in more detail. An experienced user may choose not to see this level of detail.</td>
<td>Replace washer substeps (Unscrew nut; remove washer; insert new washer; replace nut)</td>
</tr>
<tr>
<td><strong>Related-concept</strong></td>
<td>Links any node to any other node that may contain related or relevant information. Used to represent any relationships not specifically captured by existing link types.</td>
<td>Hot water system related-concept Heating system Install faucet related-concept install shower Washer related-concept Repair leaky faucet</td>
</tr>
<tr>
<td><strong>Problem-state</strong></td>
<td>Thing problem-state State Links things to the problem states that those things can be in. A problem state is an abnormal state which requires repair.</td>
<td>Pilot light problem-state Pilot light off Faucet problem-state Faucet leaky</td>
</tr>
<tr>
<td><strong>Repair-procedure</strong></td>
<td>State repair-procedure Procedure Links a problem state to a procedure that, if successfully completed, repairs the problem and returns the thing to its normal state.</td>
<td>Pilot light off repair-procedure Relight pilot light Faucet leaky repair-procedure Repair leaky faucet</td>
</tr>
<tr>
<td><strong>Outcome</strong></td>
<td>Procedure outcome State Links a procedure, which could be an entire procedure or an individual primitive step within a procedure, to the states that result from successfully carrying out that procedure. The state may be presented to the user as evidence that the procedure was successfully carried out. A procedure may have more than one possible outcome.</td>
<td>Tighten washer outcome Water does not leak through Test if water leaks outcome (Yes, No)</td>
</tr>
<tr>
<td><strong>Outcome</strong></td>
<td>State outcome State When there is no intentional intervening action, an outcome link may link a state directly to another state that may result.</td>
<td>Faucet leaky outcome Basement wet</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>State precondition Procedure Links the preconditions or enabling conditions of an action to the node that represents that action.</td>
<td>Pilot light off precondition Open water heater access panel Gasket sealed precondition Tighten bolts securely</td>
</tr>
<tr>
<td>Precondition Procedure precondition Procedure</td>
<td>Shut off water precondition unscrew top of faucet precondition remove washer</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>When the intervening state is not worth representing, a precondition link may link a procedural step directly to the next procedural step that follows.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Changes in Overall Plan and Personnel**

We replaced one graduate student (Mike Byrne; who graduated) with another graduate student (John Akers) to help with the ChemLab project and hired an additional graduate student (Viren Shah) to help implement the home repair architecture.

**Experiments**

*Chemistry*

The general description of the ChemLab software remains essentially the same as it was in the previous report. The new feature in the most recent experiment was to create a more systematic relationship between features of problems in the training and test phases of the experiment in order to allow us to look for specific differences in post-test problem solving.

The participants in the ChemLab experiment were approximately 80 undergraduates who enrolled in an introductory Chemistry course at Emory university, who participated in the experiment for extra-credit. The software itself was run on Macintosh Centris personal computers and developed in Apple's HyperCard (version 2.3).

**Journal Publications**

None

**Papers Submitted for Publication**

None

**Papers in Refereed Conference Proceedings**


**Paper Presentations**


**Upcoming Research**

Since we have completed most of our storyboarding efforts, we can now begin the system development based on the architecture we have developed.

Once the system has been built we can begin to use it to test our hypotheses about factors affecting navigation and how navigation can affect learning as manifested in tasks such as troubleshooting.
Our project focuses on researching and defining architecture guidelines for developing multimedia systems to support human learning. We base these guidelines on principles from cognitive science. During the last six months our efforts have been focused on two fronts. The first has been the continued development of an environment for teaching students chemistry, specifically, Lewis Structures. The second has been the specification of a generalized architecture for cognitive multimedia learning environments that operate on a web-based system. We are interested in how presentation factors and orientation tasks affect learning.

November 16, 1996 - May 16, 1997

Important Findings and Problems Encountered

ChemLab

We conducted two new experiments using the ChemLab software. The first of the new experiments was conducted using students at Emory University. We made a number of usability changes to the ChemLab software that we had used in a prior experiment based on feedback from students.

In the prior experiment we had used two conditions: directed browsing and problem solving. In the directed browsing condition students were asked a series of fact questions concerning Lewis Structures (e.g., How many valence electrons does Sulphur have? What is a lone pair? What is the most electronegative element? What is the chemical formula for Hydrazine?) and could navigate ChemLab to find answers. In the problem solving condition students were asked to solve a few Lewis Structure problems, that is, to determine the distribution of electron pairs and bonds for a particular molecule. After exploring ChemLab in one of these two ways, students were given a post-test (they were warned ahead of time they would be getting a post-test after interacting with the software). We did not find any difference in post-test performance between the two groups. We did, however, find differences in their browsing patterns in ChemLab suggesting that the orienting tasks (answering questions or solving problems) did influence how they interacted with
the software. One concern though was that we did not have a control group that could serve as a baseline to compare the performance of the other two groups.

In the first of the new experiments we had four conditions: directed browsing, problem solving, paper, and no ChemLab (N = 124). The first two conditions were the same as in the prior study (albeit with an improved interface). The paper condition involved students receiving a print-out of the Hypercard stack for ChemLab and being asked to study it in preparation for the post-test. Finally, students from the class who did not interact with ChemLab served as a "true" baseline. All students from the class took a final exam as part of their regular chemistry course. We were given permission to obtain final exam scores as well as copies of the final exams (coded by experimentally-assigned numbers). This allowed us to examine in detail performance on final exam questions dealing specifically with Lewis Structure problems.

The results from this study suggest that the ChemLab software did not help performance. However, once again their may be an interface-design explanation for this. First, consider the results. Performance on the post-test and the final exam by the different groups was as follows:

<table>
<thead>
<tr>
<th></th>
<th>Paper</th>
<th>Directed Browsing</th>
<th>Problem Solving</th>
<th>No ChemLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. No. of Post-Test Fact Questions Cor. (max = 7)</td>
<td>5.9</td>
<td>5.3</td>
<td>4.6</td>
<td>N/A</td>
</tr>
<tr>
<td>Avg. No. of Post-Test Problems Cor. (max = 4)</td>
<td>1.8</td>
<td>0.8</td>
<td>1.0</td>
<td>N/A</td>
</tr>
<tr>
<td>Avg. No. of Lewis-Structure Related Final Exam Problems Cor. (max = 4)</td>
<td>3.0</td>
<td>3.5</td>
<td>3.0</td>
<td>3.1</td>
</tr>
</tbody>
</table>

The performance differences on the post-test are statistically significant and suggest that the paper group was generally outperforming the other two on the post-test. The differences among the groups on the final exam were not significant although there was a trend favoring the directed browsing group.

Post-experiment interviews with a subset of students who used ChemLab suggested that one problem with the instructions in the software was that they did not make it clear that once students finished answering the orienting questions or problems the program would terminate and they would then go directly to the post-test. As a result, many of the students who interacted with the ChemLab software (as opposed to the paper version) spent relatively little time with the material. This may have accounted for the superior performance of the paper group on the post-test.

Performance on the final exam (which occurred several weeks after the experiment) suggests that the use of ChemLab might provide better retention of material as indicated by the trend towards better performance by the directed browsing condition.

In the second of the new experiments we altered the instructions to the ChemLab software to make it clearer to students that once they were done answering questions the ChemLab software would terminate. We ran this experiment at Georgia Tech partly in order to see if overall performance would be similar to our Emory sample. We were able to get only about seven participants per condition unfortunately and were not allowed access to final exams. While overall performance was higher than the Emory sample, the relative performance among the groups was about the same with the paper group showing some advantages over the other groups:
We have identified several changes we would like to make to the ChemLab software to better support navigating the system. These changes should also avoid the problem of students unknowingly finishing the program prematurely and having the program terminate before they have finished looking at all of the information.

One area of change will involve the introduction. Currently the introduction is approximately 10 screens of information, much of it involving how to use the system itself. This may have caused many students to become bored and inattentive before beginning the main part of the program. It is expected that by reworking some other aspects of the ChemLab interface, and by revising and reordering the introduction, the introduction can be shortened to 4-6 screens of information, most of it involving necessary background information about bonds and bond formation. In addition, the program will not terminate when learners are done answering questions/problems. Rather, it will terminate when learners indicate they are done exploring. Finally, greater emphasis will be placed on what the overall activity flow will be like, namely moving between the problem/question screen, the information screen, and the molecule construction kit.

Several other proposed changes to the ChemLab interface itself involve the method of navigation and the representation of the visual affordances for navigation. For instance, the full flow chart map of the entire sequence of steps will be made "clickable," so that students may click on the box for a particular step and be taken directly to the information for that step (currently the full map is not clickable - it only shows a full view of the steps to be taken). This will provide a more direct method of navigating from a global view of the overall procedure to the information details for a particular step in the procedure. Another change will be to make the location of the current step on the information screen more stable and informative. Currently the position of the boxes representing the previous, current, and next steps on the information window move around, and thus do not provide any information about whether they are primary steps, primary sub-steps, or secondary sub-steps. We believe that this may have made it more difficult to understand how the current step fit into the overall procedure. Also, the "Back" and "Next" buttons will be removed, and arrows pointing up and down in the navigation area of the information screen will be added that will scroll the steps displayed in that area up and down. Students will have to click directly on a step to display the information for that step (removing some of the ambiguity of "Back" and "Next"). The "Home" button will be reworded to say "Back to Step 1" to be more direct. The information radio buttons (definition, example, worked problem, and problem set) will be moved toward the top of the window to emphasize them more. The "Why?" button will be moved over to the right side of the window with the actual information content and away from the navigation buttons. Finally, the remaining navigation buttons (Full Map, Work, MCK, Back to Step 1) will be consolidated and placed in between the radio buttons and the clickable mini-navigation map.

Overall this reorganization should make the intended flow and actions more intuitive and easier to perform for the user, and clearly distinguish between the navigation and information functions of the various window elements.
Creating a Generalized Architecture for Cognitive Multimedia Learning Environments

Based on our experiences with ChemLab and Algonet, we are continuing to work toward a generalizable architecture for cognitive multimedia learning environments. We noted that our efforts to test alternative navigation and media selections were hampered by the complexity of building testbeds. While still exploring these issues in ChemLab, we have chosen to develop a generalized architecture in which we could easily change navigation, content, and even domain.

As reported in past reports, we have developed a knowledge structure based on cognitive multimedia theory that we hypothesize can be used to support students in learning how to conduct procedures (from repair to troubleshooting to installation) in any domain, using a wide variety of approaches (from direct instruction to scaffolded apprenticeship, from menu-based navigation to virtual reality, and from text-only to full-video multimedia content). Our first step toward testing our generalizability hypothesis is to construct a simple educational application structured with this architecture. The example domain that we have chosen is “plumbing,” where our content includes how to clear a clogged drain, how to install a toilet, and how the hot water system works at an abstract level.

We implemented the core architecture (in Java) that would display content based on student experience and a set of display rules (in a production rule format); basically, the core is a cognitive multimedia display engine. The idea is that content would be provided to the display engine along with a set of rules for how to present this content and a memory of information that the user had already provided, e.g., “If the user has identified the problem as a leaky faucet, present the known causes for a leaky faucet along with links to the procedures for testing for each of those causes.”

Next, we needed content and a sample set of rules. We developed a simple database application (in Apple Hypercard) for constructing knowledge elements to represent this content. However, after constructing several pieces of the knowledge-base, we found that the database application was clumsy to use and difficult to express complex content in. Basically, our problem was that we were constructing knowledge elements that were relating to other elements in a variety of ways, and that was difficult to represent in a simple user interface.

