Crossroads
The ACM Student Magazine

# Applications of AI in Education

by **Joseph Beck**, **Mia Stern**, and **Erik Haugsjaa**

Computers have been used in education for over 20 years. **Computer-based training (CBT)** and **computer aided instruction (CAI)** were the first such systems deployed as an attempt to teach using computers. In these kinds of systems, the instruction was not individualized to the learner's needs. Instead, the decisions about how to move a student through the material were script-like, such as ``if question 21 is answered correctly, proceed to question 54; otherwise go to question 32.'' The learner's abilities were not taken into account.

While both CBT and CAI may be somewhat effective in helping learners, they do not provide the same kind of individualized attention that a student would receive from a human tutor [**5**]. For a computer based educational system to provide such attention, it must reason about the domain and the learner. This has prompted research in the field of **intelligent tutoring systems (ITSs)**. ITSs offer considerable flexibility in presentation of material and a greater ability to respond to idiosyncratic student needs. These systems achieve their ``intelligence'' by representing **pedagogical** decisions about how to teach as well as information about the learner. This allows for greater versatility by altering the system's interactions with the student.

Intelligent tutoring systems have been shown to be highly effective at increasing students' performance and motivation. For example, students using Smithtown, an ITS for economics, performed equally well as students taking a traditional economics course, but required half as much time covering the material [**25**].

In this paper, we start by providing an overview of the main components of intelligent

tutoring systems. We then provide a brief summary of different types of ITSs. Next, we present a detailed discussion of two components, the student model and the pedagogical module. We close by discussing some of the open questions in ITS as well as future directions of the field.

## Components of Intelligent Tutoring Systems

Intelligent tutoring systems may outwardly appear to be monolithic systems, but for the purposes of conceptualization and design, it is often easier to think about them as consisting of several interdependent components. Previous research by Woolf [**32**] has identified four major components: the student model, the pedagogical module, the domain knowledge module, and the communication module. We have identified a fifth component, the expert model. Woolf includes this component as part of the domain knowledge, but we feel that it is a separate entity. Figure 1 provides a view of the interactions between the modules.

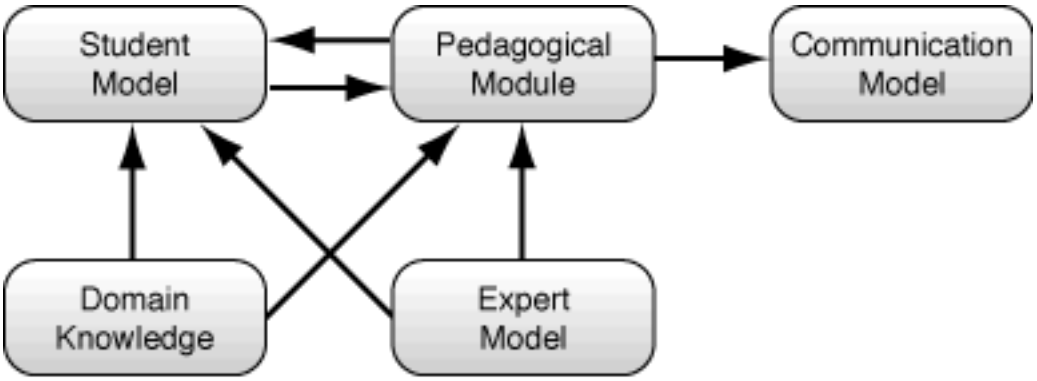



Figure 1: Interactions of components in an intelligent tutoring system.

### Student Model

The student model stores information that is specific to each individual learner. At a minimum, such a model tracks how well a student is performing on the material being taught. A possible addition to this is to also record misconceptions. Since the purpose of the student model is to provide data for the pedagogical module of the system, all of the information gathered should be able to be used by the tutor.

### Pedagogical Module

This component provides a model of the teaching process. For example, information about when to review, when to present a new topic, and which topic to present is controlled by the pedagogical module. As mentioned earlier, the student model is used

as input to this component, so the pedagogical decisions reflect the differing needs of each student.

