OwlNet: An Object-Oriented Environment for WBE

Paulo Alencar*, Donald D. Cowan*, Sergio Crespo♣[1], Marcus Felipe M. C. da Fontoura♣, and Carlos José P. de Lucena♣

*Computer Systems Group, University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

♣Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225, 22453-900 Rio de Janeiro, Brazil
(e-mail: [alencar, dcowan]@csg.uwaterloo.ca, [crespo, mafe, lucena]@inf.puc-rio.br)

PUC-RioInf.MCC18/98 Abril, 1998

ABSTRACT

The paper presents the design and the architecture of a new Web-Based Education (WBE) environment. This environment is based on the integration of two other WBE's applications: AulaNet™ and LiveBOOKs. Here we present the main features and the architecture of those two environments to justify our motivation for the development of OwlNet. Then we show the architecture and features of this new environment.

KEY WORDS: WBE environments, AulaNet™, LiveBOOKs, LivePAGE, object-oriented models.

RESUMO

O artigo apresenta o design e a arquitetura de um novo ambiente para educação baseada na Web (WBE). Este ambiente é baseado na integração de dois outros: AulaNet™ e LiveBOOKs. Aqui nós apresentamos as principais funcionalidades e a arquitetura desses dois ambientes para justificar a nossa motivação para o desenvolvimento do ambiente OwlNet. Então nós descrevemos a arquitetura e funcionalidades desse novo ambiente.

PALAVRAS CHAVES: ambientes para WBE, AulaNet™, LiveBOOKs, LivePAGE, modelos orientados a objeto.

[1] On leave from Universidade do Vale do Rio dos Sinos - UNISINOS

1. INTRODUCTION

Many learners, particularly adults, often operate under constraints placed on them by their work, family life, location, or other social and economic conditions. Thus, they often must study in an asynchronous learning mode, since they are not able to conduct their learning experience in a specific place at a specific time. In addition, they may not be able to access learning resources easily because of the same set of constraints. In spite of this need to learn when time is available, and with minimal access to learning resources, the learning experience should still be as rich and varied as possible.

An effective learning environment for the support of individuals, who must operate under the circumstances just described, should have several key characteristics. For example, several authors [3, 4, 17, 2] have concluded that learning should be an active experience; the learner should "learn-by-doing;" while Brown [5] has strongly suggested that placing the learner in an authentic setting can facilitate learning. Learning theory, verified by studies, has demonstrated that immediate and frequent feedback, cooperative learning and well-structured exposition of information can improve the learning process [2]. Other authors [12, 14] strongly believe that learners should take charge of their own learning.

"Learning-by-doing" is an approach that makes learning both active and authentic. The learner performs experiments with tools and artifacts representing the concepts to be learned; the results of these experiments suggest further experimental tasks. Obvious examples of learning-by-doing occur when learners work in science laboratories, solve mathematics problems, write and run computer programs, participate in case studies, or do research in the library. We often call learning-by-doing an interactive experiential approach, because learners are experimenting and interacting with educational materials.

The learner needs short-term and longer-term feedback through some form of validation or testing where understanding of basic concepts can be verified. Providing feedback can be achieved both by supporting interaction with the learning materials, testing frequently during the learning experience, and by testing at the end of learning modules through assignments and examinations.

Cooperative learning can be achieved by sharing learning experiences with "teachers" and "classmates," if they are part of a group with the same educational objectives. Sharing is an active experience where the learner takes charge by making the decision about what to share and with whom.

Learners need access to materials besides the ones prescribed directly for the course. They need to supplement their learning experience, and to acquire other viewpoints on the material being studied. In addition, a learner must have the ability to organize their learning process to be consistent with their own learning style.

The learning model just described is primarily constructivist in nature in that the focus is on the learner. Such an environment provides almost complete autonomy and freedom to the learner's thought process, since the learner can interact with the learning environment according to his/her cognitive style. We present here OwlNet, which is an WBE environment that was developed based on our experiences on previous environments [6, 9] in order to support the combination of the two approaches.