Therefore, we decided to design and develop a markup language which would allow us to represent linkages between knowledge elements at a symbolic level. We are referring to this language as PML (Procedural Markup Language). We have successfully:

- Defined PML (using the SGML standard DTD [document type definition] structure; see below),
- Written a simple PML to HTML translator to provide a means for reviewing the knowledge structures,
- Translated our database content into PML, and
- Translated our PML content into HTML for review.

All of these components are available for inspection at:
http://www.cc.gatech.edu/gvu/cognitive/cognitive-multimedia/home.html

In parallel, we developed a first set of rules for how to present instruction from our knowledge-base. These rules are used in the current PML-to-HTML translator, but in a static form — i.e., the static HTML cannot remember previous student selections. Our next step is to use the PML content to populate our cognitive multimedia engine and to encode our rules as production rules usable by the multimedia engine. At that point, we will have completed a proof-of-concept of the
generalizable system and can move on to conduct further experiments on navigation and media choices and push on to explore other domains.

<!DOCTYPE PML.dtd [ 

This is the document type description for a language we're calling PML. First we'll describe the elements in the language.

<!ELEMENT PML - - (thing | state | procedure)+ >

A PML document consists of "thing"s, "state"s and "procedure"s. There may be any number of each of them and they can occur in any order but there must be at least one element in the document. A PML document may not be empty.

<!ELEMENT (thing | state | procedure) - - (name+, author+, (description+ & justification? & relations? & example* & counter* ))) >

Things, states, and procedures consist of the following: at least one name followed by at least one author. Then the following may occur in any order: at least one description, only one justification, only one set of relations, any number of examples, any number of counterexamples. "Any number" includes zero. When something occurs more than once, the occurrences must be "grouped" together - you can't have examples mingled with descriptions, for instance. Procedures may no longer be nested.

<!ELEMENT (name | author | description | justification | example | counter) - - (#PCDATA) >

Names, authors, descriptions, etc. may occur inline and are composed of character data only -- they may not include other tags defined in PML.

<!ELEMENT relations - - (link+) >

Relations consists of a set of links. There must be at least one.

<!ELEMENT link - o (target+)>

A link consists of a set of targets. There must be at least one. Link does not require an end tag because it can only be followed by another link or the end of the relations tag.

<!ELEMENT target - o EMPTY>

A target is an empty tag and therefore does not require an end tag. An attribute (defined below) will be used to refer to the element we're using as the target. Doing it this way allows the parser to flag when we're referring to an element that doesn't exist. Here's what a sample of the relations part would look like:

<RELATIONS>
<LINK type=steps>
#The ordering here is important and implies Next & Previous links
<TARGET tid="Project 2, Step 1"> #These are the IDs of other procedures
<TARGET tid="Project 2, Step 2">
<TARGET tid="Project 2, Step 3">
<TARGET tid="Project 2, Step 4">

5
Now we define the attributes for the various elements.

<!ATTLIST (thing | state | procedure)
  id  ID  #REQUIRED>

Things, states and procedures are required to have an ID which will be unique across the database.

<!ATTLIST (author | description | justification | example | counter)
  src  CDATA  #IMPLIED>

The tags listed may have an optional "src" attribute to allow these tags to refer to external resources. These will generally be filenames, but can be any character data.

<!ATTLIST procedure
difficulty  CDATA  #IMPLIED>

Procedures may have a difficulty rating. We're not going to define what those ratings are, though.

<!ATTLIST description
detail  CDATA  #IMPLIED>

Descriptions may have a detail rating. Again, we'll leave the actual values undefined.

<!ATTLIST link
type  (uses | isa | hasa | connectsto | relatedto | example | counterexample | steps | pre | post | problem | repair)  #REQUIRED>

Links must have a type specified. The valid types are listed above. We could open this up so that new types may be specified as well.

<!ATTLIST target
tid  IDREF  #REQUIRED>

Targets are required to have a tid attribute (target-ID). The tid should be an ID which is elsewhere in the system.

] >
Changes in Overall Plan and Personnel

We replaced one paid graduate student (Viren Shah) with two unpaid grad students (Teresa Hubscher-Younger, Coleen Kehoe) who are interested in the project because it is related to their thesis work.

Experiments

Described above.

Journal Publications

None

Papers Submitted for Publication

None

Papers in Refereed Conference Proceedings

None

Paper Presentations

None

Upcoming Research

We will be conducting a new ChemLab experiment with the redesigned interface and improved instructions. Besides differences in browsing patterns and post-test performance, we will also be interested in observing whether the redesigned ChemLab software helps students with long-term retention as measured by final exam performance.

We will continue to develop PML and apply it to the home repair in order to come up with a web-based system that we can test.
Our project focuses on researching and defining architecture guidelines for developing multimedia systems to support human learning. We base these guidelines on principles from cognitive science. During the last six months our efforts have been focused on two fronts. The first has been the continued development of an environment for teaching students chemistry, specifically, Lewis Structures (we have conducted another experiment described below). The second has been the development of a generalized architecture for cognitive multimedia learning environments that operate on a web-based system. Our efforts on this topic during this six-month period have centered on: 1) the creation of a tool for allowing content-designers to connect the various pieces of knowledge in a knowledge base; 2) the continued development of modules in home repair (plumbing).

May 15, 1997 - November 14, 1997

Important Findings and Problems Encountered

ChemLab

We conducted a new experiment using a revised version of the ChemLab software. The revision was based on user-feedback and our own analyses of the interface. Specifically we addressed the following issues:

1) We shortened the introductory screens from approximately 10 screens of information to four. In prior experiments the long introductory information may have caused many students to become bored and inattentive before beginning the main part of the program. This reduction was accomplished largely by reworking the necessary background information about bonds and bond formation. The focus became tighter and provided learners with primarily the most-needed information. We believed such an approach would make the software more usable and therefore improve learning. Users could always consult other sources for related information not directly on the path to understanding the solving Lewis Structure problems.
2) Usability issues:

a) We improved the overall activity flow, that is, we improved the process of moving between the problem/question screen, the information screen, and the Molecule Construction Kit.

b) We improved the navigation and the representation of the visual affordances for navigation. For instance, the full flow chart map of the entire sequence of steps for doing Lewis Structures was made "clickable" so that students could click on the box for a particular step and be taken directly to the information for that step. This provides a more direct method of navigating from a global view of the overall procedure to the information details for a particular step in the procedure.

c) We made the location of the current step on the information screen more stable and informative. Previously the position of the boxes representing the previous, current, and next steps on the information window moved around, and thus did not provide any information about whether they were primary steps, primary sub-steps, or secondary sub-steps. We believe that this may have made it more difficult to understand how the current step fit into the overall procedure.

In a prior experiment we had three conditions that overlapped with the present experiment: directed browsing, problem solving, and paper. The paper condition involved students receiving a print-out of the Hypercard stack for ChemLab and being asked to study it in preparation for the post-test. The results from that study suggest that the ChemLab software did not help performance. However, we thought the interface was part of the problem and we redesigned it for the present experiment. The present experiment added an undirected browsing condition (participants were asked to explore the software without being given any orienting questions although they were told, like all participants, that there would be a post-test). Here are the primary results:

<table>
<thead>
<tr>
<th></th>
<th>Directed Browsing (n=17)</th>
<th>UnDirected Browsing (n=17)</th>
<th>Problem Solving (n=15)</th>
<th>Paper (n=18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. No. of Post-Test Fact Questions Cor. (max = 7)</td>
<td>5.2</td>
<td>5.5</td>
<td>5.1</td>
<td>5.2</td>
</tr>
<tr>
<td>Avg. No. of Post-Test Problems Cor. (max = 4)</td>
<td>0.65</td>
<td>1.3</td>
<td>1.1</td>
<td>1.4</td>
</tr>
</tbody>
</table>

The performance differences among the groups on the post-test were not statistically significant. The three ChemLab groups did differ in how much time they spent in the various areas of the ChemLab software. For instance, the Problem Solving group was the only one to actually visit and use the Molecule Construction Kit (MCK) part of the software. This is not surprising since this was the only group that had to solve training problems. Nevertheless, this did not translate into superior post-test problem solving performance. In fact, no clear connections were found between post-test performance and the time participants spent in various areas of the software.

Creating a Generalized Architecture for Cognitive Multimedia Learning Environments

The PIs created prototype modules in home repair to allow us to test our ideas about cognitive media types. This helped us work on the development of a Procedural Markup Language (PML) and to think about the translation/generation phase (see below). We have continued to refine our ideas about the types of knowledge nodes and links that define a knowledge base. However, most of our effort lately has been to develop a tool for constructing knowledge bases.
Creating a Tool for Developing a Knowledge Base

In order to facilitate the authoring of PML documents, we have developed a graphical editing tool called tkPML which can be used to create the knowledge node/link networks graphically (one of our grad students, Scott McCrickard, has been the primary developer of the software). The tkPML graph creation interface replaces the tedious, mistake-prone task of entering the node and edge information by hand with a point-and-click interface with form fill-in for the text entry. We expect that this tool will help designers to better establish and maintain mental models of their PML representations.

In tkPML, nodes and links can be created and positioned with simple mouse actions, and the layout can be seen at various levels of detail to obtain an overview of large graphs as well as a more informative view of a smaller number of nodes. Each node can be expanded to view and change the knowledge contained in it. Below is a screen shot showing various nodes being linked.

tkPML files are saved in an augmented PML format, with node locations, which is not part of the standard PML definition, saved as comments. Thus, designers can edit the saved files by hand or run presentation interpreters on the files without modification. In addition, tkPML can import files written in PML, even files which were not created using tkPML (the node locations are generated automatically). This allows designers to familiarize themselves with PML representations without wading through lots of code.
As the name suggests, tkPML is written in Tcl/Tk, a platform-independent graphical scripting language that can run on Unix machines, PCs, Macintosches, and within browsers over the World Wide Web. This allows PML representations to be exchanged and edited by users on different platforms and even published on the Web.

Nodes are created by double clicking on the background and are moved by dragging on the top "title" portion of the node. Links are created by dragging from the bottom "link" portion of one node to another. Double clicking on an existing node pops up a node information screen which allows a designer to edit information about the node.

For cognitive media information, users can edit the ID, title, author, and type. The "Edit..." button pops up the media window (shown in the middle) and the "Edit appspecific" button pops up the application-specific window. The media window allows a user to add media nodes within a knowledge node. A media node can be selected in the listbox and edited in the display area. The media nodes can be textual descriptions or names of files containing multimedia information. The application-specific window is used to enter application-specific information, that is, information that is not in the PML specification but is important for this PML instance.

Translating the Knowledge Base into HTML

In order to create a presentation based on a PML document, we are developing a PML interpreter which can interpret PML descriptions, retrieve the appropriate media in those descriptions, and create a hyperlinked presentation based on the situation and needs of the user (this effort is being headed by Coleen Kehoe, an unpaid graduate student involved in the project). For example, if a novice user is considering whether to call a plumber to repair a leaky shower, the system need not display all the details of the repair procedure but instead may choose to summarize the time, expertise, and tools necessary to perform the procedure. If the same user has previously repaired a leaky faucet, the system may display an overview of the repair procedure (the top-level steps) and provide links to the sub-substeps that are different from the previous procedure that the user has experience with. Truly interactive and dynamic multimedia systems will need such capabilities, and PML is designed to support the development of such systems.

