Education and Information Technologies **3** 277–290 (1998)

A design for a hypermedia-based learning environment

OSSI NYKÄNEN
Tampere University of Technology, Digital Media Institute, Hypermedia Laboratory, P.O. Box 692, FIN-33101 Tampere, Finland.
E-mail: ossi.nykanen@.cc.tut.fi

MARTTI ALA-RANTALA
Tampere University of Technology, Digital Media Institute, Hypermedia Laboratory, P.O. Box 692, FIN-33101 Tampere, Finland.
E-mail: martti.ala-rantala@iki.fi

This paper presents ideas and a design for a Hypermedia-Based Learning Environment, HBLE for short. As the system is in implementation phase, we are also able to present some implementation techniques, problems and solutions.
HBLE offers tools and methods for course development, teaching, maintenance, and different learner-centered study strategies. The system also has information acquisition functionality for research purposes.
We study structuring the learning material and how to adapt it for individual students. As collaboration is an important aspect of learning process, the system also includes tools and research instruments for collaborative activities between the actors in the learning process. The concrete outcomes of the project are a platform for Web-based courses and experimental courseware. **© 1998 IFIP, published by Kluwer Academic Publishers**

KEYWORDS: distance learning; design technology; Internet; open learning.

INTRODUCTION

There is an increasing interest in creating WWW-based learning tools and learning environments. Such tools use the general methodology and techniques that have resulted from the work of computer-based education and artificial intelligence pioneers in the overlapping fields of computer aided instruction (CAI), intelligent tutoring (IT) and adaptive hypermedia (AH). Although the WWW has made the production and deployment of distributed learning material easy, producing high-quality learning material requires more advanced tools than simple HTML editors.

Learning environments should provide students with tools and methods for motivation and effective self-studying. This means applying student adaptive techniques and a 'learning by doing' ideology when creating learning environments. Learning environment should also provide feedback and feelings of success.

Information presentation, user adaptation and collaboration are the fundamental sectors of learning environments, and an ideal system should combine all three. The task of creating educational hypermedia is far from trivial since it involves coupling a number of different types of theories and techniques such as goal-oriented teaching, hypermedia management and student modeling.

A major problem of realizing all this is actually designing and implementing such a large-scale system in practice. This is where software designers come in, since the process of designing and constructing open learning environments can benefit from lessons learned, e.g., from object-oriented software design.

A common pitfall in creating educational systems has been using the latest technology for its own sake as a starting point of the design. This has lead to building prototypes with little long-term use. We try to overcome this problem by co-operating with pedagogical experts, students and teachers.

This paper presents ideas and a design for a Hypermedia-Based Learning Environment (HBLE) designed primarily for teaching university level mathematics. HBLE will offer tools for authoring learning material, teaching, studying, and performing research within the learning environment. The main contribution of the final system will be combining the most important social and educational, as well as the crucial technical and material aspects of the learning and teaching processes to a reasonable extent using modern technologies and pedagogical theories. In essence, this integration includes coupling educational content, mechanisms of adaptive hypermedia and distance teaching and learning methods in a technically and pedagogically correct way.

This paper provides a snapshot of the system being studied and constructed at the Hypermedia Laboratory of TUT in its current state. The main objective of this paper is to capture important methodological and technical viewpoints, problems and solutions in relation to this ongoing work.

Pedagogical aspects of the system are explained, e.g., in (Ruokamo-Saari and Pohjolainen, 1997). We are also using and revising ideas resulting from previous work at TUT in the field of hypermedia-based educational systems, see (TUT, 1995) and (TUT, 1996).

FUNDAMENTALS OF HBLE

HBLE couples different types of educational hypermedia disciplines and techniques. These include structuring course content, communication, collaboration, and recognition of different types of user groups.

Organization of courses

Courses form the material backbone of educational content in open learning environments. Since HBLE relies on adaptive hypermedia, courses have both static and dynamic content that may appear differently to different users and include, e.g., time and user-specific information. To achieve this with practical authoring methods, we introduce three different types of courses in the system: abstract courses, concrete courses and user instances of courses, see Fig. 1.