2. TWO COMPLEMENTARY APPROACHES TO WBE

In this section we present the AulaNet™ and LiveBOOKs environments. We focus here on the functionality and architecture of each one of the environments and justify their integration in a more powerful environment.

2.1 AulaNet

AulaNet™ [9] is a WBE environment developed in the Software Engineering Laboratory, at PUC-Rio. AulaNet™ results from our experience in teaching different courses using Web technology [15, 8, 16]. AulaNet™ is based on the following principles:

1. The courses must strongly emphasize interaction, making the student participate in the learning process ("learningware") [7].
2. The author does not need to be an Internet expert.
3. The courses must provide the same resources available in conventional classroom as well as the new resources available by the Web environment. The idea is to allow a smooth transition between the traditional and the new approach to teaching.
4. Reuse of existing material available in any digital form must be possible. For example, by importing existing files (file upload).

At the present moment there are about 30 courses being developed by professors from many different areas at PUC-Rio and in other Brazilian universities (http://les.inf.puc-rio.br/aulanet).

We present the AulaNet™ object-model through OMT diagrams and explain the concepts evolved.

AulaNet™ allows several institutions to use simultaneously the same environment. Each institution may have several departments. The courses are related to institutions and may be either composite or single. Courses are also classified by a degree of difficulty (course levels). Each course has actors and groups assigned to it. Groups are composed of actors and other groups. Each actor may either be a student, a teacher, or an author. The student is the one who attends a course, the author is the one who creates a course, and the teacher is the one who delivers a course. The students can also be assigned to a student level.

A course consists of a selection of resources. Teaching, evaluation, and administrative resources may be incorporated by the author to the course. Each resource may be mapped into one or more Internet services. As an example, we can use a teaching resource, which is a "Lesson Text", implemented by Internet services for text file uploads or just through URL references. Each service can also use one or more external tools, such as chat, CU-SeeMe, e-mail, and list servers. The whole design structure is presented in Figure 1.

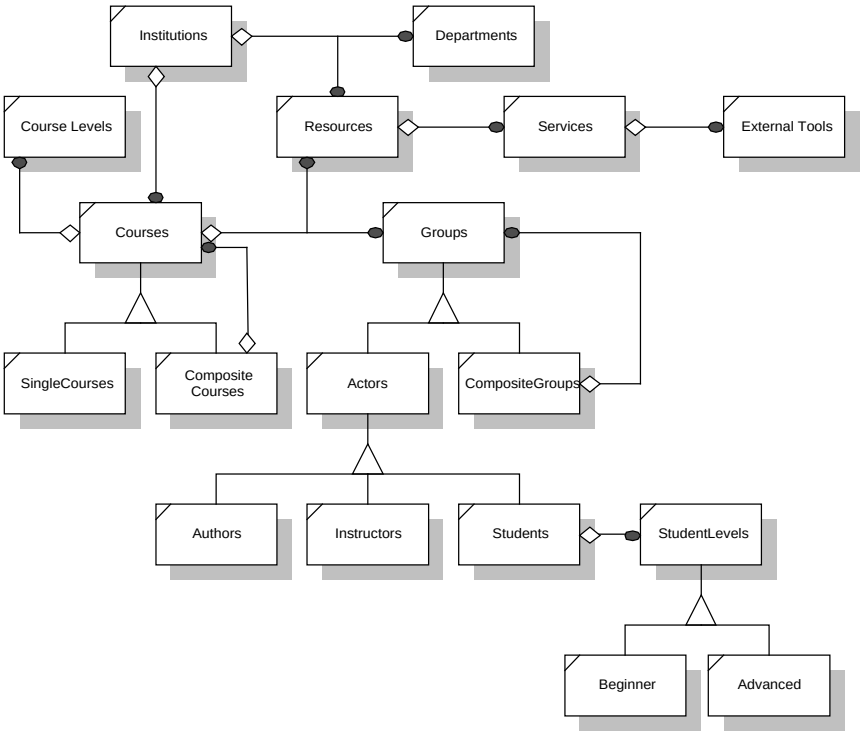

Figure 1. AulaNet OMT Class Diagram

In AulaNet™ the final user can modify the visual representation (Interface) for each one of its entities as shown in Figure 2.

