SADIO Electronic Journal of Informatics and Operations Research

http://www.dc.uba.ar/sadio/ejs

vol. 3, no. 1, pp. 13-22 (2000)

An object-oriented approach to Task Tree Management in the TANGOW system

Rosa María Carro[1] Estrella Pulido[1] Pilar Rodríguez[1]

[1]Escuela Técnica Superior de Ingeniería Informática,
Universidad Autónoma de Madrid
Campus de Cantoblanco, 28049 Madrid, Spain
E-Mail: {Rosa.Carro, Estrella.Pulido, Pilar.Rodriguez}@ii.uam.es

Abstract

This paper describes the object-oriented features of TANGOW (Task-based Adaptive learNer Guidance On the WWW), a tool for developing Internet-based courses. This system facilitates the construction of adaptive learning environments for the WWW and is able to guide the students during their learning process based on student profiles and previous actions. In the TANGOW system, the course contents is modelled in terms of objects and relationships among them. This allows the course designer to reuse the same descriptive objects in different sections of the same course, or even in completely different courses. In addition, information about the student and his/her actions when interacting with the system is also stored as dynamic objects, which are instantiated at runtime. This makes it easy to access and update student related data.

Keywords: Adaptive systems, Web-based training, Intelligent-tutoring systems, Dynamic course generation, Educational multimedia.

1 Introduction

Apart from other well known applications, the use of the WWW for distance education is becoming an important working area for researchers from a wide variety of fields such as psychology, sociology, and computing. Different systems for intelligent tutoring are currently available (Weber and Specht 1997) (de Bra and Calvi 1998) (da GraÇa et al. 1998). These kinds of systems use different approaches to structure course contents. We will make a brief review of those that are closer to the object-orientation paradigm.

In da Silva et al. (1998), the structure of the domain is based on concepts that are linked to documents and to other concepts through typed and weighted links. The student is guided towards appropriate documents based on information about his/her knowledge of each concept. Other formalisms such as the prerequisite graph model (Nykänen 1997) partition course contents into cells, where a cell is a single small topic to be studied. Relationships between cells are established by means of a prerequisite graph which states what topics should be known before studying further.

Another Internet-based teaching system that is currently in the design phase is SKILL (Neumann and Zirvas 98). In a way similar to the prerequisite graph model described above, in SKILL the course material is organized according to their prerequisites. A different approach is followed in the DCG system (Vassileva 1998) (Brown and Hansen 1998) where the concept structure of a course is represented as a road-map which is used to generate a plan for the course. The planner searches for sub-graphs that connect the concepts known by the learner with the goal-concept, and changes the plan if the student is not able to achieve a result higher than a given threshold score for a given concept.

In this paper we describe TANGOW (Task-based Adaptive learNer Guidance On the WWW), a system developed within the framework of the Interedu project, INTERnet in EDUcation (Alfonseca et al. 1998). TANGOW is an object-oriented system where not only the course contents are described in terms of objects and relationships, but also information about student features and about his/her actions when interacting with the system.

The paper is organized as follows. The architecture of the TANGOW system is presented in Section 2. Section 3 describes the objects, both static and dynamic, handled by TANGOW. Finally, in Section 4, we summarize the results presented in the paper and mention some further extensions currently being researched.

2 The TANGOW System

TANGOW is an object oriented tool for developing Internet-based courses, which facilitates the construction of adaptive learning environments for the Web (Carro et al. 1999a) (Carro et al. 1999b). It is able to guide the students during their learning process based on students profile and previous actions. Curriculum sequencing is generated dynamically, so that the same concepts may be taught in different ways, depending on student's profile and his/her actions while interacting with the course. By changing the learning strategy, even the same student may study the same concept in different ways. From the distance education point of view, the two main features of TANGOW are *adaptivity* and *dynamic page generation*. With adaptivity we refer to the fact that the same concepts can be taught in different ways depending on the student profile and student actions performed before. Dynamic generation implies that HTML pages are generated just before been presented to the students, and their concrete contents are selected at that moment, depending on the student profile.