Abstract courses consist of user-independent presentation of the structure and hypermedia content of some specific course. The hypermedia content of abstract course material is internally presented in the form of a directed acyclic graph that we call a *knowledge graph*. In its construction, a knowledge graph is essentially a special kind of weighted directed graph. However a knowledge graph has certain constraints; parallel arcs not allowed, extra arcs in paths between cells are reduced, etc. Arcs and cells have also various kinds of attributes, e.g., for student data, identifiers, labels and visual properties.

Figure 1 presents an example of a simple knowledge graph with information associated with an abstract course, a concrete course based on that, and information specific to one student instance of the course. Each cell of the knowledge graph of a abstract course is presented as a small hypermedia document. Arcs in the graph define the prerequisite relations between the cells. For instance, in the previous example the student must understand the concepts *set* and *scalar* to understand *vector*. An approach using relatively small, linked knowledge elements has many advantages. First, it makes it possible to effectively define specific goals for studying. Second, each student's knowledge of each topic can be estimated separately, enabling automated navigation guidance, and third, the hierarchical structure of information is stated explicitly.

In addition to prerequisite relations between cells, static and interactive contents, the structure of a abstract course includes also the initial design of, e.g., links to

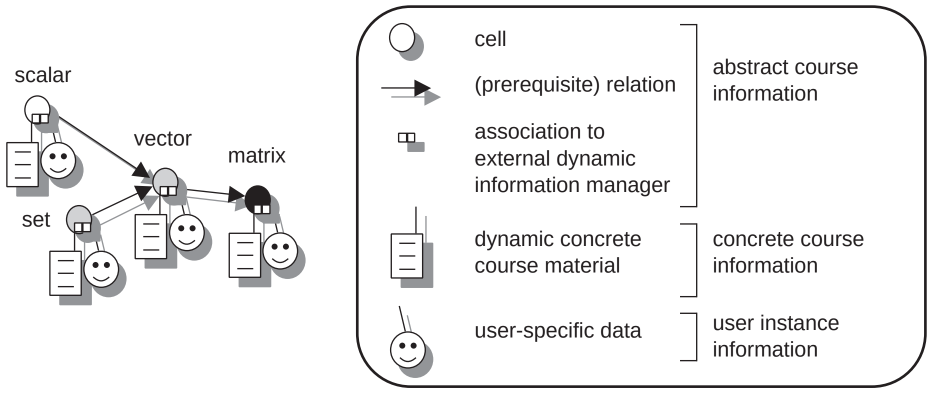
scalar
vector
matrix
set
cell
(prerequisite) relation
association to external dynamic information manager
abstract course information
dynamic concrete course material
concrete course information
user-specific data
user instance information

Figure 1. Hierarchical course structure

bulletin boards, dynamic data structures and acquisition mechanisms and access to course-specific tools. Abstract courses can be considered templates for concrete courses. *Concrete courses* are active courses being taught and studied and are based on materials and structures of a specific abstract course. In addition, a concrete course includes timetables, bulletin boards, user and workgroup information. One abstract course can act as an template for any number of concrete courses. Concrete courses exist in various forms; some progress in parallel to conventional lectures while others live independently on the Web.

A user instance of a course is a user-specific image of a concrete course. For students, for example, it includes the student's progress measurement data, personal annotations and resources and statistics. User instances of a course apply and provide various adaptive methods and tools specific to users and to various user groups in general. The content and linking of actual hypermedia documents may vary based on co-operative user modeling.

Using abstract courses as templates for concrete courses and user instances provides a practical mechanism for dynamically developing and updating courses. Since one abstract course possibly provides static content and structure for number of concrete courses, changes in one abstract course are reflected in all associated concrete courses. On the other hand, redesigning just one concrete course is merely a matter of duplicating and modifying the abstract course and creating a new concrete course based on this updated copy.