Alternatively and more simply, one may develop a PML interpreter which can create a small number of predetermined presentations (such as one that always displays substeps in detail and another which always displays substeps as titles with links to the details) and use a simple heuristic to decide which presentation to use. We have chosen this approach for our initial implementation in order to test PML representations. Specifically, we have developed, and are now refining, a simple PML-to-HTML translator which can create presentations based on simple, predetermined presentation rules. This allows us to experiment with PML and can later be replaced with a fully dynamic presentation system.

Changes in Overall Plan and Personnel

We temporarily hired one paid graduate student (Scott McCrickard) to help develop the tkPML software. He continues to work on the software but is now supported by funding from another source.

Experiments

Described above.

Journal Publications
Upcoming Research

We will be developing two knowledge bases: one in home repair and the other in cooking (specifically, sushi). We are also working on a paper describing our knowledge structure and how it is used. Once the knowledge bases are more complete, we plan to begin conducting experiments on how users interact with the knowledge bases. We are interested in issues such as how the information should be presented as a function of the user's expertise and past interactions with the software (this will involve continued development of the presentation software which takes the PML as input and then has to decide how to display it). With respect to the software itself (the knowledge base design tool, and the PML-to-HTML translator), we need to conduct usability studies to make sure the tools will be helpful to designers.
FINAL REPORT

Using Cognitive Principles to Design Multimedia Training Environments to Support Learning

John Stasko (PI)*
Richard Catrambone (Co-PI)**
Mark Guzdial (Co-PI)*
Ashwin Ram (Co-PI)*

November 19, 1998

*College of Computing
**School of Psychology

Georgia Institute of Technology
Atlanta, GA 30332
USA

Reproduction in whole or part is permitted for any purpose of the United States Government

This research was sponsored by the Cognitive and Neural Science & Technology Division, Office of Naval Research, under Grant No. N00014-95-1-0790.

Approved for public release; distribution unlimited
As implied in the table of contents below, shall we attach copies of the ICLS conference papers and a copy of the PML paper to this final report?

Table of Contents

Project Summary .................................................................................................................. 1

Project Publications and Reports ....................................................................................... 6

The role of student tasks in accessing cognitive media types. ......................... 6

Exploring interface options in multimedia educational environments ........... 21

PML: Representing procedural domains for multimedia presentations ......... 27
Project Summary

(Summary of research carried out under ONR Grant No. N00014-95-1-0790.)

This document serves as the final technical report of ONR Grant No. N00014-95-1-0790.

During this project we examined factors that influence how well people learn from multimedia systems. Our project focused on developing multimedia systems to support human learning, based on cognitive principles and guidelines from cognitive science. The question was not so much whether multimedia makes a difference, but rather how can it best be deployed to make a difference? Specifically, what combinations of media and methods of interaction are most effective for learning, and why?

Multimedia modules are frequently organized along physical media (e.g., text, graphics, video). Our work has been based on another way of thinking about the problem of how to organize the content. Thus, a learner at a given moment may want to see an example as opposed to a definition. The example might be text, graphics, video, etc. Thus, the physical media (text, graphics, etc) can be largely divorced from the “cognitive” media (example, definition, etc). Finally, our work is also significant in that it provides the beginnings of an adaptable multimedia presentation system that can adjust its content according to the learning needs of the user.

In domains such as chemistry (ChemLab) and programming (AlgoNet2) we manipulated features of the learning environment in order to examine whether certain aspects of the training environment—such as whether the learner is given the opportunity to actively construct things and how the learner is able to navigate through the environment—affect learning. As we progressed in these endeavors we came to believe that a major determinant for the creation of a successful multimedia learning environment would be the availability of tools that allow the content expert (e.g., the chemist, physicist, chef, or carpenter) to describe the relevant knowledge in a domain and to indicate relationships among bits of knowledge in that domain in a relatively simple way. Thus, we came to develop a Procedural Markup Language (PML) for knowledge specification. PML provides a way to lay out knowledge without the concern about how the knowledge would actually be presented in a multimedia training system. Other modules for presentation could be created to deal with presentation. PML was designed to make the knowledge specification task as direct as possible. Our assumption is that unless the knowledge base is specified in a clear and complete way (relatively speaking), any attempt to make the actual delivery of that information effective is likely to fail. So, we have focused towards the end of the project on making PML as straightforward and robust as possible.

The project has yielded two proceedings papers and an under-review journal paper that are listed following this summary and are also attached.

Empirical Studies

We conducted several studies to examine the effects of cognitive media types and self-explanations on learning in multimedia systems. One experiment was in the domain of chemistry (determining molecular shapes) and the other was in the domain of computer graph theory. The major findings were that the type of learning orientation induced in students when interacting with different versions of the systems leads to different browsing (through the system) patterns but does not appear to lead to differences in test performance. The difference in browsing patterns has implications for the types of “orienting” tasks one might want to give learners as they learn a new domain. Clearly some tasks engage the learner in the material more deeply while other tasks encourage the learner to skim much of the material in order to get to the “important” parts.

We have been developing a new training multimedia system in home repair (specifically,
plumbing). This new system builds on our initial findings for training in chemistry and graph theory as well as our notions about factors affecting navigation. We are attempting to create a system that allows learners to access information in multiple ways. For instance, a person might wish to know how water enters the home plumbing system, or how to repair a leaky faucet, or how to replace a washer. These questions reflect different goals and possibly background knowledge of the learner and thus, a training or help system should reflect these different goals by helping the learner traverse different paths through the system tailored to the goal. In addition to the system taking into account the learner's/user's goals, there is also the issue of how to best interact with the system. Queries could be based on a graphical interface that provides a schematic of a home that the user can systematically zero-in on in order to let the system know the components of interest. Alternatively the interface could be menu driven. Another option is a quasi-natural language interface. Our future work may explore these different interaction possibilities as well as examine the effects of different user goals on how the system is used and on how effectively the learner acquires/finds information. Finally, the system will provide information in various media (text, graphics, animations, videos) and although initially we will hold these media constant across learners (while we focus on navigation issues) we will ultimately also manipulate the media and examine the effects on learning.

Creating a Generalized Architecture for Cognitive Multimedia Learning Environments

Our other major thrust has been to continue to work toward a generalizable architecture for cognitive multimedia learning environments. We have developed a knowledge structure based on cognitive multimedia theory that we hypothesize can be used to support students in learning how to conduct procedures (from repair to troubleshooting to installation) in any domain, using a wide variety of approaches (from direct instruction to scaffolded apprenticeship, from menu-based navigation to virtual reality, and from text-only to full-video multimedia content). Our first step toward testing our generalizability hypothesis has been to construct a simple educational application structured with this architecture. The example domains that we have chosen are plumbing and cooking. The content, for plumbing, includes how to clear a clogged drain, how to install a toilet, and how the hot water system works at an abstract level.

Procedural Markup Language (PML)

We decided to design and develop a markup language which would allow us to represent linkages between knowledge elements at a symbolic level. We are referring to this language as PML (Procedural Markup Language). PML is composed of knowledge nodes connected by links and includes cognitive multimedia slots. Knowledge nodes are of three types: things, states, and procedures. The links between nodes are typed as well. More information on PML can be found in the attached under-review paper "PML: Representing procedural domains for multimedia presentations."

PML (and the tkpml tool for specifying the PML graphically; see below) provides a way for content developers to specify the knowledge in a domain and to specify the links among bits of knowledge. The resulting knowledge representation, including the cognitive and physical media, can then be used to develop actual HTML pages. Thus, developers will have a system for creating content and specifying relations in a way that can be used to create flexible web pages.

tkPML: Creating a Tool for Developing a Knowledge Base

In order to facilitate the authoring of PML documents, we have developed a graphical editing tool called tkPML which can be used to create the knowledge node/link networks graphically (one of our grad students, Scott McCrickard, has been the primary developer of the software). The tkPML graph creation interface replaces the tedious, mistake-prone task of entering the node and edge information by hand with a point-and-click interface with form fill-in for the text entry. We expect
that this tool will help designers to better establish and maintain mental models of their PML representations.

In tkPML, nodes and links can be created and positioned with simple mouse actions, and the layout can be seen at various levels of detail to obtain an overview of large graphs as well as a more informative view of a smaller number of nodes. Each node can be expanded to view and change the knowledge contained in it. Below is a screen shot showing various nodes being linked.

tkPML files are saved in an augmented PML format, with node locations, which is not part of the standard PML definition, saved as comments. Thus, designers can edit the saved files by hand or run presentation interpreters on the files without modification. In addition, tkPML can import files written in PML, even files which were not created using tkPML (the node locations are generated automatically). This allows designers to familiarize themselves with PML representations without wading through lots of code.

Nodes are created by double clicking on the background and are moved by dragging on the top "title" portion of the node. Links are created by dragging from the bottom "link" portion of one node to another. Double clicking on an existing node pops up a node information screen which allows a designer to edit information about the node.

For cognitive media information, users can edit the ID, title, author, and type. The "Edit..." button pops up the media window (shown in the middle) and the "Edit appspecific" button pops up the application-specific window. The media window allows a user to add media nodes within a
knowledge node. A media node can be selected in the listbox and edited in the display area. The media nodes can be textual descriptions or names of files containing multimedia information. The application-specific window is used to enter application-specific information, that is, information that is not in the PML specification but is important for this PML instance.

Translating the Knowledge Base into HTML

In order to create a presentation based on a PML document, we are developing a PML interpreter which can interpret PML descriptions, retrieve the appropriate media in those descriptions, and create a hyperlinked presentation based on the situation and needs of the user (this effort is being headed by Coleen Kehoe, an unpaid graduate student involved in the project). For example, if a novice user is considering whether to call a plumber to repair a leaky shower, the system need not display all the details of the repair procedure but instead may choose to summarize the time, expertise, and tools necessary to perform the procedure. If the same user has previously repaired a leaky faucet, the system may display an overview of the repair procedure (the top-level steps) and provide links to the sub-substeps that are different from the previous procedure that the user has experience with. Truly interactive and dynamic multimedia systems will need such capabilities, and PML is designed to support the development of such systems.

We implemented the core architecture (in Java) that would display content based on student experience and a set of display rules (in a production rule format); basically, the core is a cognitive multimedia display engine. The idea is that content would be provided to the display engine along with a set of rules for how to present this content and a memory of information that the user had already provided, e.g., “If the user has identified the problem as a leaky faucet, present the known causes for a leaky faucet along with links to the procedures for testing for each of those causes.” Next, we needed content and a sample set of rules.
Project Publications and Reports

Journal Publications

None.

Papers in Preparation for Journal Submission


Papers in Refereed Conference Proceedings


Paper Presentations


Web-Based Materials

Various components of the system are available for inspection at:

http://www.cc.gatech.edu/gvu/cognitive/cognitive-multimedia/home.html

tkpmi screen shots are in: http://www.cc.gatech.edu/grads/m/Scott.McCrickard/tkpmi/
tkpmi-main.gif shows the main tkPML screen displaying the toilet example
tkpmi-nodeinfo.gif shows the popup window for entering node information
tkpmi-editinfo.gif is the window for entering information for a single cognitive media instance
tkpmi-apppspec.gif is the window for entering application-specific information
PML: Representing Procedural Domains for Multimedia Presentations

Ashwin Ram, Richard Catrambone, Mark J. Guzdial, Colleen M. Kehoe, D. Scott McCrickard, John T. Stasko
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
ashwin@cc.gatech.edu


Abstract

The central issue in the development of multimedia systems is the presentation of the information to the user of the system and how to best represent that information to the designer of the system. Typically, the designers create a system in which content and presentation are inseparably linked: specific presentations and navigational aids are chosen for each piece of content and hard-coded into the system. We argue that the representation of content should be decoupled from the design of the presentation and navigational structure, both to facilitate modular system design and to permit the construction of dynamic multimedia systems that can determine appropriate presentations in a given situation on the fly. We propose a new markup language called PML (Procedural Markup Language) which allows the content to be represented in a flexible manner by specifying the knowledge structures, the underlying physical media, and the relationships between them using cognitive media roles. The PML description can then be translated into different presentations depending on such factors as the context, goals, presentation preferences, and expertise of the user.