### Domain Knowledge

This component contains information the tutor is teaching, and is the most important since without it, there would be nothing to teach the student. Generally, it requires significant knowledge engineering to represent a domain so that other parts of the tutor can access it. One related research issue is how to represent knowledge so that it easily scales up to larger domains. Another open question is how to represent domain knowledge other than facts and procedures, such as concepts and mental models.

### Communications Module

Interactions with the learner, including the dialogue and the screen layouts, are controlled by this component. How should the material be presented to the student in the most effective way? This component has not been researched as much as the others, but there has been some promising work in this area [**27**, **31**].

### Expert Model

The expert model is similar to the domain knowledge in that it must contain the information being taught to the learner. However, it is more than just a representation of the data; it is a model of how someone skilled in a particular domain represents the knowledge. Most commonly, this takes the form of a runnable expert model, i.e. one that is capable of solving problems in the domain [**8**]. By using an expert model, the tutor can compare the learner's solution to the expert's solution, pinpointing the places where the learner had difficulties.

## Types of ITSs

There are several ways of categorizing ITSs; we will concentrate on two dimensions: abstraction of the learning environment and the knowledge type of the instruction.

### Abstraction of the learning environment

Many systems attempt to provide instruction by simulating a realistic working environment in which the student can learn the task. There are many reasons for developing such systems, including the possible danger of training using the actual equipment and the lack of domain experts who can devote their expensive time to

training novices. Therefore, a realistic simulated learning environment can reduce both the cost and the risks of training.

An example of a simulation-based ITS is the Advanced Cardiac Life Support (ACLS) Tutor [**9**] in which a student takes the role of team leader in providing emergency life support for patients who have had heart attacks. The system not only monitors student actions, but runs a realistic simulation of the patient's condition and maintains an environment that is reasonably faithful to the ``real life'' situation. Thus, the goal is not only to test the student's knowledge about the correct emergency procedures, but also to allow him to experience practicing those procedures in a more realistic manner than is possible in a traditional classroom.

Some systems take a less rigorous approach to representing the environment; the situations presented are similar to the real world scenarios in which the knowledge could be applied, but they are not exact simulations. Smithtown [**25**] takes this approach by providing a simulated setting for students to test hypotheses about economics. However, the underlying model of the environment is not an exact simulation of how the laws of economics would be applied in the real world. Another example of such a system is the Design for Manufacturing Tutor [**10**].

At the extreme opposite of the simulation based tutors are those that teach knowledge in a decontextualized manner without attempting to simulate the real world. Many systems throughout the history of ITS research fall into this category [**2**, **24**]. These systems provide problems for the learner to solve without trying to connect those problems to a real world situation and are designed to teach abstract knowledge that can be transferred to multiple problem solving situations.

### Emphasis of Instruction

There is a long history of classifying instructional goals according to the type of knowledge being taught. An important early attempt at this classification is Bloom's taxonomy [**4**] and much recent work in categorizing knowledge has been derived from this. In addition to classifying learning goals by knowledge type, one can also examine what the student will be able to do upon completion of the ITS's lesson. This can vary from the student being able to perform a set of skills in a manner similar to an expert to understanding abstract concepts such as Newton's third law.

For ease of development, systems tend to concentrate on teaching one type of knowledge. The most common type of ITS teaches procedural skills; the goal is for students to learn how to perform a particular task. There has been substantial research in cognitive psychology about human skill acquisition, so analyzing the domain knowledge in this framework can prove beneficial to instruction. Systems that are designed according to these principles are often called **cognitive tutors**. The most common result of this analysis is a set of rules that are part of a runnable expert model. This set of expert rules often serves double duty as a knowledge of the domain and as the pedagogical module. If a student encounters difficulty, the specific remediation required can be determined from the expert model.

An example of a ``cognitive tutor'' is SHERLOCK, which has tutorial actions associated with each state in the ``effective problem space'' [**14**]. Another example of an ITS that uses an analysis of expert behavior is the LISP tutor [**3**], which encodes expert problem solvers' actions as production rules, and attempts to determine which rules the student is having difficulty applying.

