Distributed intelligent tutoring on the Web

Peter Brusilovsky
School of Computer Science, Carnegie Mellon University,
Pittsburgh, PA 15213 USA
E-mail: plb@cs.cmu.edu

Steven Ritter
Department of Psychology, Carnegie Mellon University,
Pittsburgh, PA 15213 USA
E-mail: sritter@cmu.edu

Elmar Schwarz
Department of Psychology, Carnegie Mellon University,
Pittsburgh, PA 15213 USA
E-mail: schw1302@uni-trier.de

Abstract: The World-Wide Web is an inherently distributed environment. As the web becomes a more important medium for education, we need to consider how different educational systems might interact. Adaptive systems are especially promising in this regard, since they offer the potential to customize themselves to suit students who have not yet used the system. In this paper, we describe our efforts to construct an integrated system which incorporates conceptual instruction through InterBook and problem-solving interaction through PAT Online. InterBook and PAT Online are two separate Web-based adaptive tutoring systems. To achieve a better level of adaptivity these systems interact and exchange information about the user's progress.

Introduction

As the World-Wide Web (WWW) becomes a more important educational medium, it is essential that authors of web-based educational applications consider making their systems more adaptive and intelligent. In the classroom, a teacher can compensate for the fact that software is unadaptive, but web-based educational applications are often used by very different classes of users without the assistance of a human teacher. Until very recently all educational applications on the WWW were non-adaptive. Now, there are several examples of more-or-less adaptive Web-based tutoring systems: CALAT [16], ELM-ART [4], PAT-OnLine, Albatros [12], and InterBook [5]. Another emerging class of adaptive Web-based applications which could be used for education is adaptive information systems such as ILEX or PEBA [14]. The key to adaptivity in all systems mentioned above is a student model (also called a user model). For each user of the system, the student model maintains up-to-date information about his or her goals, knowledge, background, etc. The more precise and correct the student model is, the more advanced the types of adaptation that can be supported.

We expect that many more adaptive Web-based systems will be developed in the coming years. Moreover, reasonably soon a course teacher will have a selection of different adaptive systems which can be used to teach the same subject. In many cases, these systems complement each other

well and it would make sense to combine them. For example, an InterBook-like system could support conceptual learning, while a PAT OnLine-like system could support problem solving in the same domain. Ideally, a teacher should be able to choose the systems which will be used together. A real integration (as opposed to just an aggregation) of several Web-based systems would result in a versatile virtual system which is "more than sum of its parts".

In our previous work [2; 21], we considered two aspects of real integration of ITS components. First, one component should be able to use the capabilities of another component and exchange or share data with it. Second, the results of students' work with any of the components during the session should be taken into account by other components so that they can adapt their performance to the knowledge level and personal features of the particular student. For example, if a student learned something new while working with an information system, a related multimedia tutoring system should take this into account and avoid re-teaching those already-learned concepts.

This paper address the integration problem in the context of the WWW, where each component is a separate Web-based adaptive system. In the next section we present possible ways to organize interaction and student model-sharing between different Web-based adaptive systems. The remainder of the paper presents an example of an integrated tutoring system. This example demonstrates a way to organize cooperation between two different adaptive tutoring systems, PAT OnLine and InterBook [5], which use basic algebra as their target domain. The integrated system is able to deliver conceptual information through InterBook and support interactive problem solving using PAT Online (Figure 1). Two sections are devoted to describing the basic goals and architectures of these two systems to the extent required to understand the problems of setting up an interaction. Following that, we describe the modifications required to be made to the systems in order to support interaction and the protocol that we chose for interaction.

