Manipulating Context to Improve Student Learning Time

Christopher Eliot
Beverly Park Woolf

CMPSCI Technical Report 93-41

Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{bev,eliot@cs.umass.edu}

Abstract

This research addresses difficult issues in generating computer responsiveness based on assumptions about the user's state of knowledge and skill. A sophisticated tutoring system is described that reasons about and manipulates the context of the learning environment to ensure that the situation continues to interest and challenge the student. Intelligent tutors and simulation systems are not currently customized to improve learning times. In the described system, a tight interaction between user modeling techniques and simulation management reduces the time each student spends with a tutor to reach a given skill level. Control mechanisms concentrate on reasoning about reducing the propagation of error in the user model, enhancing the human-computer interaction process when the model is accurate, without creating serious problems when it is not. This research makes significant contributions to development of a sophisticated user model and explores techniques for reducing the time each student spends with an interactive tutoring system to reach a given skill level.

The Problem

Control is at issue when a system must be responsive and flexible with its user. Strong methods are needed for dealing with a user's unknown skills and knowledge and with change, contradictions, and inconsistencies in assumptions made by the system about this knowledge. For example, reasoning in an intelligent tutor should be non-monotonic, i.e., inferences made about a student's knowledge, if shown to be inaccurate, should be replaced by new inferences based on more recent activities. Before a student begins to work with a tutor, control issues need to be addressed about the acquisition of and reasoning about pedagogy, domain topics, and machine responses. *After* a student begins to work with a tutor, different issues need to be addressed about the dynamic analysis of student behavior, automatic diagnosis and remediation, identification of appropriate pedagogical strategies, and generation of effective responses. The goal is not only generation of immediate responses in the short-term, or single session, but also the improvement of instruction in the long term between sessions. Current systems are too simplistic and too static to reason effectively about human learning; frequently they focus on reasoning about the risk of propagating information about the certainty (or lack of same) in a user model [Self, 1987] rather than reasoning about action to take when inferences are inaccurate, as inevitably happens.

Optimal control cannot be accomplished with a single-shot decision mechanism. Rather than render a `dead reckoning' type diagnosis and prescription for further action, a reasoner requires *processes* that make use of multiple diagnostic and reasoning methods and that modify their approach or do additional work based on the characteristics of student-machine interactions. Sophisticated control techniques are required for developing and managing the complex problem solving plan necessary to implement such processes. Additionally, action-oriented reasoning mechanisms are required which can interleave reasoning and action. The system described in this article *reasons a little, acts a little,* and then *reassess a little:* It sketches a plan of action (diagnosis and prescribed strategy), executes a small portion of this plan, assesses the plan's effectiveness, and then reformulates its goal [Fennema, 1991]. Such formulation by being action-oriented, is more efficient than traditional tutors or simulations in that it teaches and affects the simulation only when appropriate.

Classical planning systems typically cannot represent realistic domains: the search is either incomplete or impossibly time consuming. An incomplete (heuristic) search may be acceptable, however, if it can be organized so that good plans are likely to be found quickly. Selection of optimal search strategies cannot be a completely theoretical exercise: the definition of "good plans" must be derived from realistic problems. AI planning research has developed a very deep understanding of an idealized class of planning problems, but has made relatively little progress towards using these results in real domains. For instance, a nonlinear hierarchical planner can solve a classical planning problem in domains where the operators are independent. Unfortunately, classical planners work with representations that are not rich enough for most real problems. Enriching the representation sufficiently to represent real problems makes the search space too large. It has been proven that expanding the representation of planning operators (in several specific ways [Chapman, 1987]) makes the search for a solution to a planning problem arbitrarily large.

Our Solution

This research addresses the difficulty of controlling system response by implementing control mechanisms focused on goal selection, plan formation, and plan instantiation within situated contexts. Errors in the user model will fall away by the nature of the changing context. If the system suggests less than optimal strategies based on an error-full user model, the output will be weak, but will not be propagated beyond the current context and as the system quickly moves on to new contexts, the revised user model suggests improved system responses. Rather than develop the accuracy of the user model, this research focuses on how to reduce the propagation of error from a possibly inaccurate user model and on how to proceed (e.g., [Broverman and Croft, 1987]) . Control enhances the human-computer interaction process when the model is accurate, without creating serious problems when it is not. Little effort is placed on deliberating whether the user model is accurate, rather the focus is on tolerating inaccuracies by the logic of their use.