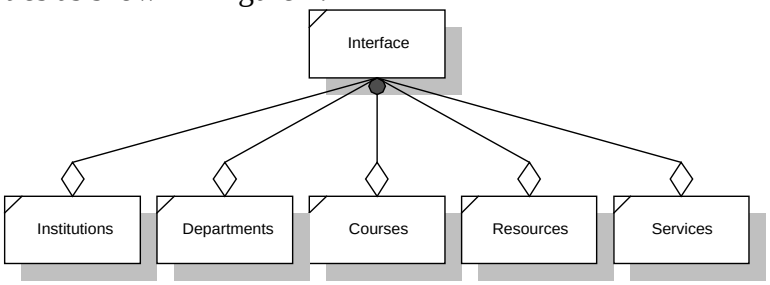

Figure 2. AulaNet Interface Class Structure

Each WBE environment is composed of two sites: a learning site and an authoring site. The learning site is used by the students to attend a specific course, while the authoring site is used by the authors to create and maintain the courses. Both can define many navigational structures. We are not going to consider these navigational aspects here. We will define the structure of both sites as hot-spots [11, 13] that are implemented by the classes Authoring Site and Learning Site.

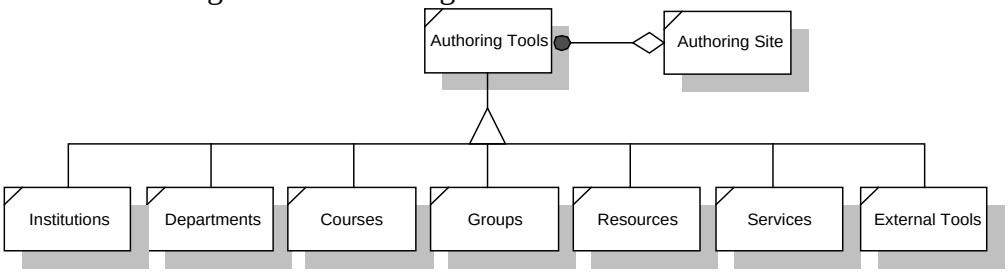

Figure 3. AulaNet Authoring Site

Figures 3 and 4 show the OMT models for the AulaNet™ authoring and learning sites, respectively. Notice that the learning site has to cope with multiple language

requirements, since a single institution may want to have their courses published in several languages.

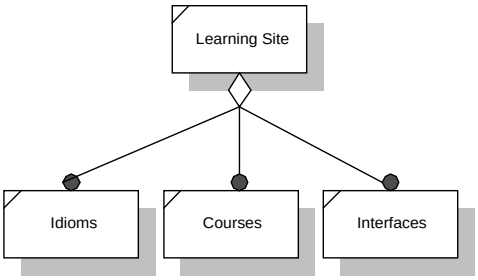

Figure 4. AulaNet LearningSite

2.2 LiveBOOKs

The LiveBOOKs [6] distributed learning/authoring environment is a computer-based teaching/learning and authoring system that has supports learning and authoring activities. It is designed for easy of use; the environment is easy for the student to use and also easy for the author of teaching materials to create those materials and include them in a specific LiveBOOK environment. We now describe the LiveBOOK class structure.

A LiveBOOK class consists of an aggregation of a LiveBOOK content class and this simple structure is illustrated in Figure 5. The LiveBOOK class has methods to support browsing, updating, generation of the Table of Contents (TOC), and providing the current location of the browsing operation in the LiveBOOK. This latter method is used by the LiveBOOK control class to determine which controls should be presented at any point in the LiveBOOK. In other words the object model supports context-sensitivity. The TOC method will be used by the user interface to display the Table of Contents when the LiveBOOK is displayed on the screen.