As illustrated in figure 1, TANGOW architecture is based on the client-server paradigm of Internet. The Process Manager is a server that launches a Student Process for each student connected to the system and transfers it the student requests during the learning process. Each Student Process consists of a Task Manager that guides the students, selecting the next set of achievable teaching tasks at every moment, and a Page Generator that builds the HTML pages presented to the student.

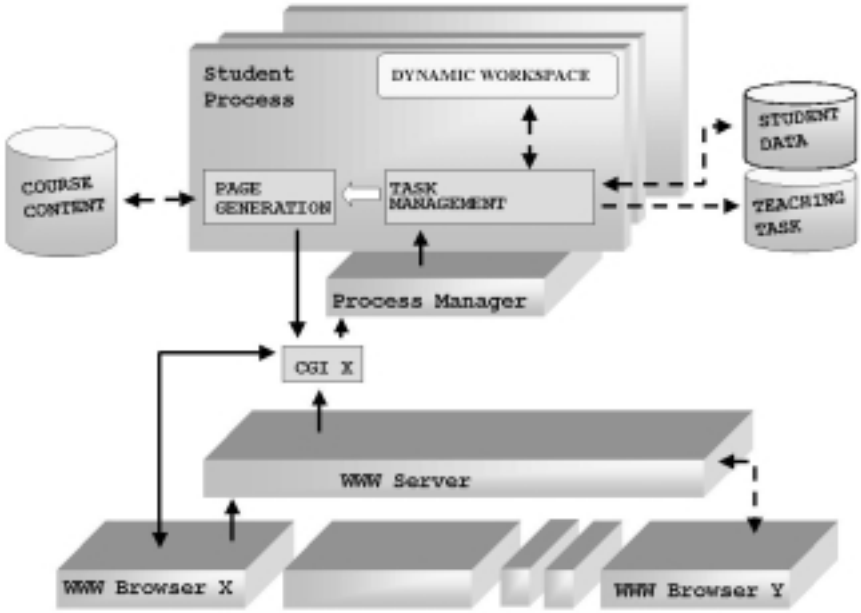

Fig. 1. TANGOW Architecture

In TANGOW, a course description is based on the instantiation of two main object classes: Teaching Tasks (TT) and rules. A teaching task is the basic unit in the learning process, and a rule describes how a task is decomposed into subtasks. Teaching tasks and rules models are stored as static objects in the Teaching Task Repository, while specific instances of these models are generated at runtime and are different for each student depending on his/her previous actions and profile.

Information about the actions performed by the student when interacting with the course is stored in the Dynamic Workspace. This information is used by TANGOW to adapt the course contents to the student's learning progress. TANGOW has also information about student profiles, which is used to select, at run-time, the contents of each HTML page presented.

Student processes are implemented in Java (Eckel 1998), while the database objects are stored and managed by using JDBC (Hamilton and Cattell 1996).

3 Objects in TANGOW

All the information handled by the TANGOW system is represented as objects and relationships. Static objects store information about tasks, rules, multimedia elements and student profiles, while dynamic objects are related to the student's actions and the concrete HTML pages generated at runtime. An object-oriented model of the information handled by the TANGOW system, including the main static and dynamic classes, is illustrated in figure 2. The representation formalism used is based on the Unified Modeling Language (Eriksson and Penker 1998).