Course material

The material of each course consists of a set of cells that can be visualized as associated hypermedia documents. Each cell introduces a single topic or a virtual section. Each topic consists one simple issue, for instance, a definition or a theorem. Virtual sections present no new information in the system but define logical groups of topics instead. They bind the relevant topics together in a meaningful fashion providing overview of the topics discussed with introduction, general background, examples, exercises and tests. Virtual sections may be combined hierarchically into larger entities.

Hypermedia documents consist of different kinds of hierarchical elements. These include theory, exercises, theoretical and practical examples as well as tests divided into difficulty levels by the author. Material is composed of combinations of text, graphics, simulations, audio, video, interfaces to mathematical software and other media types.

Documents are stored along with abstract courses in a general presentation form. User-specific document instances are created on demand by the HBLE server. Students receive documents adjusted to their skills while teachers and designers receive more information based on their individual classification and needs. Currently material development is based heavily on teaching mathematics, but in the future the scope of our approach may be extended into other disciplines as well.

The hierarchical organization of hypermedia documents using topics and virtual sections provides students with a familiar and motivating granularity similar to conventional books. HBLE also provides for users a visual and non-linear navigation tools, intelligent navigation aids and search mechanisms. In addition to embedded hypermedia elements, extended course material includes a varying set of user category -specific tools including annotation and text editors, authoring tools, simple worksheet tools for teachers and course-specific student tools to be used, e.g., parallel to demonstrations and exercises. These features and session-based functionality provide users with convenient methods for working effectively with the system.

Authoring courses is based on a three-step procedure of producing course material: designing structures for courses, generating content, the necessary hypermedia materials and interactive components, and combining these using a special authoring toolkit. The toolkit includes a tailored programmable macro language suitable for writing structural hypermedia and provides mechanisms such as automatic linking and content indexing. The same source material can be readily converted into various visual and functional forms.

After creating an abstract course, establishing a concrete course requires duplicating the structure of the abstract course and adding relevant dynamic information to it. Then users can be assigned to the concrete course and the course can be started.

Collaboration and communication

Collaborative activities, that is, working in a group and communicating with the teacher, are essential for the learning process. There are several forms of collaboration and communication supported by HBLE. The student may ask the teacher questions about the course. Each concrete course has a discussion forum associated with it. The forum is organized like the customary bulletin boards or news systems. Further, students and teachers may send comments about the course material to the author.

Social learning is supported by workgroup activities. Each working group is assigned a discussion forum and workspace with collaborative tools. Groups can assess their own work and present it publicly.

One method of assessing the student's comprehension of a topic is to have the student answer questions in essay format. The answers are graded by a human teacher and used as a basis for progress measurement just like other assessment methods.

User categories

HBLE has three main user categories: visitors, students and staff. Staff includes authors, teachers, administrators and researchers.

A visitor is just like a student flipping through the pages of a textbook in a bookstore. This category is essentially for demonstration purposes, since visitors do not participate in a concrete course in any way. The usage of the system is limited: the learning results are not stored, and the use of supplementing software, e.g., course-specific tools using Maple, may be limited. The collaborative functions are mainly disabled.

A student is registered at the university. When he/she enrols in a course through HBLE, notification is sent to the university's course enrolment system. The student takes the class according to the standard university procedure and has the right to use the aspects of HBLE that require teacher contribution. Basically, students use HBLE as one form of course material.

Authors produce the course content and are responsible for the FAQ lists. They create topics and relations between them, and group the topics into virtual sections.

Teachers make use of the system as teaching material. There is at least one teacher assigned to each course. The teacher follows the progress of students, answers the questions posed by the students and participates the discussion about the course. The teacher also grades the answers to essay problems.

Researchers use the system for information acquisition purposes. The system collects statistics about the paths students take through the material, the usage of various features of the system, the frequencies of visits to individual topics etc. Researchers can extract reports from this data.

The administrator manages the HBLE system, for example, adds users and cleans up after a course is through. This is a super-user role that requires strict security measures.

STUDENT MODELING

Since our aim is to design and create an adaptive learning environment with elementary tutoring capabilities, the need for a student model is obvious. Our approach suggests hypermedia cell structure as a natural starting point for student modeling, which quite naturally leads into using overlay modeling techniques.