Other ITSs concentrate on teaching concepts and ``mental models'' to students. These systems encounter two main difficulties. First, a more substantial domain knowledge is needed for instruction. Second, since learning concepts and frameworks is less well understood than learning procedures, there is less cognitive theory to guide knowledge representation and the pedagogical module. For these reasons, ITSs of this type require a larger domain knowledge base and are sometimes referred to as **knowledge based tutors**. As a result of not having a strong model of skill acquisition or expert performance, these systems are forced to use general teaching strategies. They also place more emphasis on the communication and presentation system in order to achieve learning gains. An example of such a system is the Pedagogical Explanation Generation (PEG) system [**27**] which has an explanation planning component that uses a substantial domain knowledge base to construct answers to student queries in the domain of electrical circuits.

These classifications are really points along a continuum, and serve as good rules of thumb rather than a definitive method of classifying intelligent tutors. A system that does not fall into either of these categories is Coach [**31**], which teaches how to use UNIX mail. This is a procedural skill, and hence cognitive in nature. However, the emphasis of this system is also knowledge based and involves generating explanations and using general pedagogical tactics for generating feedback.

Generally, tutors that teach procedural skills use a cognitive task analysis of expert behavior, while tutors that teach concepts and frameworks use a larger knowledge base and place more emphasis on communication to be effective during instruction. There are exceptions to these rules, but they serve as useful guidelines for classifying ITSs.

## The Student Model

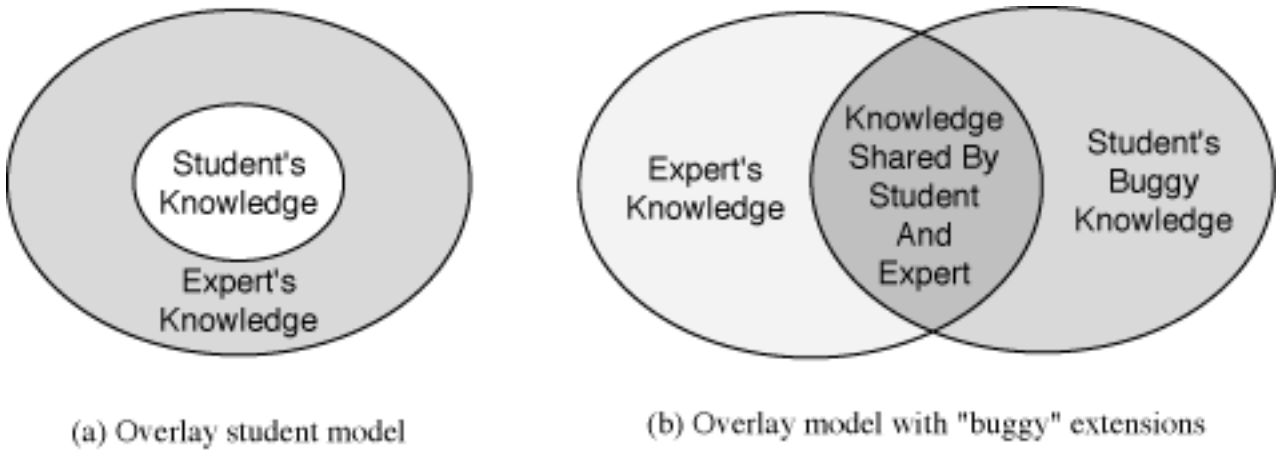
As noted previously, the student model is the component of an ITS that records information about the student. This information reflects the system's belief of the learner's current knowledge state. Since only overt student actions are visible, and the ITS only has a relatively narrow channel of communication with the user, there is difficulty in obtaining an accurate representation of the student's abilities. Therefore, the model of the student may not be perfectly accurate and steps must be taken to ensure that the system's actions on the basis of this inaccurate information are not inappropriate. For example, a tutor that interferes too much with a learner who is performing satisfactorily can obviously be detrimental.

After considering the above difficulties, an obvious question concerning student models is why to have one. Simply put, the student model is necessary in order to tailor instruction to a student's idiosyncrasies and learning needs. Without this knowledge, the pedagogical component of the tutor has no basis on which to make decisions, and is forced to treat all students similarly. This is analogous to earlier efforts in CBT and CAI which did not customize instruction for individual learners.