The sophisticated tutoring system reasons about and manipulates the environment to present explanations and content in a favorable context and to ensure that problems continue to interest and challenge the student. It improves the tutoring result by biasing the environment as the system moves on toward new contexts creating opportunities for student learning. Within a tutoring system, several design principles follow directly from the primacy of this view. The user model serves to guide allocation of the tutor's internal resources and informs the tutor's decisions about how to influence the interaction. It is well suited to support high level strategic decisions by the tutor, but not to control detailed tutor interaction. On the other hand, the user model will sometimes be inaccurate so it should not be seen by the user; it is not an evaluation tool. Nor should it be used to control the user; the system should remain responsive and interactive relative to student initiatives. These principles are combined in the design of the intelligent interactive simulation as described in the remainder of this paper.

The tutor is based on the following four abstract steps:

* Goal selection: The system reasons about the user's needs and the best context for meeting them.
* Plan formation: The system moves from one context to a new context selected for goal satisfaction. This step is complex because the domain contains dynamically changing processes and the history of a simulated process must remain consistent.
* Plan instantiation: Once a plausible way to achieve a desirable new context has been found, the system will simulate the corresponding transitions. User input is integrated with the simulation possibly forcing the system to revise or abandon outstanding plans.
* Situated reasoning. When the system reaches a desired context it applies situation specific knowledge to help achieve the user's goals.

This solution is general because most of the domain specific reasoning is encapsulated in the simulation or the situated reasoning components. The power of this approach is the modularity it implies for knowledge representation. We separate knowledge of 'where to be' and 'how to get there' from knowledge of 'what to do once there.' Examples are presented of a mechanism which presents a user with an appropriate situation, thus solving the 'where' and 'how' problems. The remaining problem of 'what to do once there' is greatly simplified because the exact machine response is made in a context chosen to be most conducive to a successful interaction. This research makes significant contributions to development of a sophisticated user model and explores techniques for reducing the time each student spends with a interactive tutoring system to reach a given skill level.

An important dilemma is that a system cannot be expected to determine when the user model is accurate. Inaccuracies must be tolerated and damped by the logic of their use. At first this may seem impossible, but it is analogous to the commonplace skill required to carry on a conversation with someone whose name has been forgotten. By choosing phrases and topics based on partial or fragmented knowledge with a suitable degree of generality, a person might avoid admitting an embarrassing memory gap. Conceptually similar techniques allow us to design a user interface that masks uncertainties of the user model.

Applying the User Model

A user model represents an individual user for the purpose of customizing program behavior. The most primitive form of user modeling records a stereotype of the user's skills and stated preferences. This is useful, but intelligent user modeling offers more and can actively develop its own model of a user, incorporating available clues to best anticipate the user's goals, needs and abilities. To be effective, a user model must have predictive content; it cannot be purely descriptive. Analysis of user actions is not practical unless it contributes to improved system behavior. In this section, principles of the user model are defined and in subsequent sections the implementation described to illustrate how the user model principles contribute toward the overall effectiveness of the system.

Implementing a user model requires: 1) a structure in which to record inferences about the user; 2) a mechanism for building a user specific model within this theory and 3) knowledge about how to apply this model to other aspects of the system's behavior. The first requirement limits what the system can "know" and amounts to a domain specific psychological theory. Knowledge about how to apply a user model is arguably the most important factor in the success of a user model, possibly more important even than efforts to improve its accuracy, although the latter is obviously important. However. without major new advances in psychological theory any user model will be subject to inaccuracies and uncertainty. Reducing these factors will help, but the system must be built so that whatever inaccuracy and uncertainty remains will be tolerated by the system.

Several inferences can be made from these observations. The user model may be inaccurate, which is to say that it may be inconsistent with the reality of the user. Hence. we cannot reason from a user model simply by classical logic because such inconsistency would render the reasoning meaningless. The system must tolerate unexpected user actions unobtrusively, since the user might be correct and the user model at fault [Broverman, 1991]. A user will also change over time, introducing another form of inconsistency that the user model must tolerate. Application of the user model is guided by these principles: 1) conclusions are always tentative and may need to be revised; 2) negative consequences of an error should be minimized; 3) positive consequences should be maximized; and 4) the weakest sufficient assumptions should be used. For example, when the user's needs can be anticipated, automation of subsequent steps is possible. However, the value of this automation must be balanced against the consequences of misinterpreting the user's intentions.