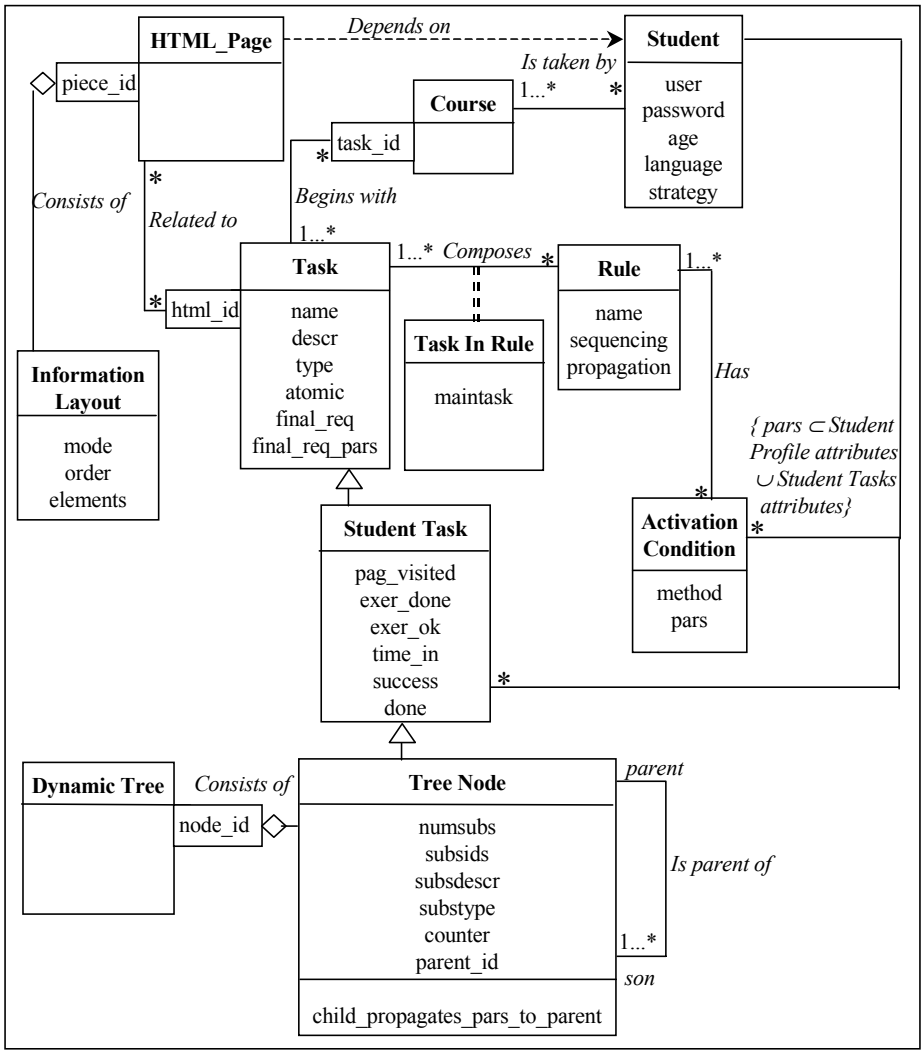

Fig. 2. UML diagram of the information handled by TANGOW

3.1 Static Objects

Static objects are those whose instances do not change during the learning process. This is the case of the objects that the designer has to define when creating a new course, such as Task, Rule or Information_Layout. The Task_In_Rule object appears from the relation between a Task and a Rule, so that a Task_In_Rule object represents a specific task appearing at a given rule. Another static object is Student, which stores the student profile at the beginning of his/her first session. The student profile is used during the learning process to adapt the course contents to each student.

Tasks & Rules

When creating a new course, the course designer must specify the different teaching tasks that compose the course and define the relations between them. Task decomposition is represented by means of rules, where

each rule describes the way a task is divided into subtasks. There may be several rules for the same TT, each of them representing a different way of decomposing it. Tasks and rules descriptions are objects stored in the Teaching Task Repository.

A Task object is described in terms of the following attributes:

name: the task name.

descr: text that describes the task.

type: distinguishes between theoretical (T), practical (P) and example tasks (E).

atomic: indicates if the task is atomic or composed.

final_req: method that decides whether the task is finished. This decision is based on parameters directly related to student actions (final_req_pars), in the case of atomic tasks, or on subtask finalisation, if the task is composed.

html_ids: optional list of pieces for HTML generation.