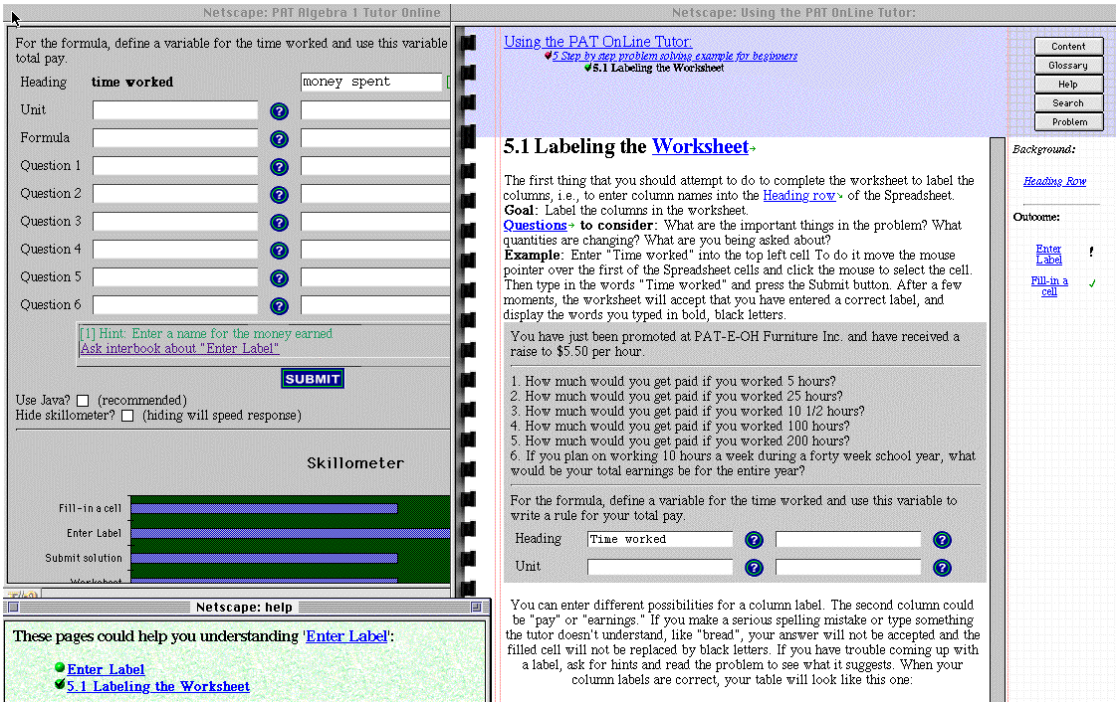

Figure 1: The Pat Online/InterBook environment. The top left window shows the PAT Online problem-solving interface in a situation where the student has to enter the worksheet label . The bottom left window is the InterBook Help window suggesting a list of manual sections which explain how to enter the label. The right window is the InterBook "Textbook window" showing one of these helpful sections.

Architectures for Distributed Adaptive Tutoring

Three different models of student model sharing between different Web-based adaptive systems are master-slave, communicating peers, and a centralized architecture.

The master-slave model is the simplest model which could be applied if one of the cooperating systems was not specially developed to support inter-system communication. If a system is adaptive and has an open user model (i.e. the student could use a kind of form-based interface to see and update the content of the user model), a specially designed master system will be able to use it as a slave. In this architecture, the master system could pre-fetch information from the slave system and show it in its own window or frame. Sharing student models in this system is simple, since the master system can update the user model using the same form-based interface as a student would. A disadvantage of this architecture is that it is asymmetrical. The student must always interact directly with the master, since student model changes that result from direct interaction with the slave will not be communicated to the master system.

A more promising architecture is communicating peers. Peers are systems which know about each other and can use special protocols to communicate. They can request information from each other, inform each other about changes in local user models, and ask each other to perform some actions. Perhaps the simplest way for the peers to exchange information is using structured URLs, since many toolkits for building Web-based systems provide facilities for generating and parsing structured URLs. The problem with this architecture is that all peers communicate directly with each other. As more peers are integrated into the system, it becomes harder to organize the communication. When a new peer is added into the system, all other peers must be informed, complicating the process.

A more flexible and open architecture which is able to integrate many independent components is a centralized architecture. The centralized architecture is based on a special user-modeling server which stores all information about a particular student. Any adaptive system working with the student is expected to get up-to-date information about the student from the server and to inform the server about any changes in its assessment of the student's knowledge, goals, etc. One benefit of this architecture is that a new system which understands the protocols of the user modeling server could easily become a part of integrated system. None of the already-existing systems would need to change their behavior because of the newly-added system.