Description of the model

The abstract course defines a knowledge graph that can be used as a basis of student modelling. The basic version of the knowledge graph consists only of cells and prerequisite relations between them, see Fig. 1. The idea of the prerequisite relation is simple. Cell A is said to be prerequisite to cell B if there exists a path from A to B. In other words, the prerequisite version of the knowledge graph indicates what topics should be known before studying the topic at hand. Arcs of the graph may be weighted to compensate for the differences in the amount of information presented in individual topics. Other types of semantic relations may be introduced later when experience is gained in using the system.

Based on student-specific evaluation (for instance, completing exercises and voting), each cell is assigned a *topic knowledge value*, a scalar representing the student's measured knowledge of the topic. In effect, cells can be regarded as the smallest knowledge items identified by the student model.

The student model associated with prerequisite semantics of the knowledge graph makes automated evaluation, student-specific guidance and analysis of goal-oriented studying possible. The knowledge graph stored in the student's instance of the course is equipped with a *comprehension measure*. Comprehension measure is a scalar-valued function based on the course's knowledge graph and knowledge values assigned to the cells of the student's instance of the course.

The comprehension measure allows analysis of the state of a user's contextual knowledge locally or in relation to large entities. With the comprehension measure HBLE is able, e.g., to pinpoint a student's strong and weak areas of knowledge and thereby help the student to focus his or her efforts on related areas. This can also be used as a basis for automatic learning strategy planning, for example by offering the student a selection of alternative strategies for basic skills improvement or target-oriented studying.

Updating the model

Since the validity of the student model is based on the accuracy of the topic knowledge measurements, updating these is critical. In most human-computer interaction the communication channel is very narrow; computer systems are incapable of effectively using natural languages, they lack the ability to understand nonverbal messages, and so on. This indicates that in order to gain relevant information from the user an exact frame of interaction has to be employed.

In our case updating the student model is based on combining information acquisition using a voting mechanism and controlled exercises defined by the *student model interface*. The student model interface is a special kind of application interface that specifies how different types of interactive exercises are added to the system. The student model interface defines exactly how and when the student model is updated. This makes it possible to add new interactive modules and new features to the interaction between a student and the student model.

Interaction modules are, for instance, adaptive applications specializing in running different types of controlled exercises. The modularity ensures that the interaction modules can be used with a variety of content domains without rewriting any of HBLE key components.

The development of the student model is non-monotonic. A student's knowledge state is changed according to the correctness of the student's decisions and exercise answers. Correct answers increase the estimate of the student's knowledge state accordingf the student's decisions and exercise answers. Correct answers increase the estimate of the student's knowledge state according to the difficulty of the exercises, while poorly solved exercises, clearly below the student's abilities, decrease it.

SYSTEM ARCHITECTURE AND IMPLEMENTATION

The governing design goal is to create a layered, loosely coupled and distributed object-oriented framework that is easily extensible and modifiable. We expect HBLE to remain in use for several years to come as a vehicle for study and research. During this time, the system needs to be maintained like any software system. A well thought-out architecture is vital to this end.

The HBLE runtime system has two main components: the client running in the Web browser and the server running in a Unix workstation. In addition to these, HBLE includes authoring (described briefly in 2.2) and administration tools.

We distinguish, quite conventionally, two levels of architecture: system and application. The system architecture is a three-tier distributed object system, see Fig. 2. The application architecture on the client side is based on MVC++ (Jaaksi, 1995), a modified version of the Model-View-Controller architecture that originates from the venerable Smalltalk programming environment. Also the physical structure of the system is of relevance.