Introduction

Suppose you are a homeowner faced with the problem of a leaky faucet. If you are reasonably comfortable with home repair, you might be able to just plunge on in and make the repair. But if you are not familiar with that kind of repair, you might seek help, perhaps using a book or even one of the new multimedia guides to home repair for your PC. Will the book present the information you need at the level you need it? Maybe you know one repair well in general, but don't know faucets. Then you probably want a book that presents the key steps, but without a lot of detail. But if you are a novice at home repair, you probably need a lot of examples (with photos and diagrams) and detailed descriptions. In the case of both the book and the multimedia guide, you may encounter the problem of reading the book or the PC screen while your hands have tools in them and you are in the midst of the repair.

This hypothetical scenario is the focus of our research. How can information be presented, dynamically, to meet the needs and prior knowledge of the learner? How should content be encoded so that the developer can present information tuned to the audience and perhaps even presented in the medium of choice?

One of the central issues in the development of multimedia systems, whether on the Web or as a standalone system, is the representation of the content, that is, the information that is to be displayed to the user of the system. For example, an educational system that teaches chemistry must contain information about atoms, molecules, and gas laws; a training system for computer technicians must contain information about memory chips and buses; and a Web site for amateur home repair must contain information about kitchens, showers, and faucets. In designing such systems, however, one must attend not only to the knowledge to be represented but to how that information is to be presented. Is the system's information about showers to be displayed in graphical form? Is information about installing memory chips to be displayed as an itemized list of textual operators? Is information about gas laws to be displayed using an animated video clip showing the movement of molecules as a gas is pressurized? Typically, the system designer must consider both content and presentation and, in doing so, create a system in which the two are separably linked; a specific presentation is chosen for each piece of information, perhaps combining several available media such as text, pictures, simulations, and sound, and hard-coded into the system so that it is available for presentation in exactly the form that the system designer intended. Specific navigational aids are also chosen and hard-coded into the system so that a predetermined hypermedia structure is encoded and available to the user.

We argue that knowledge representation and presentation design should be treated as separate activities. The knowledge engineer or content designer should focus on the knowledge that is being represented: what is known about molecules, buses, and faucets, how this knowledge is structured, and how it might be represented using basic media elements. The presentation designer should focus on the multimedia presentation of is knowledge: how should a diagnostic procedure be displayed to the user? At what level of detail, and for what level of expertise should it be presented? What should it be linked to? What navigational aids should be provided? What issues concerning how people comprehend text, graphics, simulations, etc. need to be considered?

Such an approach has several benefits. First, it is easier for domain experts (who may not be presentation experts) to build the knowledge presentation without regard to how the information will ultimately be displayed. Second, it is possible to design different presentations for the information based on the user's level of expertise, or on the task the user is engaged in (for example, learning vs. troubleshooting), or on other factors. Third, this approach permits the development of truly interactive multimedia systems in which the system creates appropriate presentations on-the-fly based on the current interactions and context. Only the knowledge needs to be specified beforehand, but whether a diagnostic procedure, for example, is presented as an itemized list of textual bullets, a graphical flow chart, an animated movie, or some combination thereof can be determined dynamically. If knowledge and presentation were tightly coupled, all these presentations would have to be created manually in advance and stored as alternative depictions of the same information.

Of course, the knowledge representation must ultimately bottom out in media: a textual definition of a gas law, a photograph of a faucet, or a schematic of a VLSI chip. Thus, it is important to provide a principled means of coupling the knowledge representation structures with the underlying media, but in a manner that provides the flexibility needed for interactive and dynamic presentations. We argue that knowledge structures should organize media according to their cognitive role [Recker, Ram, Shikano, Li, & Stasko, 1995]. Consider, for example, a student who is using an educational multimedia system to learn chemistry, or a homeowner who is using a home repair CD-ROM or Web site to help fix a leaky faucet in a bathroom. The user is unlikely to say, "I would like to see some text now" or "I could really use a WAV sound file now." Instead, the user may say, "I could really use an example," leaving it up to the system to determine whether that example is best presented as text, sound, simulation, or some combination thereof. In other words, we argue that multimedia content, consisting of physical media such as text, sound, video
ips, and so on, should be organized and coupled to knowledge structures using cognitive media roles, such as definition, example, simulation, solved problem, and so on. A cognitive media role, such as "example," specifies the function that the information plays in the cognitive processes of the user. The user might ask for an example of a faucet or a simulation of molecular forces, which in turn would be displayed using an appropriate combination of physical media as determined statically by the presentation designer and/or dynamically by the system itself.

There has been a significant amount of work attempting to disentangle knowledge representation from presentation in multimedia. Maybury [1993] emphasized this distinction in his work with intelligent multimedia interfaces. Feiner [Feiner, McKeown, 1991] has shown a system that can actually construct multimedia representations on-the-fly, drawing from a knowledge base of content and a separate knowledgebase about presentations. Research teams such as the Hyper-G group in Austria [Tomek, Maurer, & Nasser, 1993] have created Web-based applications that lower for separation between representation and presentation (e.g., keeping link information separate from the multimedia document itself). What we add to the existing science is (a) a theory of effective organization for multimedia used for learning (specifically, cognitive media roles), and (b) a notation for encoding knowledge such that an effective representation can be generated.

To facilitate the development of a system outlined above, we propose and describe in this article a new notation called Procedural Markup Language (PML). PML is a markup language written in XML that allows the content designer to encode domain knowledge in an intuitive and flexible manner by specifying the knowledge structures, the underlying physical media, and the relationship between them using cognitive media roles. We focus specifically on procedural task domains, in which the primary type of knowledge to be represented concerns the performance of procedures. The highlights of our formalism are:

- Information about a domain (e.g., plumbing) is encoded in knowledge nodes that have connections, called knowledge links, to other knowledge nodes.
- Information within a particular knowledge node (e.g., information about a faucet) is represented using physical media clusters containing media elements such as text, graphics, animations, video clips, and sound files.
- Physical media are organized under knowledge nodes using cognitive media roles, such as "definition," "example," etc. Any cognitive media role under a knowledge node could potentially contain one or more different physical media (e.g., an example of a faucet might be represented using some text and a graphic).
- The information contained in the combination of knowledge nodes, knowledge links, physical media clusters, and cognitive media roles forms the raw material that can be used by a presentation system to determine what the user or learner will see and hear, and what navigational connections and devices will be available on the screen. Different presentations may be created from the same underlying representation, depending on various factors such as the expertise of the user in the domain, the information that the user has previously seen, the current goal of the user, and so on.

In this article, we use the home repair domain as our example, and show how PML can be used to represent information about, for example, pairing a leaky faucet. The PML representation is independent of the particular presentation that is ultimately constructed. We show that the same ML representations can be used to create different presentations. Since we have focused primarily on the development of PML, our current implementation the presentation-construction system is fairly simple. Ultimately, we are interested in more sophisticated presentation-construction systems which can dynamically create an appropriate presentation based on the goals or tasks of the user, the user's level of expertise, the context, and other appropriate factors.

2. Technical Details: Knowledge Representation

ML consists of knowledge nodes that represent concepts and knowledge links that represent relationships between knowledge nodes, similar to the semantic net structures used for knowledge representation in artificial intelligence systems. However, unlike semantic net systems which are used for reasoning, nodes do not contain slot-filler representations of concepts; for multimedia presentations, nodes need to contain media that can be used to create presentations of those concepts for the user. Media are stored in physical media clusters that contain text, pictures, sounds, video clips, etc., that are the basic elements describing concepts. Media clusters are organized under knowledge nodes using cognitive media roles that represent the cognitive role (e.g., example, definition) played by the media elements in describing the concepts in the knowledge nodes. A cognitive media role, such as an example, may have one or more physical media clusters associated with it, where each cluster represents a different example. This structure is summarized in Figure 1.
Before describing the markup language itself, it is instructive to look at the representational structures that the language will need to encode. Let us briefly discuss the necessary nodes and links.

**Knowledge Nodes**

A knowledge node represents a concept that the system knows about. Information about the concept is represented using media clusters while relationships between concepts are represented using knowledge links. Following basic ontological principles that are commonly used in presentations systems in artificial intelligence and cognitive science, we divide the entities being represented into *things*, *states*, and *procedures* (see Table 1). Based on our experience with several different domains, this ontology appears to sufficient to capture the distinctions necessary to represent our target domains where the knowledge being represented is mainly procedural. In general, though, the robustness of the language can only be determined by representing a large number of domains using this formalism.

### Table 1: Knowledge Nodes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Thing</strong></td>
<td>Represents a system, physical object, part, or substance in its normal state. Things may be composed of other things.</td>
<td>Hot water system (system).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faucet (physical object).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Washer (part).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Water (substance).</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>Each thing has one or more states that it can be in. A thing’s normal state is the usual operational state of that thing; other states represent problem conditions that may need repair. Usually only problem states are represented explicitly; the main representation of a thing is assumed to represent its normal state.</td>
<td>Pilot light off.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faucet leaky.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Washer worn out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Water is brown.</td>
</tr>
<tr>
<td><strong>Procedure</strong></td>
<td>Represents a sequence of actions carried out by the user that operate on a thing in some manner. The actions that comprise a procedure may themselves be procedures; ultimately, this bottoms out when the “primitive” action is operationalized and can be directly carried out by the user.</td>
<td>Lighting a pilot lamp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replacing a leaky faucet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Installing a shower.</td>
</tr>
</tbody>
</table>