A number of existing projects represent a good background for research on integration-oriented architectures. Some aspects of inter-component communication were addressed by the work on blackboard architectures for ITS [13] and more recently by the original work of members of the French-speaking research community on multi-agent architectures for ITS [8; 9]. A multi-agent ITS is a set of independent components, such as a problem-solver, environment, or instructional planer, which may have local knowledge and which communicate by exchanging messages. The research on multi-agent architectures also investigates the problems of generic agents and centralized student modeling [17]. This research could be helpful for developing a "communicating peers" architecture.

Another group of projects [21; 22] provide background for research on a centralized architecture. They suggest that flexible communication between components can be based on some inter-application protocol such as Apple Events or DDE. Two aspects of this architecture are an application-independent semantic-level communication language and a control center which collects and delivers all inter-component messages. The topic of centralized student modeling is addressed by research on student modeling architectures [1] and user modeling shells, such as TAGUS [18; 19] and BGP-MS [10]. Central student models and user modeling shells are designed to collect information about the user from different sources and to serve queries about the user from different applications. The recent implementation of the BGP-MS user modeling shell can already communicates with its clients using Internet and TCP/IP [7].

The PAT Online Tutor

Learning Goals and Assessment

The PAT Online tutor is a variant of the PAT algebra 1 tutor described in [11]. The tutor assists students in solving word problems described by one or two linear equations. Students solve these problems by completing a spreadsheet and graph describing the problem situation and by solving equations related to the problem situation. The PAT Online tutor currently implements the spreadsheet and a portion of the graphing module.

As students work through a problem, the tutor provides help, identifies errors and updates its assessment of the student's progress. Each of the rules in the underlying production system represents a skill that the student needs to master in order to solve the problem. As described in [6], the tutor assumes a simple two-state model of skill learning. Each skill is either mastered or not, and the tutor maintains, for each student, the probability that the student has mastered the skill. At each opportunity to learn (that is, each time the student encounters a situation where that skill is needed), there is some probability, *pLearn*, that the skill will transition from the unlearned to the learned state. Two other parameters estimate the probability that the student will make an error even though the skill has been mastered (i.e. the student "slips") and the probability that the student will give the correct answer even though the skill has not been mastered (i.e. the student guesses correctly). Equations 1 and 2 describe the method for updating skill probabilities based on student performance.

If a student is successful:

$$(1a)\ knownAndSucceeded = \frac{pKnown_{T-1} * (1 - pSlip)}{pKnown_{T-1} * (1 - pSlip) + pGuess * (1 - pKnown_{T-1})}$$

$$(1b)\ pKnown_T = knownAndSucceeded + pLearn * (1 - knownAndSucceeded)$$

If a student is unsuccessful:

$$(2a)\ knownButFailed = \frac{pKnown_{T-1} * pSlip}{pKnown_{T-1} * pSlip + (1 - pKnown_{T-1}) * (1 - pGuess)}$$

$$(2b)\ pKnown_T = knownButFailed + pLearn * (1 - knownButFailed)$$

System Architecture

PAT Online differs architecturally from the tutor used in the classroom. It was designed in accordance with the plug-in tutor agent architecture described in [20; 21]. The tutor agent architecture separates the instructional knowledge from the "tools" that the student uses to interact with the system. While the agent architecture was not designed the WWW in mind, the architecture has proved to be easily adaptable to delivery over the web. The tutor agent used in PAT Online was originally developed for use with a single-user, desktop system. Transforming the "translator" of the plug-in architecture into a CGI program allowed us to use the tutor agent from the existing system as the basis for PAT Online without modification. One small modification was added to the tutor agent to facilitate interaction between PAT Online and Interbook.

Figure 2 illustrates the architecture of PAT Online. The tutor agent contains all of the tutoring knowledge for the system. It is responsible for providing help, identifying errors and assessing student performance on the component skills of the task. The student record server maintains all information about students who have accessed the web site and acts as a CGI program, allowing students to access (and change, if authorized) information contained in their student files. The translator maps student actions into messages for the tutor agent and tutor agent messages into output. The tutor CGI parses information that the student enters into an HTML form and re-constructs a new form based on feedback from the tutor.