### Representation of the student model

There are many methods for representing information about the student. Two commonly used techniques are overlay models and Bayesian networks.

The standard paradigm for representing a student model is the **overlay model** [7] in which the student's knowledge is considered to be a subset of the expert's knowledge (Figure 2a). With this representation, an ITS presents material to the student so that his knowledge will exactly match that of the expert. The knowledge types that can be represented within an overlay student model include `topics', which correspond to elements of the domain knowledge, and production rules [1].




(a) Overlay student model

(b) Overlay model with "buggy" extensions

Figure 2: Overlay student models.

A drawback of this approach is that it does not acknowledge that students may have beliefs that are not part of the expert's knowledge base. For example, students frequently have misconceptions about a domain. Therefore an extension to the overlay model explicitly represents ``buggy'' knowledge that the student may have (Figure 2b) [**11**]. This extension allows for better remediation of student mistakes, since the fact that a student believes something that is incorrect is pedagogically significant.

Another mechanism for recording a student's knowledge is **Bayesian networks** [**18**]. These networks probabilistically reason about a student's knowledge state based on his interactions with the tutor. Each node in the network has a probability indicating the likelihood of the student ``knowing'' that piece of knowledge.

### What to include in the student model

ITS designers have a tendency to include more information in the student model than the pedagogical module can use [**23**]. However, the point of having a student model is to be able to tailor the instruction for each student. Therefore, student models should only include data that the tutor will actually use in creating this instruction. On the other hand, for research purposes, it can be beneficial to include extra factors that are not used by the pedagogical module but provide ITS designers with knowledge of what may be useful to include in future student models.

Given this restriction, what should a student model contain? Clearly, it must record the student's understanding of the domain. However, at what grain size should that understanding be represented? At one extreme, the tutor could simply say ``the student knows this domain'' or ``the student does not know this domain.'' At the other

extreme, the student model could record every student action. Most student models fall in between these two end points and try to model the student at the same granularity at which the domain is represented. Since most domains are represented in terms of topics, this is the most common grain size for student models.

In addition to recording a student's understanding of the domain, a student model may also include more general pedagogical information about the student. This kind of information could include a student's general preferences, such as whether he likes to look at examples before attempting to answer any questions or whether he likes examples about ponies but not those about tanks.

Other general information about the student's learning can be included, such as acquisition and retention. **Acquisition** measures how fast students learn new topics, and **retention** measures how well they recall the material over time. Prior research suggests that examining general factors such as acquisition and retention can be beneficial for student modeling. Work with the LISP tutor [**3**] and with Stat Lady [**24**] indicates that general factors extracted from student learning data are predictive of overall learning and allow for a more accurate response to the idiosyncrasies of the student.

## Pedagogical Module

The pedagogical module uses information from the student model to determine what aspects of the domain knowledge should be presented to the learner. This information, for example, may be new material, a review of previous topics, or feedback on the current topic. One pedagogical concern for an ITS is the selection of a meta-strategy for teaching the domain. For example, the system could decide to use the Socratic method [**21**] or it could select a topic and present an example of a problem within that topic. Once the meta-strategy is selected, low level issues, such as the exact example to use, must be decided. These low level issues have been fairly well researched, and thus will be discussed first.

### Low level issues

The tutor must decide the content of the material to be presented to the student. This involves decisions about the topic, the problem, and the feedback.

*Topic selection:* To select a topic to present, the tutor must examine the student model

to determine the topics on which the student needs to focus. Many possibilities exist for the most appropriate topic on which a student should work. For example, if the meta-strategy indicates that review is in order, the tutor will select a topic the student has already ``learned.'' On the other hand, if new information is to be presented, the tutor will choose a topic that the student does not yet know.

*Problem generation:* Once the topic has been selected, a problem must be generated for the student to solve. The grain size of the problem is determined by the domain. For example, in SHERLOCK, the student will be asked to diagnose the fault in the station used to repair an F-15, while in MFD [**26**], the student will be given a simple math problem, such as adding two fractions. Whatever the granularity of the problem generated, it is important that the difficulty be appropriate for the student's level of ability, which can be determined from the student model.