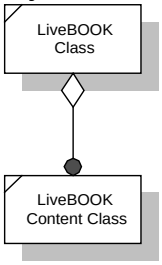

Figure 5. An OMT description of a LiveBOOK class

A LiveBOOK content class illustrated in Figure 6 is an aggregation of one or more LiveBOOK chapter classes.

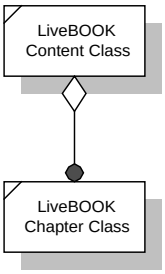

Figure 6. LiveBOOK Content Class Structure

A LiveBOOK Chapter class shown in Figure 7 is an aggregation of zero or more LiveBOOK Fragment classes and zero or more LiveBOOK Component classes. A

LiveBOOK Component class is an aggregation of zero or more LiveBOOK Fragment classes and zero or more LiveBOOK Section classes. This definition is recursive in that a LiveBOOK Section class is an aggregation of a LiveBOOK Component class. This relationship is also shown in a LiveBOOK Fragment class illustrated in Figure 8, where the different types of fragment classes inherit from the fragment base class. The fragment base class is an abstract class because the methods on the class depend on its specific type. For example, the display method would depend on the fragment being displayed on the screen. A fragment can consist of text, a binary large object (blob), an image map, a hyperlink source or hyperlink destination, and a hyperlink reference class. A blob is usually a graphic, a picture, or an audio or video clip. An image map is a graphic or picture with a map to allow accompanying hyperlinks. A hyperlink reference class activates an external program.

The user interface of the LiveBOOK is important because it controls the variability of the presentation and in fact the format and style of the educational material. There are many possible user interfaces. This LiveBOOK user interface consists of three windows: a contents Window, a Table of Contents (TOC) Window, and a Control Window. The Contents Window displays the contents of a Subsection and allows the user to interact with the individual LiveBOOK fragments in that subsection. The Control Window represents the various controls that are accessible when a specific Contents Window is displayed. In other words the controls are context sensitive.

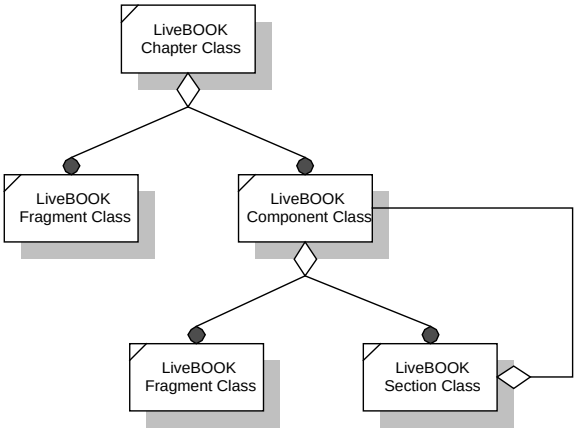

Figure 7. LiveBOOKs' Chapter Class OMT Description

This structure presented for LiveBOOKs may be seen as a service that can be added to the AulaNet™ environment. In fact, both systems are very complementary and can be combined as shown in Section 3.

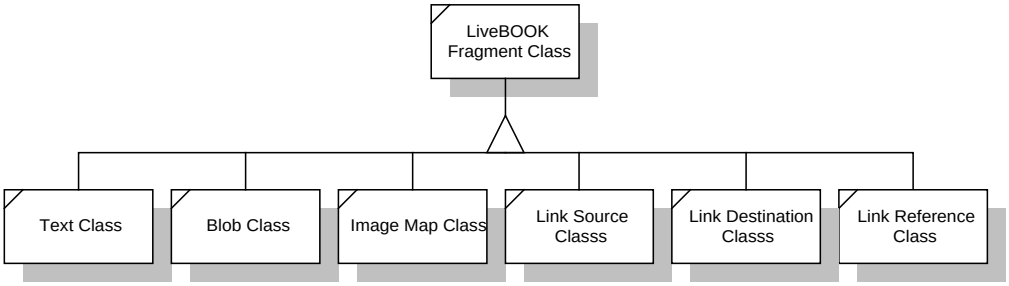

Figure 8. An OMT Description of a LiveBOOK Fragment Class

3. OWLNET ARCHITECTURE

The OwlNet environment supports the core functionality provided by AulaNet™ and LiveBOOKs. Its main concept is the one of roles. Five different roles are handled by the OwlNet environment, as follows:

1. Student: has access to a course web page (learning site), where he or she can browse the content of the course and use the available resources (e-mail, chat, list servers, conferencing). He or she can also have access to various LiveBOOKs related to the course. The Study-Mate is another service provided by the environment where the student can mark margin notes in a LiveBOOKs and perform self-evaluations.
2. Author/Editor: uses an AuthorKit that provides several features to support the course creation, such as HTML editors, organizer, file manager utilities, upload facility, and all the communication services such as e-mail and chat.
3. Instructor: is the one responsible for delivering the course. He or she also has access to the AuthorKit, besides some other services. These other services are: forms for FAQ and quizzes entry, register student facility, tool for creating student groups, and tool for recording student marks.
4. Communications Manager: is responsible for the control all the communication features, as wells as the registration of students.
5. Administrator: is responsible for all the administrative control, such as providing access to the system and setting read/write permissions.

The OwlNet design model is composed of the integration of the AulaNet™ and LiveBOOKs models. The final model is then adapted to support the concept of roles as defined above. The LiveBOOK Class structure is the same presented in Section 2.2 and is not repeated here. Figure 9 shows the OwlNet object model through an OMT Class diagram.

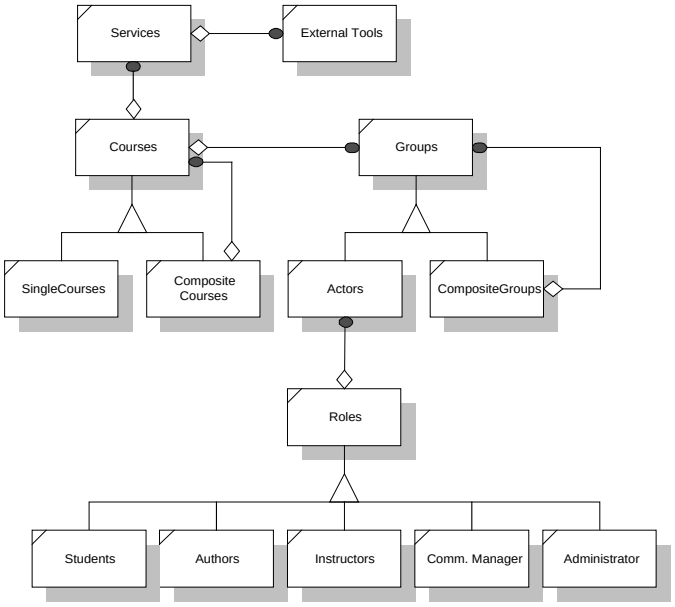

Figure 9. OwlNet OMT Class Diagram

4. IMPLEMENTATION

OwlNet will be developed based on the LivePAGE system, which is described in Section 4.1. Section 4.2 describes the approach we will use in the OwlNet development.

4.1 A brief description of the LivePAGE software

The LivePAGE software uses a relational database system to store multimedia documents. Figure 10 illustrates this implementation; the details are presented next.

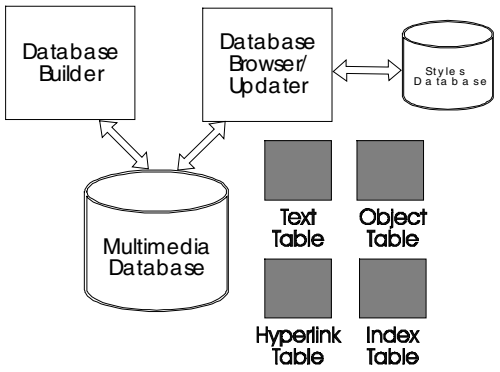

Figure 10. The Base Implementation

4.1.1 The base implementation

In the LivePAGE implementation we initially embed text, hyperlinks, image maps, and references to objects including graphics, audio, video and links to external programs into a single file, but distinguish or "separate" them by tagging conventions using an SGML-compliant tagging language. Thus, if we move to a different entity storage model, we will be able to separate the three types of information easily. Once the document is complete, we verify it against a grammar or DTD before loading the document into a single relational (SQL) database.