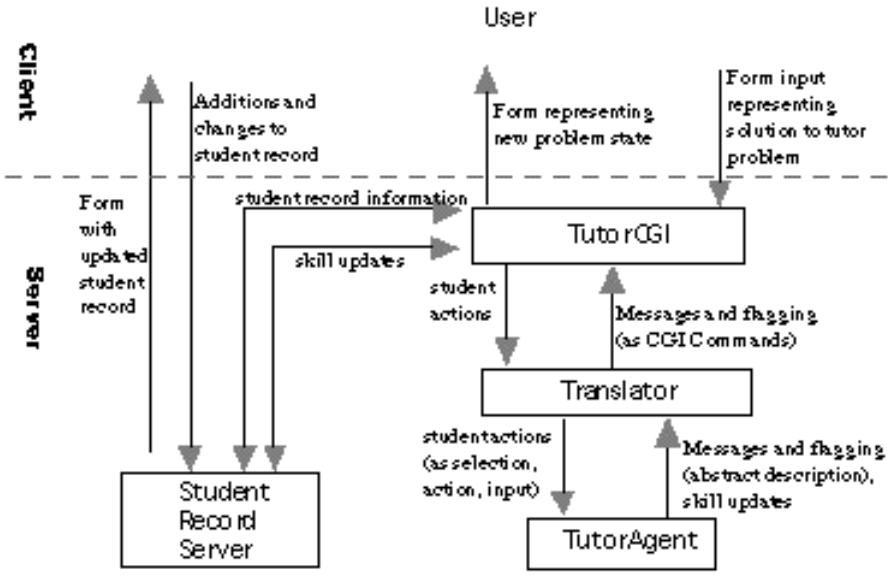

Figure 2: System architecture of PAT-Online**.**

InterBook - A User-Adapted Textbook

InterBook is an authoring and delivery tool for user-adapted textbooks on the WWW [5]. InterBook strives to provide authors with a simple authoring tool for their educational hyper- and multimedia material and to present that material in an efficient and easy-to-use user-adapted interface on the WWW. To simplify the use of InterBook, it employs technologies like incremental, user-adaptable and -adaptive knowledge tracking and adaptive navigation support [3].

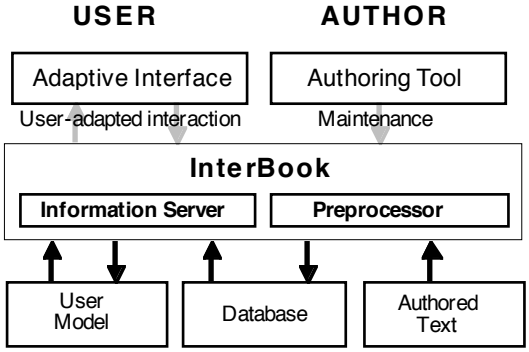

Figure 3: Structure of InterBook as an authoring and delivery tool.
The authored educational material is preprocessed by InterBook and transformed into a database. Material from this database is adaptively presented to the user according to his or her user model.

The backbone of InterBook's open architecture is its user model, which is able to store any arbitrary data about a student that might support the system in teaching this particular student. The main part stores the student's current state of knowledge. InterBook uses an extended overlay model, which is dynamically extensible to be suitable to any educational domain. Information about the student is gained by tracking user actions like reading text, looking at examples or solving multiple choice tests. This extends the simple approach of an overlay model to a multidimensional space which describes the user's current state of knowledge about a certain concept by a score on any of these dimensions. These multiple scores are finally projected into a scalar value by a simple linear expression:

$$know(c) = \frac{\sum_{scores\ available} \left(weight_{score} \cdot value_{score}(c) - threshold_{score} \right)}{\#\ available \quad scores}$$

Equation 3: The scalar value of the supposed state of knowledge of a concept *c*.

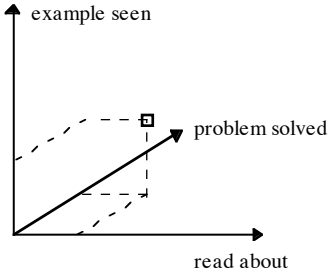

Figure 4: Extensible multidimensional overlay model in InterBook with three scores.