*Feedback:* Most tutors work smoothly as long as students get everything right. Problems arise when the student has difficulties and needs help from the tutor. In these situations, the tutor must determine the kind of feedback to provide. The issue of how much help to provide the student is also a very complex issue as too little feedback can lead to frustration and floundering [**1**] while too much feedback can interfere with learning [**12**].

Once the system decides how much feedback to give, it must determine the content of the advice. The feedback should contain enough information so that the student can proceed to the next step in solving the problem. Furthermore, the advice given to the learner should be appropriate for her ability level. Some systems use the student model to select a hint that most closely matches the learner's level of ability. For example, in MFD, the more proficient the student is at a particular skill, the more subtle the hint is. On the other hand, a student with low proficiency in a skill would be presented with a more obvious hint. By using this technique, learners will not be required to wade through many levels of hints before receiving useful help [**15**].

### Meta-strategy selection

High level strategy selection in ITSs has not received the same amount of attention as the low level decisions. This is not to say that meta-strategies have not been researched. To the contrary, educational research has identified many potential teaching strategies for use by an ITS [**13**]. Examples of these kinds of strategies

include spiral teaching [6] and the Socratic method.

However, implementing these meta-strategies in an ITS has proven a formidable problem. Most ITSs do not explicitly identify the strategies they are using for teaching and implicitly implement one of the well-known strategies [17]. A better method is to use the student model to select an appropriate strategy from those maintained by the system. Ideally a student's model could track the instructional strategies that are most effective for teaching him. However, because most systems do not have multiple teaching strategies, student models have not been designed to provide selection information. Thus, representing multiple strategies explicitly and the control knowledge to select among them is beyond the capabilities of most current systems.

Another obstacle in representing multiple teaching strategies is the limitations imposed by other components of the ITS in addition to those placed by the student model. In particular, the difficulty of representing knowledge impedes the ability to explicitly represent teaching strategies. For example, the Socratic method requires substantial ``common sense'' knowledge beyond what is maintained in a domain knowledge base. This problem of scaling up the knowledge base is not unique to ITSs and is common to many areas of AI.

## Future Work

In this section, we discuss some of the open questions in intelligent tutoring systems research. In general, many of these questions fall into two categories: (1) reducing the time and cost of development and (2) allowing students to work collaboratively.

### Reducing development time and cost

One of the main difficulties in designing intelligent tutoring systems is the time and cost required. A large team, including computer programmers, domain experts, and educational theorists, is needed to create just one ITS. Estimates of construction time indicate that 100 hours of development translates into 1 hour of instruction [20]. Clearly there is a need for techniques that will help alleviate these difficulties for instructional development.

*Authoring tools:* The goal of authoring tools is to provide a (relatively) simple development environment and as a result, fewer developers would be needed for the construction of educational software. There are two main approaches to achieving this

goal: (1) provide a simple development shell for educators to author their own courseware and (2) provide a means for programmers to more easily represent the domain and teaching strategies. Authoring tools that fall into the first category generally have a restricted scope of the types of instructional interactions a user can create, whereas those in the second category allow for considerably more flexibility at the cost of more complex authoring. REDEEM [**16**] and RAPIDS II [**29**] are examples of authoring tools of the first type and Eon [**19**] and IDE [**22**] are examples of authoring tools of the second approach.

*Modularity:* Another approach to simplifying ITS construction is to take advantage of the modularity of each system. Despite the natural breakdown of an ITS into the five components discussed previously, there has been little effort towards reusing components from one system in the development of another. This should not involve developers just reusing their own components, but should also mean sharing components among different designers.

Numerous difficulties impede such modularization. First, tutors are written in many different programming languages that are incompatible. Second, this component view of ITSs is more of an ideal situation than a reality. Frequently, implementors intertwine the components into one monolithic system. Furthermore, since the field of ITS is relatively young, there are not accepted standards for kinds of components nor for their contents. Finally, there is no protocol for communication between the various parts of an ITS.