Each fundamental tagged is loaded into a single field in a relational database table and given a unique identifier. There are three separate tables for the text, objects and hyperlinks. In addition, every word in the document is also placed in an index to facilitate searching when the database is browsed. Optionally, every structure tag can be placed in the same index to facilitate searching for words within specific tagged structures. The various objects such as graphics, video and sound, and links to external programs are stored directly in the object table as "blobs" with the appropriate attributes. The various tables are identified in Figure 10.

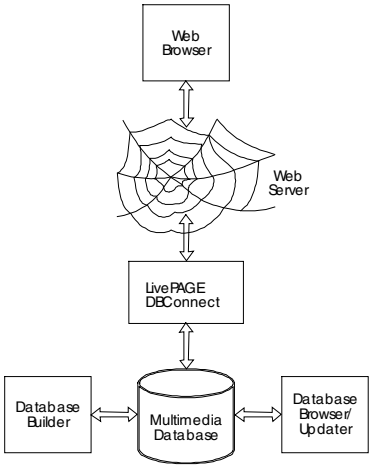

Figure 11. Access to the Database over the WWW

4.1.2 The toolkit

The LivePAGE toolkit is provided to create (the administrator), maintain (the updater), and browse and query (the browser) the database. Administrator functions, namely the verification against a DTD and creation of the database tables are described earlier. We consider documents as trees [10], and the updater allows a substructure (sub-tree) to be removed from the database and modified or replaced. While this substructure is being changed, the corresponding section of the database can be locked in order to maintain database integrity. Since we use SQL database technology, the database can be created, updated, and browsed using SQL statements. However, the LivePAGE tools provide an interface that makes the application of the SQL statements transparent.

The administrator and updater are primarily tools for the database administrator, while the browser is a tool for the general user to examine the database. The browser supports linear browsing, following hyperlinks forward and backward, and activating objects such as audio and video clips, or links to external programs. The browser also supports queries. The queries can be Boolean or free form where the results of the free-form query are presented in relevance order with the most relevant result presented first.

Text stored in the database can be extracted and modified using a structured text editor. Similarly objects stored in the database can be extracted using the updater and can be created or modified using appropriate authoring tools.

4.1.3 Presentation styles

Tools such as the browser and updater allow the client to view the document stored in the database. However, the document contains only structural information.

Presentation or style information is contained in a separate style database and is loaded into the browser or updater when it is invoked. The style database is indicated in Figure 10.

4.1.4 Connecting to the WWW

The base implementation described in Section 4.1.1 allows local access to documents but does not support access through the WWW. The LivePAGE toolkit contains two other mechanisms for this purpose.

4.1.5 dbConnect - producing a dynamic WWW site

Figure 11 illustrates the architecture of a dynamic distributed document database system accessible over the WWW. Users accessing a WWW browser such as Netscape or Microsoft Explorer can request a WWW page from a WWW server. Using the CGI, NSAPI or MSAPI protocol, the WWW server passes the request to the LivePAGE dbConnect module that then accesses a database of WWW pages. The specific WWW page that was requested is retrieved from the database and returned through the WWW server to the WWW browser for presentation. The style or appearance of each WWW page is governed by a Web Style File that is also an input to the dbConnect module. This file contains commands that govern how the content of the database is delivered to the browser. For example, a Table of Contents and navigation buttons can be generated automatically, thus relieving the author of the WWW site of creating navigational aids, and allowing users to orient themselves by returning to the TOC whenever they feel “lost in hyperspace”. The LivePAGE dbConnect module is not restricted to a single database, but can retrieve WWW pages from multiple document databases.

4.2 OwlNet Implementation Approach

The OwlNet environment will be implemented using the LivePAGE system to store its contents and control the navigational aspects. The object-model presented here, however, will be implemented in an additional database. The access to this database will also be handled by LivePAGE.