The expression results in a score that can assume any arbitrary value from negative to positive infinity. Each of these scores are used to estimate the educational state of any concept and teaching operation. InterBook employs the user model for user-adapted interaction such as adaptive link annotation, prerequisite-based help, and adaptive sorting [3]. This strong user-adapted interface enables InterBook to support the student in choosing the next relevant teaching operation at any point of his or her state of learning. The quality of this support is limited by the accuracy of user tracking and the derived knowledge score. InterBook lacks any mechanism to track a student's state of knowledge during problem solving, which is often considered to be the most reliable source of user modeling. The solution to this problem is to extend the system so that problem-solving systems can communicate information about a student's knowledge to InterBook.

Interaction between PAT Online and InterBook

PAT Online and InterBook are good examples of adaptive systems which could complement each other when used in parallel for teaching the same subject (such as high-school algebra). PAT Online can support problem solving activity and track the student's procedural knowledge. InterBook can support learning from an electronic textbook or manual and track the student's conceptual knowledge. Together PAT Online and InterBook could provide a complete adaptive learning environment. It depends on an ability to communicate and to exchange their knowledge about the student. PAT Online has to know what conceptual information the student has learned with InterBook. InterBook has to know which knowledge concepts has been applied successfully (or not) in the problem-solving activity. This section describes how PAT Online and InterBook interact with each other when they are used together as an integrated environment.

Communicating updates

The interaction between PAT Online and InterBook is a peer-communication architecture that supports two kinds of information sharing (see Figure 5). First, the tutor and interbook share assessment information about students. When a student works with PAT Online, any changes in PAT's assessment of the students skills are communicated to Interbook. Similarly, when a student works with InterBook, information about the concepts explained on pages the student reads are sent to the PAT Online tutor. In addition, when the student asks the PAT Online tutor for a hint (or when PAT Online presents a bug message), the tutor includes a link to an InterBook page explaining the underlying skill. Similarly, students using InterBook can follow a link directly to an appropriate problem in PAT Online.

The design of the interaction between the two systems was influenced by two goals. First, we wanted to make sure that nothing in the design of either system assumed that students had access to the other. This is necessary because, unlike in a classroom version of the tutor, on the world-wide web, we have little control over how students come into the tutoring environment. Thus, it is important that we support students who come directly to PAT Online or directly to InterBook. Our

second design goal was to ensure that the communication protocol be generic enough that the systems can interact with other environments. Neither system should depend the internal workings of the other. At the same time, we were thinking more about designing a useful protocol to cover a set of communication goals at hand rather than about establishing a comprehensive standard.

Within PAT Online, communication with InterBook is handled entirely through the student record server. When a student record is created in InterBook, the InterBook site accesses PAT Online's student record server CGI, passing it the form information for the student's name and password along with several initialization parameters. In our system, for example, we use returnURL, helpURL and helpWIN (see Figure 5). The returnURL parameter specifies a URL which should receive skill update information. This URL is then stored with the student's record so that, whenever the student uses PAT Online, the student record server knows to not only save the skill update information but pass it along to the returnURL location (which, in this case, is InterBook). The helpURL represents a URL that, when appended with a concept name, offers help on the concept. The helpWIN parameter specifies the name of the window in which to display the page associated with this link.

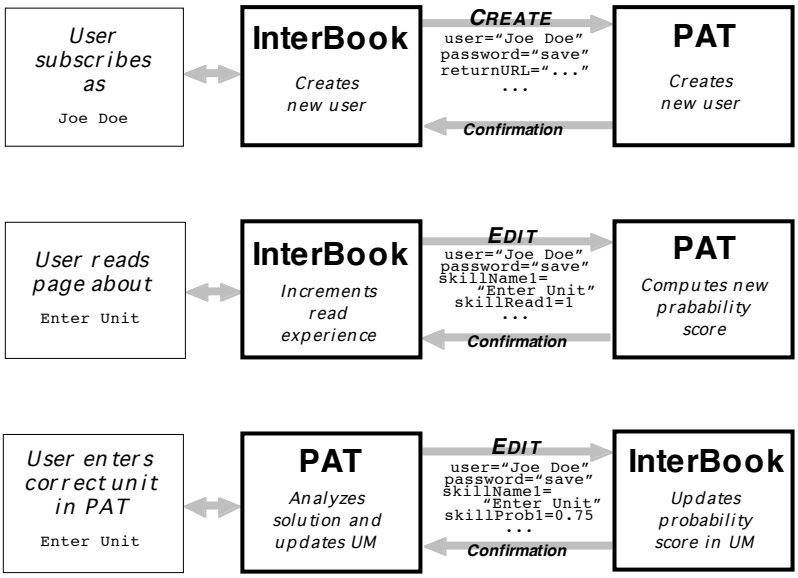

Figure 5: Protocol for Exchanging User Model Data between InterBook and PAT Online.