**Knowledge Links**

Knowledge nodes may be linked to other knowledge nodes using knowledge links that represent conceptual relationships between those knowledge nodes (see Table 2). Note that knowledge links are strongly typed; for example, the *precondition* link always connects states to procedures. The
Order in which multiple links of the same type are listed under any given node is not significant, with the exception of steps which are listed in the order in which they should be carried out. Knowledge links may be traversed in either direction by the system, although each link has an explicit source and destination endpoint. The reverse links are also listed in Table 2. These reverse links are managed by the system and not manually created by the user. Our formalism provides the following set of knowledge links; as before, determining the sufficiency of this set is an empirical question, although this set has been adequate for several domains that we have investigated.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is-a</td>
<td>Thing is-a Thing</td>
<td>Single-lever faucet is-a Faucet</td>
</tr>
<tr>
<td></td>
<td>Represents the broader category of a thing, or (the other direction) the particular types of a thing. The reverse link is subtype.</td>
<td>Faucet subtype single-lever faucet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pilot light is-a Ignition device</td>
</tr>
<tr>
<td>Is-a</td>
<td>Procedure is-a Procedure</td>
<td>Plunge-drain is-a unclog-drain</td>
</tr>
<tr>
<td></td>
<td>Represents the broader category of a procedure, or (the other direction) particular styles of a procedure. The reverse link is subtype.</td>
<td>Unclog-drain subtype plunge-drain</td>
</tr>
<tr>
<td>Has-a</td>
<td>Thing has-a Thing</td>
<td>Water heater has-a Pilot light</td>
</tr>
<tr>
<td></td>
<td>Represents a subsystem of a system or a part of a physical object. Only things can have parts, which are other things. The reverse link is part-of.</td>
<td>Pilot light part-of water heater</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faucet has-a Washer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hot water system has-a Shutoff valve</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hot water system has-a Stepped pipes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hot water system has-a Water heater</td>
</tr>
<tr>
<td>Connects-to</td>
<td>Thing connects-to Thing</td>
<td>Shutoff valve connects-to Stepped pipes</td>
</tr>
<tr>
<td></td>
<td>Represents contiguous or connecting pieces of an overall physical system. The overall physical system, represented as a thing, would have has-a links to the individual things comprising it as well. The reverse link is connects-to.</td>
<td>connects-to Water heater connects-to Hot water supply line</td>
</tr>
<tr>
<td>Steps</td>
<td>Procedure steps Procedure</td>
<td>Replace washer steps (Unscrew nut; Remove washer; Insert new washer; Replace nut)</td>
</tr>
<tr>
<td></td>
<td>Represents the substeps of a procedure, that is, steps that represent the procedure in more detail (these steps may, in turn, be further broken down into substeps). An experienced user may choose not to see this level of detail. There is an implied ordering of the steps of a procedure. The reverse link is step-of.</td>
<td>Unscrew nut step-of Replace washer</td>
</tr>
<tr>
<td>Problem-state</td>
<td>Thing problem-state State</td>
<td>Water heater problem-state Pilot light off</td>
</tr>
<tr>
<td></td>
<td>Links things to the problem states that those things can be in. A problem state is an abnormal state that requires repair. A thing may have multiple problem states. The reverse link is problem-state-of.</td>
<td>Pilot light off problem-state-of water heater</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faucet problem-state Faucet leaky</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faucet leaky</td>
</tr>
<tr>
<td>Repair-procedure</td>
<td>State repair-procedure Procedure</td>
<td>Pilot light off repair-procedure Relight pilot light</td>
</tr>
<tr>
<td></td>
<td>Links a problem state to a procedure that, if successfully completed, repairs the problem and returns the thing to its normal state. A problem state may have multiple repair procedures. An installation procedure is also represented as a repair procedure. The reverse link is repair-procedure-for.</td>
<td>Relight pilot light repair-procedure-for pilot light off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faucet leaky repair-procedure Repair leaky faucet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No bathtub in bathroom repair-procedure Install bathtub</td>
</tr>
<tr>
<td>Outcome</td>
<td>Procedure outcome</td>
<td>State outcome</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outcome</td>
<td>State outcome</td>
<td>State</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precondition</td>
<td>State precondition</td>
<td>Procedure</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uses</td>
<td>Procedure uses</td>
<td>Thing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Related-to</td>
<td>Links any node</td>
<td>to any other node that may contain related or relevant information. Used (sparingly!) to represent any relationships not specifically captured by existing link types. The reverse link is related-to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Physical Media Clusters**

Physical media cluster (or simply media cluster) contains the actual information about a knowledge node that the system can display to the user. We will see below that all media are stored in separate files, referenced via the MEDIA tag, with the exception of text which may be included in-line in the PML document for authoring convenience. A media cluster may contain more than one type of physical media (text, video, etc.). A knowledge node may contain one or more media clusters; these are organized using cognitive media roles that provide the connections between the knowledge structures and the media clusters.

**Cognitive Media Roles**

The media clusters within a knowledge node are organized in terms of the cognitive roles they play in the problem-solving task in which the user is engaged. For example, a particular mixed-media text-and-pictures description of a faucet may serve as an "example" of a single-lever faucet; in this case, the knowledge node for "single-lever faucet" will contain an "example" role which contains a media cluster that represents that text-and-pictures description. Our formalism provides the following cognitive media roles [Recker et al., 1995].
Table 3: Cognitive media roles

<table>
<thead>
<tr>
<th>Name/Title</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Title</td>
<td>The name of the item being represented in the knowledge node. Usually one or a few words of text.</td>
<td>&quot;Pilot light.&quot;</td>
</tr>
<tr>
<td>Definition/Description</td>
<td>The definition of the concept being represented in the knowledge node or, more informally, its description. Usually a textual description accompanied by diagrams.</td>
<td>&quot;A pilot light is a small gas flame that is continually burning. It is used to light the furnace when ... ...&quot;</td>
</tr>
<tr>
<td>Example</td>
<td>An example of the concept being represented in the knowledge node.</td>
<td>[Schematic of pilot light]</td>
</tr>
<tr>
<td>Counter-example</td>
<td>An example of an alternative design, an alternative state, etc.</td>
<td>[Photograph of a flint-based lighting system.]</td>
</tr>
<tr>
<td>Justification</td>
<td>An explanation of a step being carried out in a procedure, or an explanation of the functional role of a thing that is part of a larger thing.</td>
<td>&quot;You want to turn off the water at the supply first because ...&quot;</td>
</tr>
</tbody>
</table>

To illustrate how these nodes and links fit together to provide an overall representation of the domain of interest, consider a snippet of the presentation of faucets from the home repair domain, shown in Figure 2 in pictorial form.
6. Technical Details: Procedural Markup Language

The previous section described our knowledge representation framework. Briefly, knowledge is organized into nodes (concepts) that are linked by relationships. Each node organizes physical media into clusters corresponding to roles for the media. This was summarized in Figure 1. In this section, we describe the notation that we use for articulating this representation.

We have developed a language called PML that allows authors to encode our knowledge representation in a set of files. Authoring in PML is analogous to authoring in HTML or other markup languages, but with the crucial difference that the author focuses on representing information about the domain and not primarily on information about presentation. That is, the two types of information are decoupled. For example, if one were to create a Web page in HTML describing how to install the latest release of a piece of software, one might do this as follows:

```
To install this release, perform the following steps: <BR>
<OL>
<LI> Download the file.
<LI> Unstuff the file.
<LI> Double-click on the installer icon.
</OL>
```

Notice that the procedure is described using a series of steps, but in order to state the steps one has to choose a particular physical representation (here, a numbered list of items). Using PML, however, one would specify the steps independent of the presentation:

```
<PROCEDURE ID="install">
```

Figure 2: Pictorial form of a PML knowledge structure about faucets.
Installing the software

To install this release, perform the following steps:

**LINK**

**TARGET ID="download"**

**TARGET ID="unstuff"**

**TARGET ID="execute"**

This example presents a **procedure** knowledge node with two cognitive media roles: **title** and **description**. The knowledge node has **step** knowledge links to separate "download," "unstuff," and "execute" procedure nodes that are represented using PML as well. Note that the PML does not specify whether to display the three steps as a numbered list, a flow chart, or as three separate pages with navigational arrows between them; that decision can be made independently and, if desired, dynamically (limited only by the physical media provided). Note also that the PML presentation allows additional types of information to be represented, such as the preconditions and outcomes of the procedure, things that might go wrong and how to recover from them, justifications for the procedures, and so on. These could be hard-coded into the HTML representation too, but again the presentation would be static. Finally, the PML representation allows procedures and subprocedures to be represented hierarchically; the level of detail that is actually presented can be determined dynamically and should be dependent on factors such as the expertise of the user.

ML is written in Extensible Markup Language (XML) [Bray, 1998], a language for describing other markup languages. XML is a simplified version of Standard Generalized Markup Language (SGML) [Sperberg-McQueen, 1993], an international standard for creating structured documents. A markup language is essentially a set of tags that an author uses to describe parts of a document and a document that uses these tags is a kind of structured document. Currently, the most well-known markup language is HTML which contains tags like <TITLE>, <H1>, <IMG>, c. While these tags are useful for describing basic document structure, they do not describe the content of a document very well. More powerful and most likely domain-specific markup languages are needed for this purpose and XML was developed as a common way to define these different markup languages. XML, however, has utility far beyond the World Wide Web; it can be thought of as a platform-independent way to represent knowledge in a machine-readable format. XML has already been used to specify markup languages for dozens of applications ranging from chemistry to electronic commerce [Cover, 1998]. Having a common way to specify these markup languages allows tools to be built that can work with any of the languages specified in XML. For example, we have developed a PML-to-HTML translator that uses a PML parser. This parser is generic, however, understanding XML and therefore any markup language specified using XML.

The notation used in XML descriptions is fairly standard (see [Bray, 1998] for details). Each ELEMENT statement is a production rule with the first item being the left-hand side of the rule and the second item (in parentheses) being the right-hand side. Every element corresponds to a tag in the markup language. A vertical bar indicates a choice and a comma indicates a sequence. The plus sign stands for "one or more" and the asterisk stands for "zero or more." An ATTLIST statement lists the attributes for a particular element (i.e. tag). It specifies the type of the attribute and whether it is required (REQUIRED) or optional (IMPLIED). The complete specification of our PML language is given in Table 4. Appendix A contains an annotated example PML representation of a snippet of an everyday procedural domain: baking a cake.

Table 4: Specification of PML

```xml
<---- Things ---->

<!ELEMENT thing (title, (author | description | justification | link | appspecific | example | counterexample *))>
<!ATTLIST thing id ID #REQUIRED>

<---- States ---->

<!ELEMENT state (title, (author | description | justification | link | appspecific | example | counterexample *))>
<!ATTLIST state id ID #REQUIRED>

<---- Procedures ---->

<!ELEMENT procedure (title, (author | description | justification | link | appspecific | example | counterexample *))>
<!ATTLIST procedure id ID #REQUIRED>

<---- Cognitive Media Types & Identifying Information ---->

<!ELEMENT title (#PCDATA | media)*>
<!ELEMENT author (#PCDATA | media)*>
<!ELEMENT description (#PCDATA | media)*>
<!ELEMENT justification (#PCDATA | media)*>
<!ELEMENT example (#PCDATA | media)*>
<!ELEMENT counterexample (#PCDATA | media)*>
<!ATTLIST description type CDATA IMPLIED>
```
Authoring tools

In order to facilitate the authoring of PML documents, we have developed a graphical editing tool called tkPML that can be used to create the knowledge node/link networks graphically (see Figure 3). As might be expected, textual hand-authoring of PML structures can be tedious and mistake-prone. The tkPML graph creation interface replaces the task of entering the node and link information by hand with a point-and-click interface with form fill-in for the required text entry. We expect that this tool will help designers to better establish and maintain mental models of their PML representations.
Figure 3: A screenshot of a tkPML session. Nodes are created by double clicking on the background and are moved by dragging on the top "title" portion of the node. Links are created by dragging from the bottom "link" portion of one node to another. Double clicking on an existing node pops up a node information screen (shown in Figure 4) that allows a designer to edit information about the node.

tkPML, nodes and links can be created and positioned with simple mouse actions, and the layout can be seen at various levels of detail to obtain an overview of large knowledge structures as well as a more informative view of a smaller number of nodes. Each node can be expanded to view and change the knowledge contained in it (see Figure 4).
The tkPML tool saves a PML description file in an augmented PML format file. The one addition is simply node location, which is not part of the standard PML definition and is saved as a comment. Thus, designers can edit the saved files by hand or run presentation interpreters on the files without modification. In addition, tkPML can import files written in PML, even files that were not created using tkPML. For such files, tkPML uses simple graph layout algorithm to generate an initial display.

As the name suggests, tkPML is written in Tcl/Tk, a platform-independent graphical scripting language that can run on Unix machines, PCs, Macintoshes, and within browsers over the World Wide Web [Ousterhout, 1994]. This allows PML representations to be exchanged and edited by users on different platforms and even published on the Web.