The use of the dbConnect interface will allows us to configure the environment’s interface through the use of the styles database (Figure 10). Figure 12 illustrates this implementation architecture.

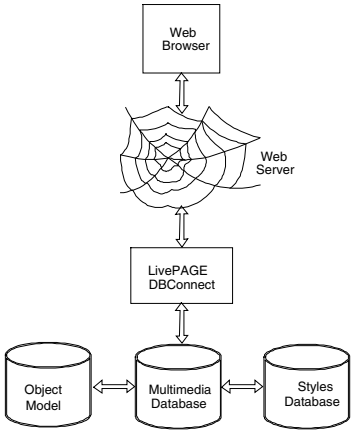

Figure 12. OwlNet Implementation Architecture

5. CONCLUSION

Based on our experience in developing WBE environments [6, 9] and on domain elicitation analysis [1] we could define a very generic architecture for this application domain. Based on this architecture we propose here the design and implementation approach of a new environment: OwlNet.

The complete architecture for OwlNet has been described and the implementation approach based on the LivePAGE software was proposed. The next step of our research is the actual implementation of this environment, followed by an evaluation period.

REFERENCES

1. P. Alencar, D. Cowan, S. Crespo, M. F. Fontoura, and C. J. Lucena, "Using Viewpoints to Derive a Conceptual Model for Web-Based Education Environments", MCC17/98, Monografias em Ciência da Computação, Departamento de Informática, PUC-Rio, 1998 (also submitted to ASE'98).
2. R. O. Briggs, V. Ramesh, N. C. Romano, and J Latimer, "The Exemplar Project: Using group support systems to improve the learning environment", Journal of Eductional Technology Systems, 23(3), pp. 277-291, 1995.
3. A. L. Brown, "Development schooling and the acquisition of knowing about knowledge", Schooling and the Acquisition of Knowledge, Hillsdale, N.J.: Lawrence Erlbaum Associates Inc., pp. 241-253, 1977.
4. A. L. Brown, "Design experiments: theoretical and methodological challenges in creating complex interventions in classroom settings", Design of the Learning Sciences, 2(2), pp. 141-178, 1992.
5. J. S. Brown, A. Collins, and P. Duguid, "Situated cognition and the culture of learning", Educational Researcher, 18(1), pp. 32-41, 1989.
6. D. Cowan, "An Object-Oriented Framework for LiveBOOKs", Technical Report, CS-98, University of Waterloo, Ontario, Canada, 1998.
7. Internet 2, <http://www.internet2.edu>.
8. ICC, <http://www.les.inf.puc-rio.br/icc> (in Portuguese).
9. C. Lucena, H. Fuks, R. Milidiu, L. Macedo, N. Santos, C. Laufer, M. Ribeiro, M. Fontoura, R. Noya, S. Crespo, V. Torres, L. Daflon, and L. Lukowiecki, "AulaNet™ - An Environment for the Development and Maintenance of Courses on the Web", in ICEE'98, 1998 (to appear).
10. E. Mackie and J. Zobel, "Retrieval of Tree-structured Data from Disc", Databases'92, Melbourne, Australia, February 1992. Third Australian Database Conference.
11. W. Pree, "Design Patterns for Object-Oriented Software Development", Addison-Wesley, 1995.
12. D. E. Rummelhart, "Schemata: the building blocks of cognition", in theoretical issues in reading comprehension, Erlbaum, Hillsdale New Jersey, pp 33-58, 1980.
13. H. A. Schmid, "Systematic Framework Design by Generalization", Communications of the ACM, Volume 40, Number 10, October 1997.
14. T. J. Shuel, "Cognitive conceptions of learning. Review of Educational Research", 56, pp. 411-436, 1987.
15. Sociedade da Informação, <http://www.les.inf.puc-rio.br/socinfo> (in Portuguese).
16. Transferência de Calor, <http://www.les.inf.puc-rio.br/transcal> (in Portuguese).
17. A. N. Whitehead, "The Aims of Education, New York: McMillan", 1929.