Biasing the Simulation

Many tutoring systems are based upon a simulation of a real world task in which each situation confronts the student with a different set of problems. One simulated situation might enable a student to learn better than another situation. For example, as a student learns more material, problems should be made more complex to remain challenging, but not overwhelming. Similarly, a student will better grasp an explanation if it relates to a recently encountered situation than if the staement is made out of context.

We have built a simulation which is biased toward situations that create opportunities for student learning. In some domains, biasing a simulation is straight-forward. For example, a flight simulation can present new problems for the student naturally by altering what lies beyond the horizon and allowing the student to fly into it. However, we are working in a medical domain which has more constraints. The simulation must proceed in physiologically and clinically plausible sequences, taking into account all interventions that have been applied so far. These path constraints severely restrict the order in which our system can present problems.

Another concern is development of a user model rich enough to express the goals toward which the simulation should be directed. The model of an individual student should be accurate enough so the system can reason effectively about which tutoring goals to set for that student. Accuracy in a user model is an important issue and requires substantial research effort. The medical domain user model integrates medical constraints, pedagogical constraints and user model features to determine a current set of tutoring goals, their relative priority and plans for achieving them. Such plans are built and used to improve the learning time for students based on those user model principles proposed in the previous section.

Finally, there are statistical constraints on the simulation: events in the simulation must occur with a realistic overall probability distribution. Making the simulation goal-directed inherently conflicts with making it realistic. Resolving this conflict demands achieving a balance between goal-directedness and probabilistic behavior that is efficient and psychologically satisfying to the user.

These conflicting constraints affect the relationship between biasing the system and its resulting tutoring effectiveness: adding more bias reduces the time between important tutoring steps and reduces the time between steps in the tutorial lesson, but does not necessarily translate into greater overall teaching effectiveness.

The system uses a process based model simulation language, although some features of object based models have been incorporated. The representations supported by this type of simulation language can be easily related to plans and the resulting reasoning strongly complements what can be performed in traditional planners. An object based model (Simuletter 88) results from organizing the simulation language around domain objects. The behavior of each object is a process. More advanced models allow objects to respond to messages sent by other objects. A process based model is an extension of the event based model [Kiviat, 1973]. The basic unit of modularity represents an ongoing process, not just an instantaneous event. Primitives in the simulation language provide for synchronization among multiple processes and the ability to wait for events or state changes to occur. In practice, the synchronization primitives cause affected processes to be stored in the event queue, in much the same way that a suspended task waits for CPU time or IO events to occur. A process model looks much like a concurrent program, but all delays and scheduling refer to the simulated clock and not to real time.

The Biasing Mechanism

The biasing mechanism was based on: 1) a non-cooperative mixed initiative planner designed and implemented for a complex non-deterministic domain, and 2) enabling the planner to alter the simulation without destroying its realism. In addition, the biasing mechanism operates efficiently enough to be used within a real-time simulation.

For the purpose of this discussion it is sufficient to think of the planner as a directed graph with arcs for user actions and state changes. Nodes represent periods when no qualitative changes occur; those which are expected to best support tutoring are selected and ranked as goal nodes. The time needed to reach a goal is one of the factors to be optimized so it is a cost function associated with nodes in the graph. If the user does nothing then eventually the simulation will follow one of the arcs to a new state. The time this will take defines one probability distribution constrained by the domain model. Alternatively, the user may take some action altering the relative likelihood of successor states and resulting in a faster state change. Combining the user model and domain model gives an overall probabilistic model of how likely each state change is and and when it will occur.

Some of these probabilities define the domain model and so are under control of the tutor. The planner will alter these probability distributions to reach a goal node while simultaneously minimizing the cost and the amount of probability alteration. Maximizing the priority of the goal reached is a further constraint.