Presentation tools

In order to create a presentation based on a PML document, one may develop a PML interpreter-generator that can interpret PML descriptions, retrieve the appropriate media in those descriptions, and create a hyperlinked presentation based on the situation and needs of the user. For example, a novice user is considering whether to call a plumber to repair a leaky shower, the system need not display all the details of the repair procedure but instead may choose to summarize the time, expertise, and tools necessary to perform the procedure. If the same user has previously repaired a leaky faucet, the system may display an overview of the repair procedure (the top-level steps) and provide links to the substeps that are different on the previous procedure with which the user has experience. Truly interactive and dynamic multimedia systems will need such capabilities, and PML is designed to support the development of such systems. The PML interpreter-generator would need to be based on principles of instructional design and interaction design. PML is designed to support experimentation with such principles.

Alternatively and more simply, one may develop a PML interpreter that can construct a small number of predetermined presentations (such as one that always displays substeps in detail and another which always displays substeps as titles with links to the details) and use a simple heuristic to decide which presentation to use. We have chosen this approach for our initial implementation in order to test PML representations. Specifically, we have developed a PML-to-HTML translator that can create presentations based on simple, predetermined presentation rules (see Figure 5). This allows us to experiment with PML and gain knowledge that will facilitate later development of a fully dynamic presentation system.

Both of the presentations in Figure 5 are designed to help the user unplug a toilet drain using an auger. The left presentation is aimed at a novice, and the right presentation is aimed at an expert. For the expert, the basic steps are presented with minimal explanation, but with links that lead to more information, if desired. For the novice, more introductory information is provided (not shown) such as what an auger is, and each step is expanded with its explanation and any examples available for the step. Both of these presentations assume a web browser on a personal computer as the target platform. If the target were, say, a handheld personal computer (which might be more useful when working in the bathroom) or even an audio presentation, a different structure should be generated (e.g., the handheld should not have a long scrolling presentation, as does the novice presentation depicted in Figure 5).
1. Conclusions

In this paper we present a new markup language called PML to facilitate the authoring of dynamic multimedia systems for procedural domains. We show how PML can be used to represent domain knowledge (concepts and relationships) independent of presentation issues and how this knowledge can be loosely coupled to presentation media via cognitive media roles. PML involves knowledge nodes connected by knowledge links. The knowledge nodes can contain cognitive media roles holding physical media clusters.

Cognitive media roles have been used successfully in educational multimedia systems for teaching graph algorithms in an undergraduate computer science course [Recker et al., 1995; Shikano, Recker, & Ram, 1998; Shippey, Ram, Albrecht, Roberts, Guzdial, Catrambone, Byrne, & Stasko, 1996], and Lewis structures in an undergraduate chemistry course [Byrne, Guzdial, Ram, Catrambone, Ram, Stasko, Shippey, & Albrecht, 1996]. ML has also been used for other tasks and domains; we are using it to represent cases of object-oriented design and programming [Guzdial 1997], and to encode process information about operations in an electronic assembly “Clean Room” [Realff et al., 1998]. While these earlier systems were of dynamic, they do illustrate the generality and value of the knowledge representation and the notational tools.

In our own research, we are developing PML-based systems to investigate cognitive issues relevant in the design of dynamic multimedia systems. More broadly, in a learning situation, the goals that students bring to the learning task will affect their learning processes [Ram & Leake, 1995] and therefore a hypermedia support system for learning should have the capability to adjust itself in response to the user's goals. PML allows us to examine these issues empirically. For example, we would like to conduct systematic experiments that look at what factors actually play a role in the effectiveness of different presentations to the user or learner. For instance, is it really the case that a “high-level” presentation of a procedure for a more knowledgeable person is more effective (measured, perhaps, in how well the person can do the procedure and how long it takes, counting presentation time) than providing him or her with all the details? In order to empirically answer questions such as these, we need a system capable of selecting alternative presentations of some underlying information, which is precisely the goal of PML.

Acknowledgements

This research was supported by the Office of Naval Research under contract N00014-95-1-0790. We thank Mimi Recker for her comments on an earlier draft of this paper.

References

- JS - MANY REFS FROM THE BODY ARE MISSING HERE.
Table 5: Example PML file

```xml
<PML>
  # All PML documents must start with the PML tag
  <PROCEDURE id="cake_1">
    # This says we're beginning a procedure. We've given it an id of "cake 1."
    # This is the name you use to refer to this procedure in other places.
    <TITLE>How to Bake a Cake</TITLE>
    <AUTHOR>Colleen Kehoe</AUTHOR>
    # We define the title of this procedure and the author. Title is required.
    # The title may be the same as the id in the procedure tag, if desired.
    <DESCRIPTION>
      This procedure tells you how to bake your basic cake. It assumes you're at or near sea level. You'll need a different procedure if you're at a high altitude.
      # We give a description of the overall procedure
      # Since it is text, we can include it here or reference an external file via a MEDIA tag.
    </DESCRIPTION>
    <JUSTIFICATION>
      Everybody likes cake.
    </JUSTIFICATION>
    # We give a justification for this procedure. This is optional.
    <APPSPECIFIC key="difficulty" value="easy"/>
    # Here we may associate some application-specific information with this node.
    # This may be used for indexing purposes or for deciding how to display this node.
    <EXAMPLE>
      <MEDIA SRC="cake.gif" CAPTION="Here is a picture of someone baking a cake."
      <MEDIA SRC="cake.mov" CAPTION="Here is a movie of a baker at work."
    </EXAMPLE>
    # An example containing two physical media files.
    # Any number of examples or counterexamples are allowed.
    # Now we list all of this links from this node to other nodes in the system.
    <LINK type="uses"> # This is a list of the equipment this procedure uses.
      <TARGET id="mixer"/> # This is a "thing" node.
    </LINK>
    <LINK type="problem-state"> # The following nodes are problems.
      <TARGET id="cake didn't rise"/> # Each is a "state" node.
      <TARGET id="cake burnt"/>
      <TARGET id="cake tastes salty"/>
    </LINK>
    <LINK type="outcome"> # The following node is an outcome.
      <TARGET id="cake is done"/> # This is a "state" node.
    </LINK>
  </PROCEDURE>
</PML>
```
This is a list of the steps in this procedure.

These are "procedure" nodes.

These are "procedure" nodes.

This marks the end of this procedure.

This is the beginning of a new procedure. Notice that this is the first step in the procedure we just defined above. Procedures are defined hierarchically.

Mix the Ingredients

Get all the ingredients together and mix them.

# For this application, we've provided two descriptions, both text.

You will need: 2 eggs, 2 c. flour, 1/2 c. milk, 3 tbsp. butter, 1 tsp. baking soda, 1/4 tsp. salt, 1/4 c. water, 1 c. sugar. Combine the dry ingredients in one bowl. Combine the wet ingredients in another bowl. Gradually add the dry to the wet, blending with an electric mixer.

# Here, we provide a very detailed description.

# As before, we list the links from this node to other nodes in the system.

This is a list of the equipment this procedure uses.

This is a "thing" node.

These are the rest of the steps in the "cake 1" procedure. Details are omitted in the interest of space, but they would be similar to the one above.

put in oven

This is the beginning of a new procedure. Notice that this is the first step in the procedure we just defined above. Procedures are defined hierarchically.

Put in oven

Test for doneness

Cool

Baked Good

A baked good is usually found in a bakery. They are things like: bread, cookies, cakes, muffins, etc.

# Here, we provide a counterexample to a baked good.

It is in an external media file, but we provide a textual caption as well.

Cake didn't rise properly

# This marks the end of the thing node.

Cake didn't rise properly

# Now we define a "state" node. This was one of the problem states referred to in the "cake 1" procedure.

Cake didn't rise properly
The cake didn't rise above the edge of the pan. This is usually caused by accidentally leaving out the baking powder or salt.

These are links to repair procedures.
These are "procedure" nodes.
This is a state.

Other procedures, things, and states are defined similarly.
Exploring Interface Options in Multimedia Educational Environments

Gordon Shippey
College Of Computing
shippey@cc.gatech.edu

Ashwin Ram
College Of Computing
ashwin@cc.gatech.edu

Florian Albrecht
College Of Computing
florian@cc.gatech.edu

Janis Roberts
College Of Computing
janis@cc.gatech.edu

Mark Guzdial
College Of Computing
guzdial@cc.gatech.edu

Richard Catrambone
School Of Psychology
rc7@prism.gatech.edu

Michael Byrne
School Of Psychology
byrne@cc.gatech.edu

John Stasko
College Of Computing
stasko@cc.gatech.edu

Georgia Institute of Technology
Atlanta, Georgia 30332, USA

Introduction

Multimedia technology allows information to be organized and represented in a wide variety of ways. For a computer-based educational environment to be effective, the subject matter must be presented in a clear and comprehensible format. Many multimedia applications organize information based on its physical format, i.e., sound clips, still images, text, and so on. However it is also possible to organize information based on the learner’s plan to use information. Such “cognitive media types”, media organized around cognition rather than physical format, include definitions, examples, worked problems, and problem sets [Recker, Ram, Shikano, Li & Stasko 1995].

AlgoNet was an educational multimedia environment designed to test the effectiveness of these cognitive media types. Subjects using AlgoNet’s cognitive media organization fared significantly better in a post-test than those using a traditional physical media organization [Recker et al. 1995]. While AlgoNet did show that cognitive media types helped students learn, how to best interface cognitive media types into a learning environment remained an open question. AlgoNet2 was created in order to explore some of the possible interface options for supporting cognitive multimedia. AlgoNet2 includes exactly the same domain information as AlgoNet, but with several new enhancements that allow users to navigate more easily and keep better track of their progress through the system.

In order to test AlgoNet2 in a real-life situation, we deployed the system in an introductory undergraduate computer science class at the Georgia Institute of Technology. The results of our study provide insights into the factors important in the design of a computer-based educational environment.

The AlgoNet2 System

The AlgoNet2 program is a prototype of an educational software package incorporating multimedia technology combined with features applying recent findings in cognitive science. AlgoNet2 is the second version of a computer learning environment that is being developed to explore possible interfaces for cognitive media learning environments. This version, as its predecessor, AlgoNet, teaches basic concepts in graph theory.

Figure 1 shows a typical AlgoNet screen. Domain information is presented in the large window in the upper left hand corner of the screen. To the right of the domain window is the Topic Tree & History window. This window provides information about the organization of the domain information, the user’s current position within the system, and a graphical trace of pages the user has visited. In the lower left part of the screen are four cognitive media buttons that provide access to different kinds of cognitive media. In the box on the lower right, users can select from a list of questions to ask about the current topic. The bottom edge of the screen contains buttons for moving back to the last node visited, invoking the help system, and for exiting AlgoNet2.
A graph is made up of two elements: VERTICES and EDGES.
A VERTEX is a node in the graph. This node can represent a condition or a state in some situation. VERTICES can be reached by traveling along the EDGES between nodes. Hence, VERTICES can be thought of as the "stopping points" in the graph.

An EDGE is simply a line or path that connects two VERTICES. One EDGE can only connect two VERTICES. However, a single VERTEX may have many different EDGES connected to it.

Just as in AlgoNet, information presented in AlgoNet2 is divided into groups of pages. Each of these groups (or nodes) covers exactly one topic. Each page within a node belongs to one of four cognitive media types: definitions, examples, pseudocode, and animations (dynamic visualizations).