## Collaborative learning

Collaborative learning refers to students working in groups to solve problems. These environments have been shown to be beneficial, both cognitively and socially [**30**]. In these situations, the focus of the interactions is not typically between the teacher and the learners, as students can teach each other without input from the instructor. For the purposes of this discussion, we restrict the term **collaborative learning** to refer to students working together, with the aid of an ITS, via a computer network. Because other students are involved in the learning, the design of the ITS should be easier since the instruction does not have to be perfect -- if a student becomes confused, another student may be able to help without relying on the ITS for assistance.

An important aspect of collaborative environments is that in group situations, not all students will be of the same ability. This creates two problems. The first, the classic

problem of credit assignment, affects how student models are updated. Should credit be awarded only to the first person to produce the correct answer or should all members of the group receive equal reward? The second problem concerns the pedagogical decisions of how to advance the group through the curriculum. Should one learner dictate the pace of the entire group? And if so, which one?

Although collaborative learning as we have defined it is in its infancy, there have been some efforts in this direction. For example, Belvedere [**28**] provides a set of tools to help groups of students construct theories (for example, on evolution), and can then critique these theories.

## Conclusion

Intelligent tutoring systems have been shown to be highly effective in increasing student motivation and learning. In designing these systems, it is useful to view them as being composed of five components: the student model, the pedagogical module, the domain knowledge, the communications module, and the expert model. Research has been done on each of these modules, but only a few are very well understood. Specifically, incorporating multiple teaching strategies in the pedagogical module is a large open research question.

In addition to the continuing work on these components, one important research issue is reducing the time and cost to develop such systems. Current strategies for doing this include the development of authoring tools and creating systems in a modular fashion. Solving this problem will be an enormous breakthrough in ITS research, since more systems could be constructed and thus more research into the effectiveness of computer based instruction could be performed.

## References


**1**

Anderson, J. 1993. ***Rules of the Mind***. New Jersey, Lawrence Erlbaum Associates.

**2**

Anderson, J., Boyle, C., and Yost, G. 1985. The Geometry Tutor. In *Proceedings of the Ninth IJCAI*, Los Angeles, Morgan-Kaufmann, San Mateo, Calif.

**3**

Anderson, J. and Reiser, B. 1985. The LISP Tutor. *Byte*, 10, 4, pp. 159-175.

**4**

Bloom, B. (ed.) 1956. *Taxonomy of Educational Objectives*. New York, NY, Mackay Publishing.

**5**

Bloom, B.S. 1984. The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher*, 13, pp. 3-16.

**6**

Bruner, J. 1992. Science Education and Teachers: A Karplus Lecture. *Journal of Science Education and Technology*, 1, 1, March, pp. 5-12.

**7**

Carr, B. and Goldstein, I. 1977. Overlays: a Theory of Modeling for Computer-aided Instruction, Technical Report, AI Lab Memo 406, MIT.

**8**

Clancey, W. and Letsinger, R. 1981. Tutoring Rules for Guiding a Case Method Dialog. In *Proceedings of the Sixth IJCAI*, Vancouver, B.C., Morgan-Kaufmann, San Mateo, California, pp. 829-835.

**9**

Eliot, C. and Woolf, B. 1995. An Adaptive Student Centered Curriculum for an Intelligent Training System. *User Modeling and User-Adapted Interaction*, 5, 1, pp. 67-86.

**10**

Haugsjaa, E. and Woolf, B. 1996. 3D Visualization Tools in a Design for Manufacturing Tutor. In *Proceedings of Educational Multimedia and Hypermedia*, Boston, Mass.

**11**

Holt, P., Dubs, S., Jones, M., and Greer, J. 1993. The State of Student Modelling. In *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, Greer, J. and McCalla, G., eds., Springer-Verlag, New York.

**12**

Kashihara, A., A. Sugano, K. Matsumura, T. Hirashima, and J. Toyoda. 1994. A Cognitive Load Application Approach to Tutoring. In *Proceedings of the Fourth International Conference on User Modeling*, pp. 163-168.

**13**

Kearsley, G. Learning and Instruction: The TIP Database. ***http://gwis2.circ.gwu.edu/~kearsley/***.