Goals in this directed graph correspond with simulation states that have been identified as providing an opportunity for useful tutoring actions. Several interrelations among these goals exist: several goal states in the graph may satisfy a single tutoring goal, tutoring goals may fail even though the simulation reaches the chosen state and all tutoring goals need to be satisfied eventually. Several measures were specified to make this problem well formed. In particular, there must be a way to specify how much the domain probabilities have been altered and a function to combine this measure with the cost of arc traversal to obtain a global measure of the utility of a proposed solution.

We call the degree of domain probability alteration "improbability." Entropy is a measure of randomness so conceptually these concepts are opposites like heat and cold. It is important to realize that improbability is defined procedurally, like a random number, rather than logically, like a prime number. It is a measure of how much the procedure for generating a simulation has been changed and cannot be determined from the results of the simulation alone. For example, the number "5" is a random number if it is the result of a fair roll of dice but it is not a random number if it is the result of computing the sum of the first two prime numbers since that computation always yields the same answer.

Improbability is defined as the combined changes to the simulation. The problem is reduced to defining a measure of improbability for each random node and defining a combination function. The biased random node is created by changing the probabilities. The sum of the amount by which each probability is increased is used as the measure of improbability for a random node. For example, changing a random node from a 50-50% outcome into a 70-30% outcome has a cost of 20 points of improbability. Note that this definition counts only the amount by which the probability of an outcome has been increased. The intuition behind this is that randomly choosing an outcome that has been made less likely cannot make the simulation seem more improbable. The sum of the absolute differences is another intuitively attractive formula but not significantly different since there is clearly only a constant factor of two difference between the sums. Once the required utility function is defined we still need a mechanism for computing an optimal (or satisfactory) set of improbabilities. Theoretically we may consider all possible ways of altering the probability distributions in the simulation, but in practice a more efficient search technique is devised.

Both improbability and the time needed to reach a goal are minimized in the working system. The simultaneous optimization of two variables often requires a tradeoff; in this system we impose an upper limit on the improbability, keeping the simulation realistic, and then attempt to improve the tutoring time as much as possible.

The Cardiac Tutor: A Case Study

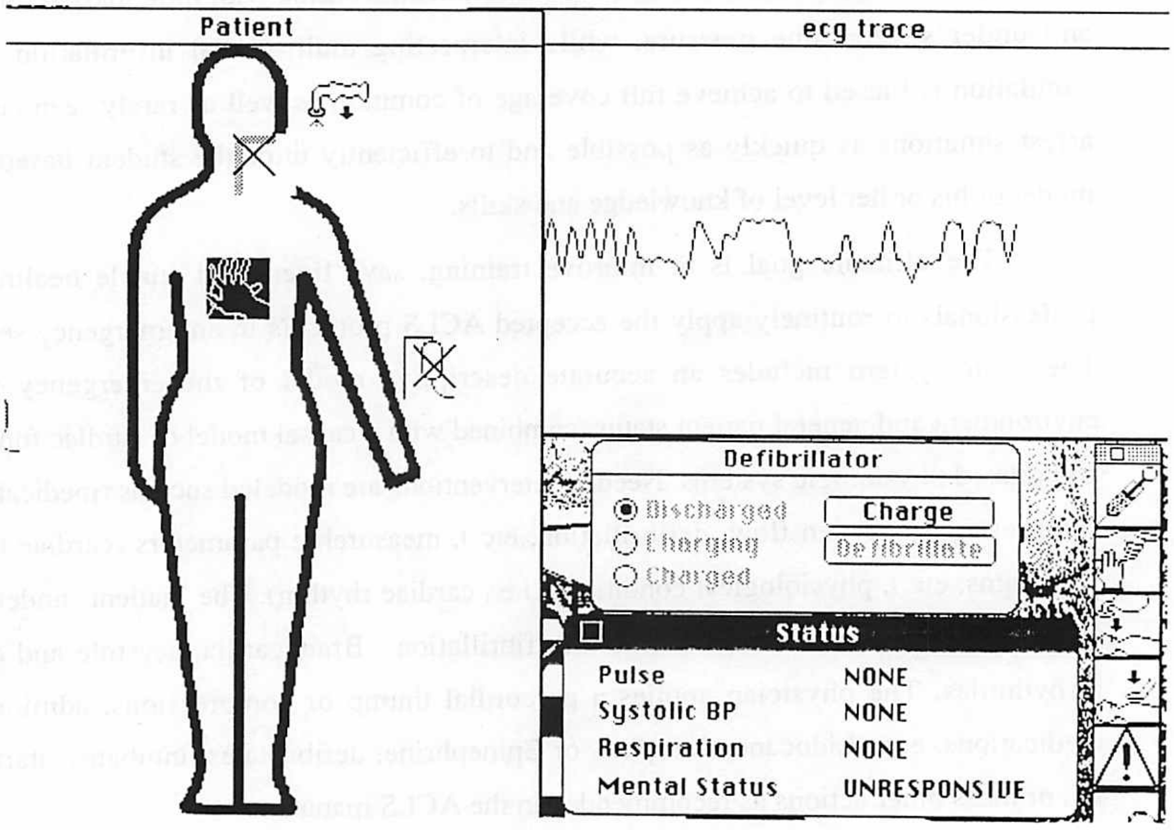
The issues discussed above were studied in the context of building a system for training medical professionals to lead cardiac resuscitation teams. The system contains an intelligent interactive simulation of the patient, emergency room team and hospital environment, see Figures 1 and 2. The simulation provides a realistic exposure to the task and allows physicians and other medical personnel to practice realistic simulated treatments. The simulation presents a continually changing context automatically updated. Its progress is partly under control of the user (through intervention selection) and partly dictated by clinical reality.