Incorporating student model information

As described above, PAT Online's student model records the tutor's estimate of the probability that the student has mastered each skill. This is a complicated quantity and not easily mapped to quantities stored in other student models (such as percent correct). One of the most difficult issues involved in sharing student models is how to convert information from one format and unit of measure to another [1; 19]. For this initial version of the tutor, we treat reading about a concept or following an example in Interbook as being equivalent to a request for help and update the probability according to Equation 4.

$$(4)\quad pKnown_T = pKnown_{T-1} + pLearn * (1 - pKnown_{T-1})$$

This formula provides an increment in the tutor's estimate of the user's skill which is intermediate between those updates caused by correct or erroneous performance in the tutor. This is probably an inaccurate approximation, since the help offered by InterBook is more extensive than that offered by PAT Online, and the equation does not account for any differences between reading

about a concept and viewing an example using the concept. We hope to collect data on students using the combined system so that we can refine this estimate.

Since InterBook's student model can accommodate different kinds of information about a single concept, it was relatively straightforward to accommodate updates to the student model suggested by PAT Online (or any other ITS following the communication protocol). When InterBook receives student model information from an ITS, it integrates the information into its own knowledge representation structure using Equation 3. This extension allows InterBook to be an information server for any domain which might be covered by an ITS system on the WWW. This ability frees ITS designers from many secondary duties, which can be taken over more efficiently by InterBook.

Conclusion

We have described an integrated Web-based learning environment consisting of two adaptive tutoring systems. We see the work described here as providing an important first step towards a flexible architecture for communicating between adaptive tutors on the world-wide web. While the communications architecture in our environment is peer-to-peer, requiring the two systems to communicate directly with each other, the protocol is general enough so that any ITS can communicate with InterBook and any conceptual learning system can communicate with PAT Online. This provides a necessary foundation for allowing the two systems to act as components of a centralized architecture incorporating many different web-based educational systems. As the number of adaptive systems capable of this kind of communication grows, we look forward to a day when teachers will be able to integrate learning systems into a seamless and effective web-based educational environment.

Our motivation was to show the potential of integrating several adaptive Web-based systems into a coherent environment which is more than the sum of its parts. This paper addresses the communication side of the integration problem. Further research is needed to address the conceptual side of this problem, i.e. how systems with different domain and student models can find a common semantics for the communication. A first step in defining a common vocabulary for different knowledge-based educational systems can be found in [15].

References

1. Brusilovsky, P.: Student model centered architecture for intelligent learning environment. In: Proc. of Fourth International Conference on User Modeling. MITRE (1994) 31-36
2. Brusilovsky, P.: Intelligent learning environments for programming: The case for integration and adaptation. In: Greer, J. (ed.) Proc. of AI-ED'95, 7th World Conference on Artificial Intelligence in Education. AACE (1995) 1-8
3. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction 6, 2-3 (1996) 87-129
4. Brusilovsky, P., Schwarz, E., Weber, G.: ELM-ART: An intelligent tutoring system on World Wide Web. In: Frasson, C., Gauthier, G., Lesgold, A. (eds.) Proc. of Third International Conference on Intelligent Tutoring Systems, ITS-96. Lecture Notes in Computer Science, Vol. 1086. Springer Verlag (1996) 261-269
5. Brusilovsky, P., Schwarz, E., Weber, G.: A tool for developing adaptive electronic textbooks on WWW. In: Maurer, H. (ed.) Proc. of WebNet'96, World Conference of the Web Society. AACE (1996) 64-69
6. Corbett, A.T., Anderson, J.R.: Student modeling and mastery learning in a computer-based programming tutor. In: Frasson, C., Gauthier, G., McCalla, G. (eds.) Proc. of Second International Conference on Intelligent Tutoring Systems, ITS'92. Lecture Notes in Computer Science, Vol. 608. Springer-Verlag (1992) 413-420
7. Fink, J., Kobsa, A., Nill, A.: User-oriented adaptivity and adaptability in the AVANTI project. In: Proc. of Conference "Designing for the Web: Empirical Studies". Microsoft Usability Group (1996)