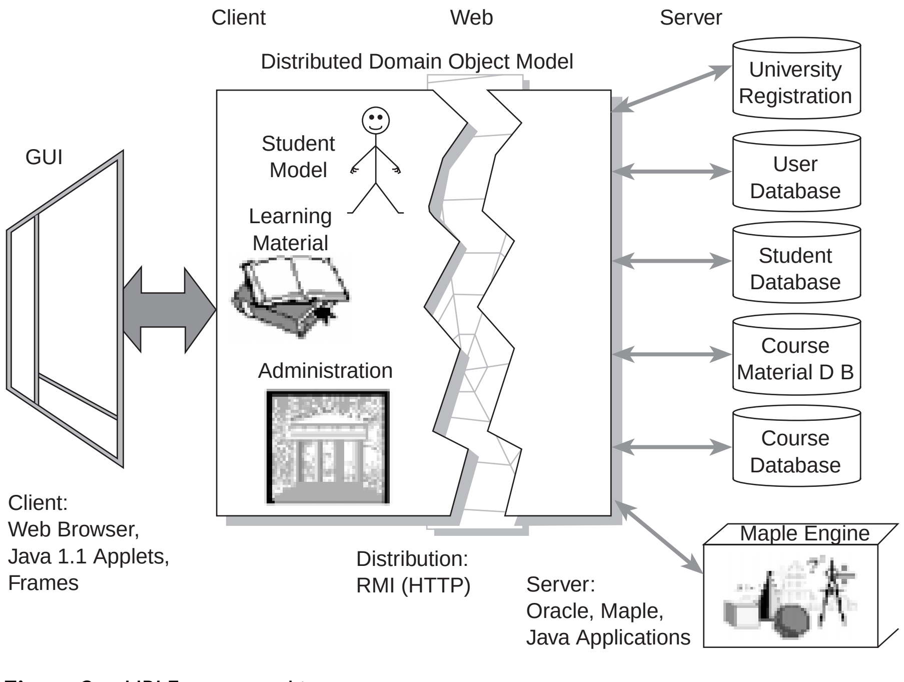

Figure 2. HBLE system architecture

Physical architecture

The server comprises a gatekeeper process plus a set of session server processes. The gatekeeper listens to incoming login requests. After authentication it creates a user session and spawns a server process associated with that session. The client is based on a Java-enabled Web browser. *Client Manager*, an applet that lives for the entire duration of the session, controls user interaction and communicates with the session server process.

Both client and server are implemented in Java using JDK (Java Development Kit) 1.1. This gives us a clean and uniform environment: the same object model and the same language are used throughout the system. On the other hand, using Java may induce performance penalties.

System architecture

At the core of the design is the Distributed Domain Object Model (DDOM) that captures the domain entities and their behavior. Examples of such entities in HBLE are user, student model, course, topic and knowledge graph. For instance, a user object is capable of loading its data from and storing it to permanent storage. In the case of a student user, the object also has operations like retrieving the list of courses the student is enrolled in. Below, we shall briefly discuss some of the various types of classes that designing a DDOM produces.

Distributed classes look like regular (Java) classes to the client, but implement their functionality on the server side. *Proxy* classes are distributed classes whose implementation is split between client and server sides. To create a distributed object system, a set of common problems must be solved. Standard solutions exist, and we have chosen to use RMI (Remote Method Invocation), a package that is a part of JDK 1.1.

An instance of a *persistent* class may outlive the process that created it. Hence, such an instance must be able to save and load its internal state so that we can store the object and later restore it, perhaps in another process space. We use an Oracle database and a set of flat files for permanent object storage. HBLE accesses the database by way of JDBC (Java Database Connectivity) interfaces provided with JDK 1.1 and Oracle JDBC drivers.

The same database engine we use in HBLE also runs the information system of our university. HBLE connects to this system, e.g., to verify that a student wishing to use the system is enrolled at the university.