To allow access to pages within the current node, we provide graphical buttons for the cognitive media types (Figure 1, lower left corner). The small images simplify the mental association between the button and content or type of domain information "behind" it. The current active view is highlighted. Buttons corresponding to cognitive media types unavailable in the current node are "ghosted" and drawn in light gray. The four graphical
buttons remain on the screen at all times, unlike AlgoNet's buttons which were simple text boxes that changed from node to node.

AlgoNet2 allows for two different types of navigation between topics (nodes). One of these navigation styles was based on the idea that it would be very natural and familiar for a student to ask questions about the current topic. To avoid the rigors of natural language understanding, we instead presented a box with pre-formulated questions for each topic (Figure 1, lower right). When the user clicks on a question, a related topic containing the answer to this question is shown. The "back" button allows users to return to the previously visited node. It was hoped that the question-based format would encourage students to ask questions of both their peers and instructors as well as themselves. This question asking/answering behavior has been linked to learning goals [Chi, Bassok, Lewis, Reimann & Glasser 1988; Chi & VanLehn 1991; Ng & Bereiter 1991; Ram 1991; Ram & Leake 1995; VanLehn & Chi 1992]. We intentionally omitted a "next page" function (contained in the original AlgoNet) that allows users to proceed through the system in some pre-formulated, linear fashion. By removing this feature we forced the users to choose their own paths through the system, hoping to encouraging active learning and reflection on learning goals and strategies.

Another new feature in AlgoNet2 was the Topic Tree & History window. This window provides a hierarchical, "bird’s eye" view of the information in AlgoNet2. Each node is an instance or type of its parent, e.g., cycles and paths are two types of graph basics. We anticipated that adding this structure to the domain information would encourage students to incorporate a similar structure in their own mental models, facilitating the retention of the material.

![Figure 3. Topic tree links are shown as solid lines. User generated jumps are represented as dotted lines.](image)
Evaluation

The AlgoNet2 system was evaluated in anticipation of fielding the system as a routine component of the introductory level computer science curriculum at Georgia Tech. Participants in the empirical studies were 148 students enrolled in a first-year introductory computer science course. The AlgoNet2 study was designed to take the place of one of the regular laboratory sessions in this course.

One goal of the empirical study was to determine the extent to which students could use the system as a stand-alone module for self-directed learning. As such, the subject material included in the AlgoNet2 system was not covered in classroom lectures. In addition, no formal instructions were given to the participants on the use of the system itself.

The laboratory sessions in this course are not normally designed to provide students with hands-on experience in computer problem-solving. In order to evaluate the system in as natural a setting as possible, the empirical studies were conducted within the context of a routine laboratory assignment and were directed by the students' normal laboratory instructors and teaching assistants. Our lab assignment asked students to solve a problem involving graph theory imbedded in a situation from the popular computer game Doom. By applying graph theory to a graph representing a part of one level of the game, students were able to "win" the game and solve the lab.

The students were given a lab assignment and were told only that AlgoNet2 would be of assistance in completing the assignment. In order to successfully complete the lab assignment, students needed to learn an appropriate graph algorithm and underlying concepts, (one of two "minimum spanning tree" algorithms) from among those presented in the system and then execute the algorithm. No attempt was made to explicitly direct students to specific graph concepts or definitions.

After completing the lab assignment, the students were directed to exit AlgoNet2 and complete a post-lab questionnaire. The questionnaire elicited personal information from each student (e.g., about their previous computer experience) and asked some general system evaluation questions. The questionnaire also included a section that contained questions designed to test the students' understanding of some basic graph concepts.

In addition to the lab assignment and the post-lab questionnaire, there were two other sources of data. The AlgoNet2 system automatically logged each participant's interactions with the system. We also videotaped four volunteers to help identify any system usage difficulties or other activities of interest.

Results

<table>
<thead>
<tr>
<th>Navigation Method</th>
<th>Rationale</th>
<th>Average Uses per Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question-Asking Interface</td>
<td>Natural method of inquiry, encourages metacognition.</td>
<td>4.2</td>
</tr>
<tr>
<td>Topic Tree</td>
<td>Puts current node in context, shows overall layout of system, shows users where they've been.</td>
<td>22.6</td>
</tr>
<tr>
<td>Cognitive Media Icons</td>
<td>Provides consistent, easy access to the cognitive media types.</td>
<td>14.6</td>
</tr>
</tbody>
</table>

Table 1: Students used the topic tree far more often than the question-asking interface.

Recall that students had three different ways of navigating through the AlgoNet2. Table 4 recounts these methods and the rationale behind each one. The right hand column shows the average number of times each student used each type of navigation. Users used the topic tree far more than the question-asking interface even though the two provide essential the same functionality: the ability to move between different nodes in the system.

Overall, users showed a slight preference for examples and visualizations over pseudocodes and definitions. However there was no significant correlation between users' experience levels and their preference for one cognitive media type over another. Over two-thirds (68%) of the students responding said that they were satisfied with the AlgoNet2 system in the post-lab survey.

Students tended to view visualizations far longer than any other form of cognitive media type as measured by the amount of time spent with each media type (Figure 4). This tendency to dwell on visualizations was echoed in the original AlgoNet study [Recker et al. 1995].

Analysis of the numerical data produced several meaningful and statistically significant correlations. Students' time viewing the visualization was positively correlated with their performance on the lab assignment, \( r = .290, p < .01 \). SAT-verbal scores were negatively correlated with time viewing the mostly textual definition and...
example pages, \( r = .335, p < .01 \). SAT-verbal scores were also negatively correlated with use of the question-asking interface \( (r = -.211, p < .05) \). Number of uses of the topic tree interface was negatively correlated with the students' self-report of the difficulty of the lab \((r = .192, p < .1)\). Post-lab quiz scores correlated positively with students' viewing times for definitions and examples \((r = .323 \text{ and } .215 \text{ respectively, both at } p < .05)\).

Some correlations were conspicuously absent. Previous background, including major field, SAT scores, and previous computer science background had no significant bearing on students' performance on the lab or the post-lab quiz.

![Figure 4. Relative viewing times for cognitive media types. Students spent more than three times as much time viewing visualizations as any other media type.](image)

**Discussion**

Student reports of the difficulty of the lab were negatively correlated with their usage of the topic tree. Students using the topic tree more often found the lab to be easier overall. While this result is purely correlational, we are hopeful that we can demonstrate that our experimental interface does in fact increase ease-of-use.

The question-asking interface and the topic tree both provide the same function: the ability to move from one node to the next. However, the students used the topic tree far more than the question-asking interface. One reason for this difference might be that the topic tree allowed instant access to any node in the entire system while the question-asking interface only allows one-hop moves along the topic tree. The students' preference for the topic tree may be pure convenience. Alternately, the larger, more graphical topic tree may draw the students' attention. Finally, the topic tree may actually be a more comprehensible interface. Additional research is needed to explore these possibilities.

The negative correlation between SAT-verbal scores and time viewing definitions (which tend to be composed of plain-English definitions of concepts) can be explained in the following way: highly verbal students viewed the same number of definitions and examples, but needed less time to absorb the same amount of content as less-verbal students. This suggests that students with different abilities use cognitive media types differently.

One of the clearest messages in the log data is that "glitzy" multimedia, i.e., fancy animations, is not an atheoretical luxury. Multimedia is capable of capturing and holding the subject's attention far better than any other media type we studied. There was only one visualization in the entire system, but users spent far more time at this page than of any other category of page. This finding is confirmed by the post-lab questionnaire as well. When asked what the system lacked, students often requested more multimedia support. Students' suggestions ranged from, "More animations for algorithms," to "Add kewl (sic) sounds."

When queried on their learning strategies, very few students could give a clear answer, if any at all. Of the students responding to the question, the most common strategies cited were "wander randomly," and "read everything." Students also reported frustration with this type of instruction, citing a lack of "direction" and "what to do next information." This clearly demonstrates that our question-asking protocol was not enough to encourage students to effectively guide their own thinking process and suggests that there is a need for the effective teaching of learning strategies.

While students were unable to identify their learning strategies, experimenters circulating through the labs did notice a consistent learning strategy. Many students tended to scan all of the content information quickly, then begin solving the lab. As they worked on their lab, students often returned to the AlgoNet2 system, suggesting that
they were actively hunting for information they needed. This suggests that students do use learning strategies, but may not be aware of them.

**Future Work**

Our pilot study of AlgoNet2 produced large amounts of data. Our next step will be to continue the analysis of the data and report any additional trends discovered. When all the data have been analyzed, we will begin work on AlgoNet3, a third version of the AlgoNet system which will include more cognitive media types such as problem sets and worked problems as well as more multimedia and interactive components. The next system will also be applied to domains outside of graph theory.

Future versions of AlgoNet can be useful in three distinct ways. First, they can be deployed in actual classrooms as instructional tools. Second, we can continue to study students’ interaction with the systems in order to refine the learning environments and develop design guidelines as we have done in this paper. Finally, future systems can help us explore the learning itself, revealing students’ strategies and understandings of their own learning processes.

**References**


**Acknowledgments**

Funding for this research was provided by the Office of Naval Research (contract number N00014-95-1-0790) by the National Science Foundation’s SUCCEED initiative (cooperative agreement EID-9109853), and by the EduTech Institute.
The Role of Student Tasks in Accessing Cognitive Media Types

Michael Byrne†, Mark Guzdial.*, Preetha Ram**, Richard Catrambone†, Ashwin Ram*, John Stasko*, Gordon Shippey*, Florian Albrecht*

*College of Computing
†School of Psychology
Georgia Institute of Technology
Atlanta, Georgia 30332
**Emory University
Department of Chemistry
Atlanta, Georgia
byrne@cc.gatech.edu, guzdial@cc.gatech.edu, chempr@emory.edu, rc7@prism.gatech.edu, ashwin@cc.gatech.edu, stasko@cc.gatech.edu, shippey@cc.gatech.edu, florian@cc.gatech.edu

Abstract: We believe that identifying media by their cognitive roles (e.g., definition, explanation, pseudo-code, visualization) can improve comprehension and usability in hypermedia systems designed for learning. We refer to media links organized around their cognitive role as cognitive media types [Recker, Ram, Shikano, Li, & Stasko, 1995]. Our hypothesis is that the goals that students bring to the learning task will affect how they will use the hypermedia support system [Ram & Leake, 1995]. We explored student use of a hypermedia system based on cognitive media types where students performed different orienting tasks: undirected, browsing in order to answer specific questions, problem-solving, and problem-solving with prompted self-explanations. We found significant differences in use behavior between problem-solving and browsing students, though no learning differences.

Introduction

Hypermedia is typically oriented around physical media types, for instance, text, video, graphics, and audio. The World Wide Web’s use of Mime types, which are based on physical characteristics, is a good example [Berners-Lee, Cailliau, Luotonen, Nielsen, & Secret, 1994]. We believe that identifying media by their cognitive roles (e.g., definition, explanation, pseudo-code, visualization) can improve comprehension and usability in hypermedia systems designed for learning. We refer to media links organized around their cognitive role as cognitive media types [Recker, et al., 1995]. In a recent study, students using a hypermedia system organized around cognitive media types performed better on a post-test on the content in the system than students using a hypermedia system organized around physical media types [Recker, et al., 1995].

However, we also believe that the goals that students bring to the learning task will affect how the students will use the hypermedia support system [Ram & Leake, 1995]. Certainly, the system can be tuned to different tasks. Researchers working on the Superbook hypermedia system found that different interfaces better supported either browsing activity or searching for a particular kind of information [Egan, Remde, Gomez, Landauer, Eberhardt, et al., 1990]. Our question was whether setting different tasks for students (and thus, different goals) would affect (a) how students utilized cognitive media types and (b) student learning.