A Rule object has the following attributes:

name: name of the rule.

sequencing: order in which subtasks must be tackled. XOR indicates that only one of the subtasks must be performed, OR means that at least one of the subtasks must be performed, AND indicates that all the subtasks must be performed in the order they appear in the list, and ANY indicates that all the subtasks must be performed, but that they can be performed in any order.

propagation: indicates how parameter values for the task in the left-hand side of the rule are calculated in terms of parameter values of the subtasks in its right-hand side.

activation condition: optional list of preconditions that have to be satisfied for initiating the task in the left-hand side of the rule. It may depend on information about the tasks already achieved, the student's profile and/or the learning strategy in use.

The Task_In_Rule object has a single attribute that indicates the relation between a task and its associated subtasks:

maintask: 'Y' indicates that the corresponding task appears in the left-hand side of the rule, whereas 'N' indicates that it appears in its right-hand side.

Information Layout

As explained above, HTML pages are generated dynamically starting from the descriptions of several pieces that compose the pages related to the task that is being performed. Each piece can be reused in different tasks and even in different courses. The specific elements included in the pages will depend on information about student profile, and are selected 'on the fly' just before presenting the pages.

Information_Layout objects are descriptions stored in the Course Content Repository and have the following attributes:

mode: the layout of the media elements that appear in this document component. There exists a description language which specifies the relative positions of the media elements in the piece and is used to construct the HTML pages that will be presented to the students.

order: position of the piece in the page (the first piece, the second one, ...)

elements: list of file names containing texts, images, sounds, videos, animations and applets that appear in this piece. If any of the media elements corresponds to an exercise, the correct answer is also given.

The multimedia elements are stored and classified into different directories, depending on their nature. The course designer can decide aspects (such as the level of difficulty, language, ...) which may differentiate some elements from others related to the same task and corresponding to the same 'virtual' name. For example, a concept can be explained by means of texts that make use of vocabulary of different difficulty. This is useful for adapting course contents to the student.

Student Profiles

The students are asked for personal data the first time they enter the system. This information is stored at the beginning of the first session and restored each time the student accesses the system. The Student object includes information about:

user: the name of the student that is launching the system.

strategy: the preferred strategy for the learning process.

password, age and language.

This student profile is used for adapting the course to the student by checking the rules that describe the next achievable task decomposition and whose preconditions depend on student data, and selecting those whose activation conditions are satisfied. It is also used to select the concrete media elements that will appear in the HTML pages presented to the student.

3.2 Dynamic Objects

Dynamic objects are those instantiated during the process of learning. At execution time, all teaching tasks performed by the student are stored in her/his dynamic workspace. There, information about the sequence of tasks already initiated by the students is kept. The Teaching Tasks instances are created dynamically, depending on the student actions and profile, and this information is enriched by the addition of dynamic attributes that are related to student actions on specific tasks. Let us see more details about these dynamic objects, which are: HTML_Page, Student_Task, Dynamic_Tree and Tree_Node.

A Student_Task object has the same slots as a TT, plus those reflecting the student's actions while learning it:

time_in: the time the student has spent learning this task.
pag_visited: the number of visited pages.
exer_done: the number of exercises done (in the case of a practical task).
exer_ok: the number of exercises correctly solved (in the case of a practical task).
success: a percentage representing the grade of learning achieved calculated by means of a function that makes use of the above mentioned dynamic parameters.
done: a flag that indicates whether the task is finished.

Dynamic task trees

All the information about the student tasks during the learning sessions is stored as a Dynamic_Tree, in which a node corresponds to a Student_Task plus information about the relationships with other nodes in the tree (see figure 3):

numsubs: the number of subtasks in which the task is decomposed.
subsids, subsdescr, substype: information about the subtasks (identifier, description and type).
counter: the number of subtasks already initiated (that is, the number of children for this node).
parent_id: the node's parent in the tree.
subtasks_ids: pointers to the nodes corresponding to the subtasks already initiated.
child_propagate_pars_to_parent: method used after task execution for propagating its parameter values to its parent.