Application architecture

According to MVC++, the application is arranged in three layers, each of which has a well-defined set of duties, see Fig. 3. HBLE offers application designers a frame-

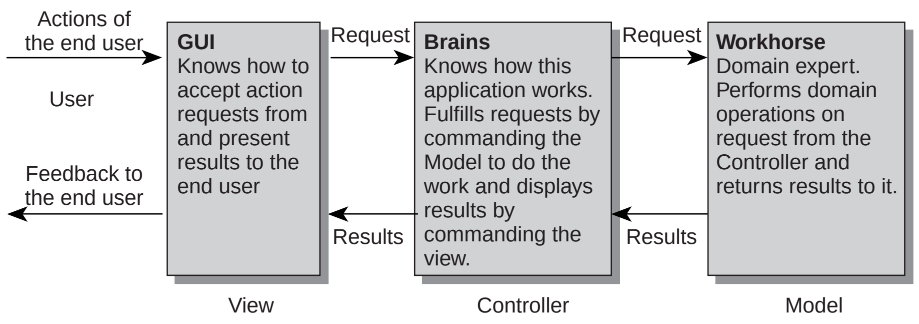
Actions of the end user
User
Feedback to the end user
GUI
Knows how to accept action requests from and present results to the end user
Request
Results
Brains
Knows how this application works. Fulfills requests by commanding the Model to do the work and displays results by commanding the view.
Request
Results
Workhorse
Domain expert. Performs domain operations on request from the Controller and returns results to it.
View
Controller
Model

Figure 3. The workings of MVC++

work that hides the details of underlying environment, such as the Web browser and the distribution mechanism.

In MVC++, there is always an application object that creates and governs the Model-View-Controller triad. In HBLE, Client Manager acts as the MVC++ application object.

Figure 4 illustrates the logical structure of HBLE GUI or the View. On top, the main window has a command bar frame that displays the commands that are presently active. The content window frame is in one of two modes: single or split. Typically, in the split mode the secondary window would contain navigational information whereas the content window would display the actual material under study. Floating windows are used, for instance, for course-specific tools.

The Model part consists of an object hierarchy that contains both local and distribu-

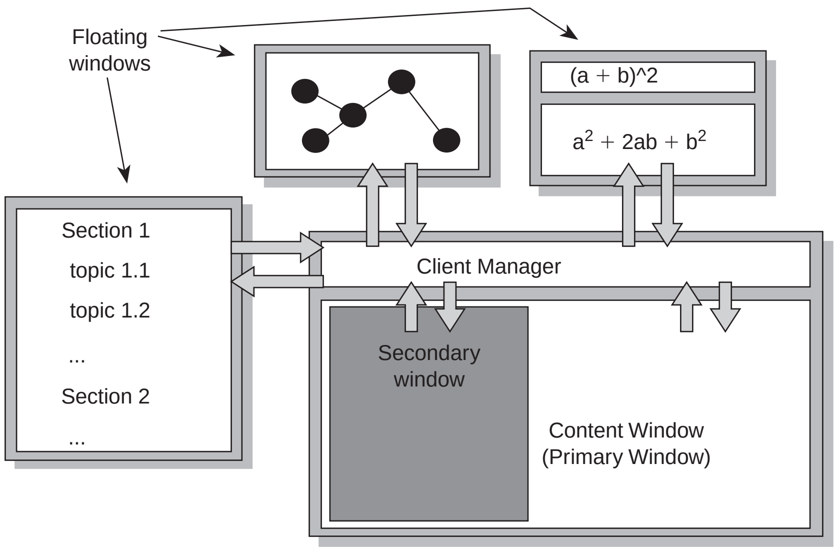

Figure 4. Logical GUI structure

ted objects. The Controller receives events from the user via View, from other applets as well as from the Model. In response to these events, the Controller changes its state and performs appropriate actions by commanding the View and the Model.

Implementation problems

Our implementation language, Java, has the reputation of being a clean, logical and rather pure OO language. This may well be true for Java alone, but the Web browser environment brings about some additional complications. This makes implementing the architecture described quite challenging.

From the implementation point of view the outcome of this research project is two-fold. We will test the feasibility of building the architecture previously described in Web/Java environments in general, and particularly its suitability for creating Web-based learning material. The implementation is currently in too early stage to give answers to these questions.