In particular, we explored two kinds of task differences:

• Problem-solving vs. browsing: We hypothesized that students who had a specific problem to solve would access the hypermedia organized around cognitive media types differently and would learn more than students who were more or less simply browsing for interesting content.

• Self-explanations: The work of Chi and others [Chi, 1992] [Chi, Bassok, Lewis, Reimman, & Glaser, 1990] suggests that students who self-explain the content and their actions learn more effectively than those who do not self-explain. We hypothesized that we might be able to prompt students to self-explain based on their activity in the hypermedia system to achieve gains in learning. Our efforts here are similar to those of [Bielaczyc, Pirolli, & Brown, 1991] [Bielaczyc & Recker, 1991] who also engineered prompts to encourage and teach self-explanation behavior. Our additional question was whether students prompted for self-explanations would access the cognitive media types differently than students without such prompts.

Self-explanation prompts are less interesting in a browsing condition than in a problem-solving condition.
because there is less student activity to explain. Therefore, we only used self-explanation prompts as a second problem-solving condition. We note, however, that the prompts in a self-explanation condition can also serve to direct student exploration. To explore that role, we had two browsing conditions – one without direction, and one directed to answer specific questions.

Methods

In an attempt to test some of our ideas about cognitive media types, students’ tasks, and self-explanation, we designed instructional software for teaching students how to solve problems involving the determination of molecular shape. This subject is particularly difficult for introductory Chemistry students so the potential benefits in this domain are considerable.

The software, which we called ChemLab, is based on an outline of the problem solution procedure developed by our domain expert (P. Ram). The steps in the procedure are laid out in a map that students can follow to arrive at a solution. There are 22 steps in the procedure that the students could examine. Each step contained up to four cognitive media types:

1) Definitions. Key concepts relevant to this step and the operations required at this step of the procedure were presented here.

2) Examples. Concrete examples of this step in the procedure or the key concept in the step were presented here.

3) Worked problems. This was generally a "before and after" presentation in which the students saw a partial solution before the step was performed and then after it was performed. Explanations for the operations were also present.

4) Problem sets. This was similar to a worked problem but provides an opportunity to test one's knowledge. The screen presents a "before the step" situation and asked the learner to execute the step in their head or on scratch paper. The learner could request that the solution then be shown to verify their solution.

Because this particular domain is very visual, ChemLab makes extensive use of figures along with the text [see Fig. 1]. However, use of other physical media types were limited. The semi-public computer cluster environment constrained our ability to make use of sound and time constraints made construction of animations impossible. Future versions of ChemLab will make more extensive use of animation to illustrate the more dynamic content, though the use of sound is still an unresolved issue.

Subjects

The subjects in the ChemLab experiment were approximately 80 undergraduates who were taking an introductory Chemistry course at Emory University. They participated in the experiment for extra-credit. Subjects had attended lectures (scattered over a two week period) in their course which addressed the material covered in ChemLab. The participant sample was a strong group of students with a self-reported grade-point average of 3.4 out of 4.0 and mean SAT scores of 571 on the Verbal scale and 663 on the Quantitative scale. Thus, it was a very capable group that used the ChemLab software. Random assignment procedures were effective in that there were no mean group differences on any of these participant variables.

Apparatus/Materials

Subjects were given paper guides to help them navigate the ChemLab software. The software itself was run on Macintosh Centris personal computers and developed in Apple's HyperCard (version 2.3).
When the central atom has four electron pairs around it, its geometry is tetrahedral and the angle between the bonds (or between any bond and any electron cloud of the lone pair) is 109.5°. A tetrahedron is a 3-dimensional pyramid shape with four sides.

If one of the electron groups around the central atom is a lone pair, the total geometry of the molecule is still tetrahedral but the shape of the molecule is trigonal pyramidal.

If two of the electron groups are lone pairs (and not bonds), then the geometry of the molecule is bent.

Figure 1: ChemLab screen shot

Design

All subjects had access to the same instructional material in the software and were given the same post-test after using the software. The post-test was a short quiz designed by the regular Chemistry instructors for the course. The ChemLab software also kept a log of mouse clicks so it was possible to analyze browsing patterns for the subjects.

The presentation of the ChemLab software was in slightly different contexts for different subjects. There were four presentation groups:

1) Passive watching. In this condition, subjects were given access to the ChemLab instructional material and little guidance. They were instructed to "work through the materials until you are pretty sure you are prepared to solve some problems." This is, in essence, the control condition.

2) Directed watching. In this condition, we wanted to give the subjects some learning goals while browsing through ChemLab. Thus, subjects were given a series of questions to answer to help guide their browsing. These questions concerned different aspects: some were oriented towards the procedure (e.g. "how do you do X?" or "what do you do after step Y?") , some were oriented toward specific content (e.g. "what's the chemical formula for Hydrazine?") , some were definitions, and many were purposely obscure things that subjects would have to find as distractors.

3) Problem solving. Subjects in this condition could not only browse ChemLab, but were asked to solve two specific molecular shape problems. As they were doing so, they were allowed to use ChemLab as a resource to help them solve the problems. This condition was included to encourage subjects to spontaneously generate learning goals directly relevant to solving real problems.

4) Problem solving with prompting. This condition was identical to condition 3 with one addition: at various points while subjects solved the problems, they were prompted to explain what they were doing and why. It was hoped that this manipulation would increase reflection and self-explanation.

Subjects in the two problem-solving conditions were asked to use special software called the Molecule Construction Kit to solve the problems [see Fig. 2]. This made the whole environment more self-contained and gave us the opportunity to examine partial solutions.
Procedures

Participants first filled out the consent form. After this, they were allowed to spend as much time as they wanted with the ChemLab software. The software also prompted them to report their grade-point average and SAT scores. Once they were satisfied with the time they had spent using ChemLab, they received a brief questionnaire asking them to rate their confidence in their ability to solve molecular shape problems. Once they had completed the questionnaire, they were given the post-test on paper. They were proctored during the post-test and given unlimited time to solve the problems on the quiz.

An example problem on the post-test was: Arrange the following species in order of decreasing $F-A-F$ bond angles, where $A$ is the central atom: $BF_3$, $BeF_2$, $CF_4$. The correct solution was $BeF_2$, $BF_3$, $CF_4$.

Results

Browsing patterns were clearly affected by the condition to which subjects were assigned. Subjects in the problem-solving conditions visited an average of only 5.4 of the 22 steps, while subjects in the browsing conditions visited an average of 17.6 of the 22 steps, and this difference is reliable ($F(1,74) = 82.70$, $p < 0.001$). This is an extremely large difference; on average, subjects in the problem-solving conditions saw less than one-fourth of the steps in the procedure while browsing subjects saw 80% of the steps. This difference carried into the number of visits to the different media types; for all four media types, subjects in the browsing conditions had more visits than subjects in the problem-solving conditions.

This may have been compensated for by the amount of time subjects spent looking at the screens when they were presented. An average visit to a screen for the subjects in the browsing conditions lasted 36 seconds, while for the problem-solving subjects the average visit lasted 86 seconds. Again, this difference is reliable ($F(1,43) = 12.10$, $p =$
0.001) and is quite large; subjects in the problem-solving conditions spent almost two and a half times as long looking at a screen of information than subjects in the browsing conditions. The average total time (not including reading the initial instructions) for the experiment was 17.85 minutes. This was weakly related to post-test score ($r(77) = 0.62, p = 0.058$).

While there were clear effects on browsing patterns for problem-solving vs. browsing, there were no effects associated with prompting except the total amount of time spent using ChemLab. The source of this difference is probably the extra typing subjects in the prompted condition did in answering the prompts. In any case, the difference here, though reliable, was not large.

Unfortunately, there were no reliable differences in post-test scores between the groups. Table 1 summarizes the four groups' post-test scores. Overall, subjects did reasonably well on the post-test, averaging 8.15 points of a possible 10. Interestingly, self-reported confidence (the rating taken right before the post-test) was only weakly correlated with the actual quiz scores, $r(80) = 0.20, p = 0.07$; self-reported confidence was more highly correlated with SAT Quantitative scores $r(69) = 0.28, p = 0.02$ even though SAT scores were not correlated with quiz performance.

<table>
<thead>
<tr>
<th>Group</th>
<th>Average Score</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8.34</td>
<td>1.91</td>
</tr>
<tr>
<td>2</td>
<td>8.34</td>
<td>1.26</td>
</tr>
<tr>
<td>3</td>
<td>8.00</td>
<td>1.52</td>
</tr>
<tr>
<td>4</td>
<td>7.96</td>
<td>1.59</td>
</tr>
</tbody>
</table>

Table 1: Summarizing Four Groups' Post-Test Scores

Discussion

The lack of differences between groups' post-test scores are surprising – our hypotheses predicted differences both in browsing behavior (which was observed) and in learning performance (which was not). However, this lack of difference might be attributed to the complexity of the task (or rather, lack thereof) and student ability. The students were very capable, and even those who received relatively little support from the hypermedia system (i.e., the problem-solving group students who saw less than 25% of the steps in the procedures) performed very well on the post-test (8.15 out of 10 points, with a standard deviation of less than 2.0 – almost nobody got less than half the quiz right), which may also suggest a ceiling effect. The students were doing so well that there was little opportunity to discover group differences.

More interesting is the dramatic differences between use patterns among the groups. Students who were trying to solve a problem visited fewer cognitive media types but spent more time studying the media that they did visit. It may be that if the task were more complex and the students needed more of the information in the hypermedia system to solve the task, as opposed to having seen much of the information in class already, the differences in use behaviors might result in greater differences in performance on the learning task and on the post-test.

The observation that problem-solving students spent more time studying the screens is in marked contrast with prevailing wisdom in hypermedia design. Generally, hypermedia designers reduce the amount of text in their systems in favor of more “glitzy” media, such as graphics, sound, and video, arguing that users are not willing to spend the time reading text. In fact, some researchers have even claimed that reading will become an obsolete skill as multimedia computers become more common [Papert, 1993]. While the case and value of “glitz” is real and important, our data suggest that if users are actively trying to problem-solve and use the content in a hypermedia system, they will invest the time on the content, perhaps even reading the text.

Our results offer no insights on the question of self-explanation prompts. We have no evidence of differences in learning performance nor of differences in use behavior due to self-explanation prompts, other than the time differences required to respond to the prompts. It may be that if we were to repeat the experiment with a more complex task and with more necessary hypermedia content, then students might be more motivated to respond to the prompts more thoughtfully, resulting in greater learning differences.

Conclusions

While we have not been able to support our hypotheses about self-explanation and about learning effects, our results do support the hypothesis that a difference in user task will cause a difference in browsing behavior. Thus, there are several paths of interest for future work:
Certainly, repeating the experiment with a more complex task where hypermedia support is more critical might provide useful insights on the learning question. Since the problem-solving students spend more time studying content but view less of the overall system, good navigation mechanisms that lead these students to the necessary content will be a critical feature of future systems [Shippey, Ram, Albrecht, Roberts, Guzdial, et al., 1996].

Time spent on the content suggests an interesting avenue for an exploration where the relative amounts of different cognitive media types are varied to investigate further the relationship between media types, task, and learning.

Varying the kinds of physical media corresponding to cognitive media and varying the kinds of interaction with the system offers another interesting interface component whose use may vary with task.

References