When a student finalizes a learning session his/her dynamic tree is updated and stored in the Student Data Repository.

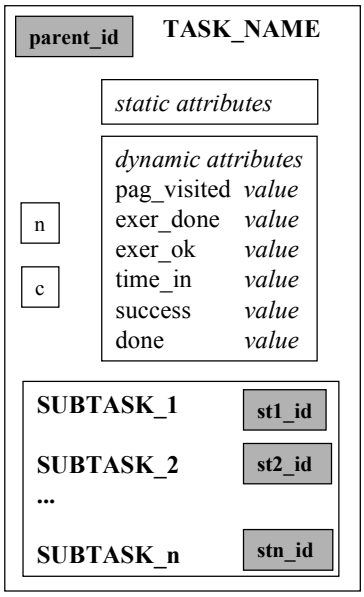

Fig. 3. A generic dynamic tree node

Dynamic Tree Snapshots

As stated above, task trees are dynamic since they can adopt different shapes at runtime, depending on students' profiles and actions. In this section two dynamic trees are analyzed, whose different shapes arise from the fact that they correspond to two different student profiles. The course they are related to deals with traffic signs, and course designers decided that the student's age, one of the previously defined static attributes, is significant when introducing some of the subjects covered by the course. In this sense, it was decided that students aged less than eighteen should be presented with examples in addition to the pure theory. This is the case of the sign priority concept, as appearing in the dynamic trees of figures 4 and 5. In both figures, the root of the tree is the same task, TRAFFIC_SIGNS, which corresponds to the initial task of the course. That is the reason why no *parent_id* arrow is drawn out of that task.

In figure 4 the TRAFFIC_SIGNS task is decomposed into two subtasks, namely: TYPES and PRIORITY. This decomposition is due to the fact that the snapshot corresponds to a student younger than eighteen. The TYPES task has not been performed yet, while the PRIORITY one is completed (see their subtask counters that record 0 and 2, respectively). The two subtasks corresponding to the PRIORITY task, PRIORITY_THEORY and PRIORITY_EXAMPLES, have already been performed and are both atomic.

With respect to attribute propagation, it is performed in a bottom-up way, mostly by adding all subtask values to the own parent task ones. An example is the *time_in* attribute, which propagates upwards, including the time spent by the student in the parent task and its descendents. *Pag_visited*, *exer_done* and *exer_ok* values are propagated in a similar way, while *success* and *done* values are propagated by using different estimation methods.

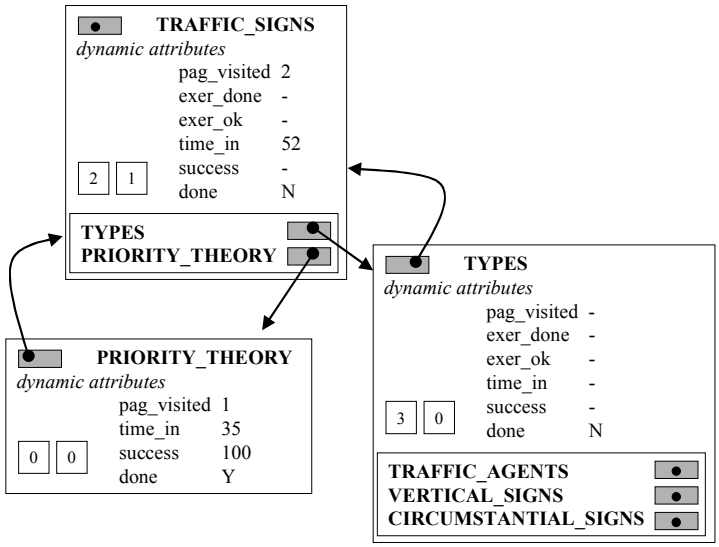

Fig. 4. Dynamic Tree Snapshots: Example 1

On the other hand, figure 5 shows the initial dynamic tree for a student older than eighteen. In this case, the course designers have not considered examples about sign priority essential. This is the reason why PRIORITY_THEORY appears directly as a subtask of the TRAFFIC_SIGNS task, at the same level as the TYPES subtask, which is not detailed in the figure. The corresponding dynamic attributes are propagated accordingly, as can be seen in the figure.