RELATED WORK

Creating educational hypermedia and related software involves the coupling of different types of theories and technologies. Information presentation, user adaptation and collaboration are the fundamental sectors of learning environments and an ideal system should combine all of them. Today most systems work effectively in one of these areas. A brief view of the field indicates that the WWW has become a rather popular test field for running stand-alone curricula. Sites fall typically in the following categories; Web-based curricula and non-interactive static courses, individual tutorials and interactive exercises and educational ftp-like resources. Some sites, such as *The World Lecture Hall* (ACITS, 1998), and *Mathematics Archives WWW Server* (MATHARC, 1997), gather links to various WWW course resources providing a good starting point for the casual browser.

Most sites available on the WWW present the subject matter in a book-like fashion sometimes incorporating some tutoring capabilities. Sites such as *ELM-ART*, (ELMART, 1997), *Mathmania* (UVIC, 1996), and *Interactive Real Analysis* (Wachsmuth, B. G., 1996), present straightforward learning material in an interesting fashion but offer little or no interaction. In addition to individually crafted courses, there are also few large-scale development environments for creating and running Web-based courses. A good example of such an environment is *WebCT* (WEBCT, 1998), a system for creating, authoring and teaching Web courses.

Most of the individual exercises on the Web are based on the intuitive use of HTML forms or Java applets. The main problem with these is that with few exceptions, most of them are created from scratch with little systematic planning. Lists of exercises can be found, e.g., from the site *Mathematics Archive: JAVA and Other Interactive WWW Pages* (INT, 1998). Another type of educational resource is, for instance, *Mathematica* or *Maple* resource packages consisting software-specific features.

All research related to the area of educational hypermedia is not published as HTML or distributed through URL addresses. From our point of view the most interesting conventional publications deal with formalizing human-computer interaction and structuring information. Publications are not discussed here in detail; relevant references are listed in the end of this document.

One interesting field study of exploratory learning strategies in an average office environment conducted by Rieman (Rieman, 1996), however, is worth mentioning. Rieman studied the learning strategies of 14 informants with different backgrounds using applications like word processors, programming tools, email and database programs. Rieman found evidence that task-driven and 'just-in-time' approaches are the main strategies when learning under time pressures. This is what most researchers would have expected. The rather striking overall finding was, however, that on-line help systems were inferior to more conventional manuals. The fact that informants preferred paper manuals implies that on-line help systems need revising. This, on the other hand, suggests how the development of goal-oriented learning environments should evolve. Implications are clear: User-Centered Design (UCD) should play a major role in designing educational hypermedia.

DISCUSSION

In this article we have presented ideas and a design for a hypermedia-based learning environment. At present, the system is in the design and implementation phase. Our approach combines the WWW, a method for learning material structuring, adaptivity based on student modeling, and modern object-oriented software engineering methods.

After the first functional version is up and running, it will be used for conducting both pedagogical and technical experiments. The data gathered by HBLE during these experiments will be analyzed to produce three main types of output: information about learning and teaching, information about using open learning environments in general, and finally technical information about HBLE itself that will aid further development of the system.

ACKNOWLEDGEMENTS

The research is supported by the Academy of Finland, the Finnish Ministry of Education and Tampere University of Technology. Special thanks to our colleagues for their helpful comments.

REFERENCES

ACITS (1998) *The World Lecture Hall*. A collection of worldwide links to educational material, University of Texas at Austin. Referenced March 25, 1998. <http://www.utexas.edu/world/lecture/index.html>

Brusilovsky, P. (1996) Methods and Techniques of Adaptive Hypermedia. *User Modelling and User-Adapted Interaction*, **6**, 87–129.

Carbonaro, A., Maniezzo, V., Roccetti, M. and Salomoni, P. (1996) Modeling the Student in Pitagora 2.0. *User Modelling and User-Adapted Interaction*, **4**, 233–51.

Cockburn, A. and Jones, S. (1996) Which way now? Analysing and easing inadequacies in WWW navigation. *Human-Computer Studies*, **45**, 105–29.

Dillon, A. and Watson, C. (1996) User analysis in HCI – the historical lessons from individual differences research. *Human-Computer Studies*, **45**, 619–37.