8. Frasson, C., Mengelle, T., Aïmeur, E., Gouardères, G.: An actor-based architecture for intelligent tutoring systems. In: Frasson, C., Gauthier, G., Lesgold, A. (eds.) Proc. of Third International Conference on Intelligent Tutoring Systems, ITS-96. Lecture Notes in Computer Science, Vol. 1086. Springer Verlag (1996) 57-65
9. Futtersack, M., Labat, J.-M.: QUIZ, a distributed intelligent tutoring system. In: Tomek, I. (ed.) Proc. of 4th International Conference, ICCAL'92. Springer-Verlag (1992) 225-237
10. Kobsa, A., Pohl, W.: The BGP-MS user modeling system. User Modeling and User-Adapted Interaction 4, 2 (1995) 59-106
11. Koedinger, K.R., Anderson, J.R., Hadley, W.H., Mark, M.A.: Intelligent tutoring goes to school in the big city. In: Greer, J. (ed.) Proc. of AI-ED'95, 7th World Conference on Artificial Intelligence in Education. AACE (1995) 421-428
12. Lai, M.-C., Chen, B.-H., Yuan, S.-M.: Toward a new educational environment. In: Proc. of 4th International World Wide Web Conference. (1995)
13. McCalla, G.I., Greer, J.E., Scent, R.T.: SCENT-3: An architecture for intelligent advising in problem-solving domains. In: Frasson, C., Gauthier, G. (eds.): Intelligent Tutoring Systems: At the crossroads of artificial intelligence and education. Ablex Publishing, Norwood (1990) 140-161
14. Milosavljevic, M., Tulloch, A., Dale, R.: Text Generation in a Dynamic Hypertext Environment. In: Proc. of Nineteenth Australasian Computer Science Conference (ACSC'96). (1996) 417-426
15. Mizoguchi, R., Sinitsa, K., Ikeda, M.: Task ontology design for intelligent educational/training systems. In: Proc. of Workshop "Architectures and Methods for designing Cost-Effective and Reusable ITSs" at the Third International Conference on Intelligent Tutoring Systems, ITS-96. (1996)
16. Nakabayashi, K., Maruyama, M., Koike, Y., Fukuhara, Y., Nakamura, Y.: An intelligent tutoring system on the WWW supporting interactive simulation environments with a multimedia viewer control mechanism. In: Maurer, H. (ed.) Proc. of WebNet'96, World Conference of the Web Society. AACE (1996) 366-371
17. Néhémie, P.: A systemic approach for student modelling in a multi-agent aided learning environment. In: Frasson, C., Gauthier, G., McCalla, G.I. (eds.) Proc. of Second International Conference, ITS'92. Springer-Verlag (1992) 475-482
18. Paiva, A., Self, J.: TAGUS - a user and learner modeling workbench. User Modeling and User-Adapted Interaction 4, 3 (1995) 197-226
19. Paiva, A., Self, J., Hartley, R.: Externalising learner models. In: Greer, J. (ed.) Proc. of AI-ED'95, 7th World Conference on Artificial Intelligence in Education. AACE (1995) 509-516
20. Ritter, S.: Comminication, cooperation and competition among multiple tutor agents. In: Boulay, B.d., Mizoguchi, R. (eds.) Proc. of AI-ED'97, 8th World Conference on Artificial Intelligence in Education. IOS (1997)
21. Ritter, S., Koedinger, K.R.: Towards lightweight tutoring agents. In: Greer, J. (ed.) Proc. of AI-ED'95, 7th World Conference on Artificial Intelligence in Education. AACE (1995) 91-98
22. van Rosmalen, P.: SAM, Simulation and Multimedia. In: de Jong, T., Sarti, L. (eds.): Design and production of multimedia and simulation-based learning material. Kluwer Academic Publishers, Dordrecht (1994) 167-187