These examples refer to a course that can be found at http://helena.ii.uam.es/html/courses.html.

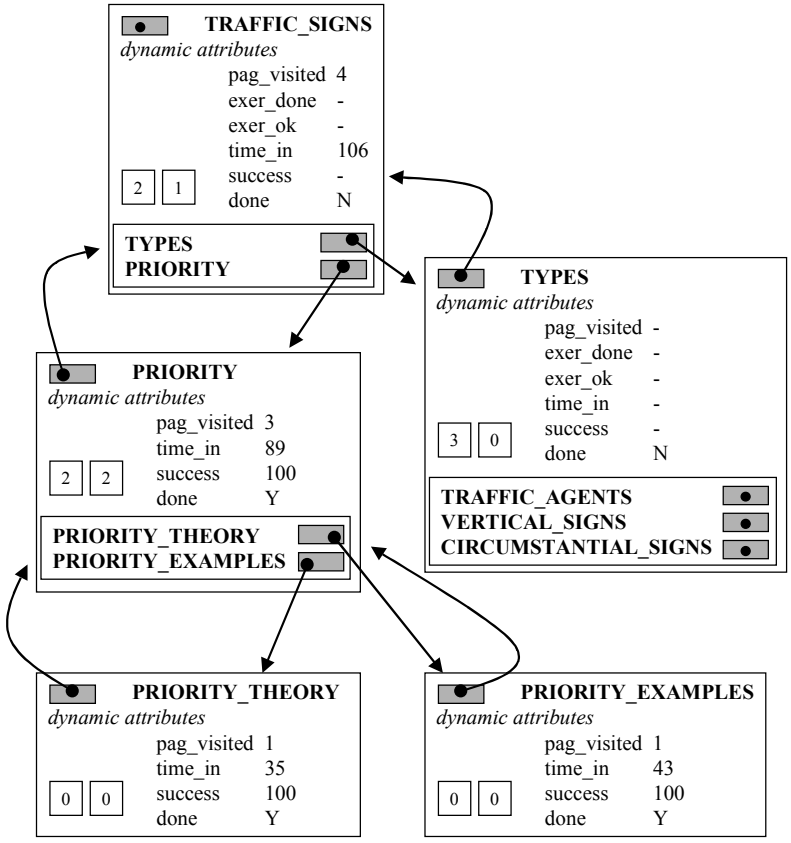

Fig. 5. Dynamic Tree Snapshots: Example 2

4 Conclusions

The object-oriented features of the TANGOW system allow course designers to develop adaptive learning environments for the WWW by using tasks and rules objects to describe the courses. These tasks and rules are used at execution time to guide the students during their learning process, so that they will be presented with different HTML pages depending on their profile, their previous actions, and the active learning strategy.

TANGOW is written in Java and is widely accessible through Internet by using any standard Web browser. Thanks to the storage of tasks, rules and multimedia objects in databases, the cost of course maintenance is low since designers may change, add or remove course components easily. The use of databases also allows the reuse of components in different courses. Currently we are working on a course designer object oriented interface for making the course development process easier.

In the near future, we intend to extend the TANGOW system with collaborative work facilities, essentially by establishing communication among student processes. Similar approaches have already being successfully followed by other systems (Verdejo et al. 1998) (Schlichter 1997).

5 Acknowledgements

This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TEL97-0306.