ELMART (1997) *ELM-ART: Episodic Learner Model – The Adaptive Remote Tutor.* A course on Lisp programming in English and German, University of Trier, Germany. <http://www.psychologie.uni-trier.de:8000/elmart>

Hohl, H., Bêcker, H-D. and Gunzenhäuser, R. (1996) Hypadapter: An Adaptive Hypertext System for Exploratory Learning and Programming. *User Modeling and User-Adapted Interaction*, **6**, 131–56.

INT (1998) *Mathematics Archive: JAVA and Other Interactive WWW Pages.* A collection of interactive pages with mathematics-related content, University of Tennessee, Knoxville. References March 25, 1998. http://archives.math.utk.edu/cgi-bin/interactive.html>

Jaaksi, A. (1995) Implementing Interactive Applications in C++. *Software Practice & Experience*, Vol 25, No. 3, March 1995, pp. 271–89.

Kaplan, G., Fenwick, J. and Chen, J. (1993) Adaptive Hypertext Navigation Based On User Goals and Context. *User Modeling and User-Adapted Interaction* **3**, 193–220.

Lee, K., Lee, Y. and Berra, P. (1997) Management of Multi-structured Hypermedia Documents: A Data Model, Query Language, and Indexing Scheme. *Multimedia Tools and Applications*, **4**, 199–223.

Lucarella, D. and Zanzi, A. (1996) A Visual Retrieval Environment for Hypermedia Information Systems. *ACM Transactions on Information Systems*, **14**(1), 3–29.

MATHARC (1997) *Mathematics Archives WWW Server.* A selection of Internet mathematical resources, University of Tennessee, Knoxville. Referenced March 25, 1998. <http://archives.math.utk.edu/>

Norman, D. and Spohrer, J. (1996) Learner-Centered Education. *Communications of the ACM*, **39**(4), 24–27.

Pinson, L. J. and Wiener, R. S. (1988) *An Introduction to Object-Oriented Programming and Smalltalk*. Addison-Wesley, Reading, MA.

Pohjolainen, S., Multisilta, J. and Antchev, K. (1996) Matrix algebra with hypermedia. *Education and Information Technologies*, **1**, 123–141.

Powell, D. (1996) Group Communication. *Communications of the ACM*, **39**(4), 50–53.

Ragnemalm, E. L. (1996) Student Diagnosis in Practice; Bridging a Gap. *User Modeling and User-Adapted Interaction*, **5**, 93–116.

Rieman, J. (1996) A Field Study of Exploratory Learning Strategies. *ACM Transactions on Computer-Human Interaction*, **3**(3), 189–218.

Ruokamo-Saari, H. and Pohjolainen, S. (1997) Pedagogical Issues for the Design of a Hypermedia-Based Learning Environment. In *Proceedings of the IFIP WG 3.3 Working Conference*. Edited by Darina Dicheva and Ivan Stanchev. Sozopol, Bulgary, May. pp. 82–91.

Soloway, E. and Pryor, A. (1996) The Next Generation in Human-Computer Interaction. *Communications of the ACM*, **39**(4), 16–18.

TUT (1995) *Matriisilaskenta I* (*Matrix Algebra I*). A Web course in Finnish, Tampere University of Technology. Referenced March 25, 1998. <http://matwww.ee.tut.fi/matriisi/toc73109.html>

TUT (1996) *Johdatus Korkeakoulumatematiikkaan* (*Introduction to University Mathematics*). A Web course in Finnish, Tampere University of Technology. Referenced March 25, 1998. <http://matwww.ee.tut.fi/jkkm/toc.html>

UVIC (1996) *MATHMANIA*. A Web course on advanced mathematical topics, University of Victoria, Canada. N.B. currently down. Referenced March 25, 1998. <http://csr.uvic.ca/~mmania/>

Wachsmuth, B. G. (1996) *Interactive Real Analysis*. A Web course, Seton Hall University. Referenced March 25, 1998. <http://www.shu.edu/projects/reals/reals.html>

WEBCT (1998) *WebCT*. A Web course authoring tool, University of British Columbia. Referenced March 25, 1998. <http://homebrew.cs.ubc.ca/webct/>