Emergency medical techniques for acute cardiac failure saves many lives. Yet, proper training for the Advanced Cardiac Life Support (ACLS) team leader requires approximately two years of closely supervised training in an emergency room, ambulance, or similar medical facility. The cost of this training period is high, both in patient care and for the health industry. Team leaders, e.g., paramedics, EMTs, and resident physicians, have different prior levels of expertise, yet each must perform the same difficult cognitive tasks in a noisy environment, working with little margin for error and under severe time pressure, while interpreting multi-modal information. The simulation is biased to achieve full coverage of common as well as rarely seen cardiac arrest situations as quickly as possible and to efficiently tutor the student based on a model of his or her level of knowledge and skills.

The ultimate goal is to improve training, save lives, and enable health care professionals to routinely apply the accepted ACLS protocols in an emergency setting. The basic system includes an accurate descriptive model of the emergency room environment and general patient status, combined with a causal model of cardiac function and related physiologic systems. Needed interventions are modeled such as (medications, compressions, oxygen flow, defibrillation, etc.), measurable parameters (cardiac trace, vital signs, etc.), physiological conditions (i.e., cardiac rhythm). The `patient' undergoes simulated cardiac arrest, e.g., ventricular fibrillation, Bradycardia, asystole and other arrhythmias. The physician applies a precordial thump or compressions, administers medications, e.g., Lidocane, Atropine, or Epinephrine, defibrillates, intubates, starts an IV, or takes other actions as recommended in the ACLS manual.

The high level simulation coordinates the multimedia modules and corresponds closely with the ACLS protocol both in terms of level of detail and structure. Generally, it is a descriptive model rather than a causal model and is synchronized with the real-time clock of the computer. Events within the simulation drive the sound and graphic routines, and pass information up to the tutoring module. Student input is treated as events within the simulation. The low level causal simulation can be called directly from the high level simulation when a substantial physiologic change is expected.

Alternatively the steady state description of the standard physiological states can be computed using the causal simulation in an off line mode with the results stored and recalled as needed. Medical interventions by the team leader are mediated by the computer and compared to the accepted protocols. The computer offers automated tutorial help in addition to recording, restoring, critiquing and grading student performance. It customizes the simulation to suit different previous levels of achievement and assists the learning process dynamically. Good or improved performance is noted with positive feedback; incorrect behavior is categorized and commented upon.



- [✓ Discharged]
- Charging
- Charged

Pulse	NONE
Systolic BP	NONE
Respiration	NONE
Mental Status	UNRESPONSIVE

Figure 1: The Cardiac Tutor, showing the patient, computed cardiac rhythm, and several intervention windows

Computer generated graphics provide a real time "ECG trace" based upon the causal cardiac model, see Figure 1. Other pertinent information is shown using a schematic patient diagram, including current interventions. Both background sounds (emergency room noise) and foreground sounds (shouted responses) are generated by computer and can be played through headphones. When the tutor is used by a student in a public space or library, input can be accepted using a graphic input device (mouse). When the tutor is used at home or in a laboratory, students can speak commands within a restricted set of natural language phrases through a microphone attached to head gear.