**14**

Lajoie, S. and A. Lesgold. 1992. Apprenticeship Training in the Workplace: Computer-Coached Practice Environment as a New Form of Apprenticeship. In

***Intelligent Instruction by Computer***, Farr and Psotka, eds., Taylor & Francis, Washington, DC, pp. 15-36.

**15**

Lajoie, S.P. 1993. Computer Environments as Cognitive Tools for Enhancing Learning. ***Computers as Cognitive Tools***, Lajoie, S. and Derry, S., eds., Lawrence Erlbaum Associates, NJ, pp. 261-288.

**16**

Major, N. 1995. REDEEM: Creating Reusable Intelligent Courseware. In *Proceedings of the International Conference on Artificial Intelligence in Education*., Greer, J., ed., AACE, Charlottesville, VA, pp. 75-82.

**17**

Major, N. and H. Reichgelt. 1992. COCA - A Shell for Intelligent Tutoring Systems. In *Proceedings of Intelligent Tutoring Systems*, Frasson, C., Gautheir, G. and McCalla, G., eds., Springer Verlag, Berlin.

**18**

Martin, J. & VanLehn, K. (in press) Student assessment using Bayesian nets. International Journal of Human-Computer Studies, Vol. 42.

**19**

Murray, T. 1996. Having It All, Maybe: Design Tradeoffs in ITS Authoring Tools. In *Proceedings of the Third International Conference on Intelligent Tutoring Systems*, Montreal.

**20**

Murray, T. and B. Woolf. 1992. Results of Encoding Knowledge with Tutor Construction Tools. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, July, pp. 17-23.

**21**

Plato. 1924. *Laches, Protagoras, Meno, and Euthydemus.* (W.R.M. Lamb, trans.) Harvard University Press, Cambridge, Mass.

**22**

Russell, D. 1988. IDE: The Interpreter. ***Intelligent Tutoring Systems: Lessons Learned***, Psotka, J., L. Massey, and S. Mutter, eds., Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 323-349.

**23**

Self, J. 1990. Bypassing the Intractable Problem of Student Modelling. In *Intelligent Tutoring Systems: at the Crossroads of Artificial Intelligence and Education*, C. Frasson and G. Gauthier, eds., Ablex Publishing Company, Norwood, NJ.

**24**

Shute, V. 1995. Smart: Student Modeling Approach for Responsive Tutoring. *User Modeling and User-Adapted Interaction*, 5, 1, pp. 1-44.

**25**

Shute, V., R. Glaser, and K. Raghaven. 1989. Inference and Discovery in an Exploratory Laboratory. *Learning and Individual Differences*, Ackerman, P., R. Sterberg, and R. Glaser, eds., pp. 279-326.

**26**

Stern, M., J. Beck, and B. Woolf. 1996. Adaptation of Presentation and Feedback in an Intelligent Mathematics Tutor. In *Proceedings of the Third International Conference on Intelligent Tutoring Systems*, Montreal.

**27**

Suthers, D. 1992. Answering Student Queries: Functionality and Mechanisms. In *Proceedings of the 2nd International Conference on ITS (ITS-92)*, Montreal, June.

**28**

Suthers, D., A. Weiner, J. Connelly, and M. Paolucci. 1995. Belvedere: Engaging Students in Critical Discussion of Science and Public Policy Issues. In *Proceedings of Artificial Intelligence in Education,* Greer, J., ed., AACE, Charlottesville, VA, pp. 266-273.

**29**

Towne, D. and A. Munro. 1992. Supporting Diverse Instructional Strategies in a Simulation-Oriented Training Environment. ***Cognitive Approaches to Automated Instruction***, Regian, J. and V. Shute, eds., Lawrence Erlbaum, Hillsdale, NJ, pp. 107-134.

**30**

Webb, N. 1982. Student Interaction and Learning in Small Group. *Review of Educational Research*, 52, pp. 421-445.

**31**

Winkels, R., J. Sandberg, and J. Breuker. 1990. The Coach. In *EUROHELP: Developing Intelligent Help Systems*, Breuker, J., ed., EC: Copenhagen, pp. 119-146.

**32**

Woolf, B. 1992. AI in Education. ***Encyclopedia of Artificial Intelligence***, Shapiro, S., ed., John Wiley & Sons, Inc., New York, pp. 434-444.