References

Alfonseca, M., Carro, R.M., Moriyón, R., Pulido, E., Rodríguez P., Sigüenza J.A.: InterEdu: Internet en la Educación. Proceedings of II Congreso Nacional de Ingeniería de Telecomunicación, Madrid, Spain, Junio (1998) 321-325

Brown, A., Hansen, L.: Software Tools for Distance Learning: The Benchmarks Project. Proceedings of Society for Information Technology and Teacher Education SITE'98 , Washington DC, March 10-14 (1998).

Carro, R.M., Pulido, E., Rodríguez, P.: Task-based Adaptive learNer Guidance On the WWW: the TANGOW System. Second Workshop on Adaptive Systems and User Modeling on the Web at the Eigth International World Wide Web Conference, Toronto, Canada, May 11-14 (1999a)

Carro, R.M., Moriyón, R., Pulido, E., Rodríguez, P.: Teaching Tasks in an Adaptive Learning Environment. The 8th International Conference on Human-Computer Interaction, Munich, Germany, August 22-27 (1999b), forthcoming

da GraÇa, M., Benedito, J., Pontin, R.: Tools for Authoring and Presenting Structured Teaching Material in the WWW. Proceedings of WebNet 98 World Conference of the WWW, Internet & Intranet, Orlando, Florida, November 7-12 (1998) 194-199

da Silva, D.P., Van Durm, R., Duval, E., Olivi, H.: Concepts and documents for adaptive educational hypermedia: a model and a prototype. Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia at the Ninth ACM Conference on Hypertext and Hypermedia, Pittsburgh, USA, June 20-24 (1998) 35-43

de Bra, P., Calvi, L.: AHA: A Generic Adaptive Hypermedia System. Proceedings of the Second Workshop on Adaptive Hypertext and Hypermedia at the Ninth ACM Conference on Hypertext and Hypermedia, Pittsburgh, USA, June 20-24 (1998) 5-11

Eckel, B.: Thinking in Java. Prentice Hall PTR, Prentice-Hall Inc., A Simon & Schuster Company. Upper Saddle River, New Jersey 07458 (1998) http://www.phptr.com

Eriksson, H., Penker, M.: UML Toolkit. Wiley Computer Publishing, John Wiley & Sons, INC., (1998)

Hamilton, G., Cattell, R.(ed.): JavaSoft. JDBC™: A Java SQL API. Sun Microsystems Inc., Graham Hamilton & Rick Cattell , 2550 Garcia Avenue, Mountain View, CA 94043 (1996)

Neumann, G., Zirvas, J.: SKILL: A Scalable Internet-Based Teaching and Learning System. Proceedings of WebNet 98 World Conference of the WWW, Internet & Intranet, Orlando, Florida, November 7-12 (1998) 688-693

Nykänen, O.: User Modeling in WWW with Prerequisite Graph Model. Proceedings of the workshop Adaptive Systems and User Modeling on the World Wide Web. The Sixth International Conference on User Modeling, Chia Laguna, Sardinia, June 2-5 (1997)

Schlichter, J., Koch, M., Bürger, M.: Workspace Awareness for Distributed Teams. Proc. Coordination Technology for Collaborative Applications - Organizations, Processes, and Agents, Singapore, Lecture Notes on Computer Science 1364, W. Conen, G. Neumann (eds.), Springer Verlag, Berlin (1997) 199 - 218

Vassileva, J.: A Task-Centred Approach for User Modeling in a Hypermedia Office Documentation System. In Brusilovsky, P., Kobsa, A. and Vassileva J. (Eds.) Adaptive Hypertext and Hypermedia, Kluwer Academic Publ. Dordrecht, Chapter 8, (1998) 209-247

Verdejo, M.F., Barros, B., Abad, M.T.: Supporting Distance Learners for Collaborative Problem Solving. Published in ED_MEDIA & ED_TELECOM 98 (Edited by Thomas Ottman & Ivan TomeK), AACE, Freiburg, Germany, June 20-25 (1998) 1407-1412

Weber, G., Specht, M.: User modeling and adaptive navigation support in WWW-based tutoring systems. Proceedings of User Modeling '97,. Italy, June (1997) 289-300