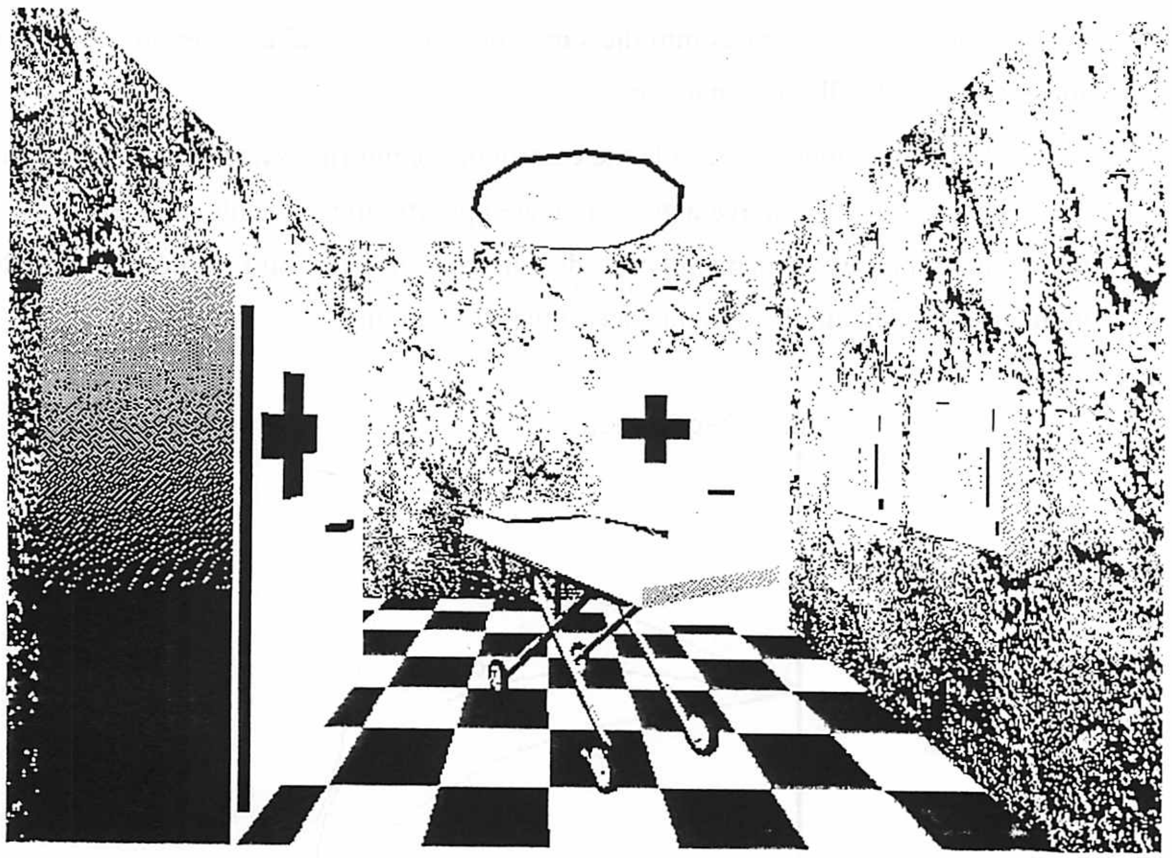


Figure 2: The Simulated Emergency Room

Several crucial problems were solved on route to developing this biasing mechanism. 1) Domain knowledge was encoded to enumerate the entire curriculum. Covering the required topics is a goal of the tutoring system. The relative importance and difficulty of each topic is represented, so the system is able to reason about the priority of tutoring goals. 2) The student's understanding of each topic is represented and the user model updated in response to correct or erroneous student interactions with the simulation. A simple overlay model of the user determines which topics remains to be covered. 3) The system reasons about the medical constraints of the domain and forms goals to teach specific domain topics. However, each topic requires the simulation to reach a certain state. Hence the system finds ways to drive the simulation into a state where a relevant tutoring subject was one of the outstanding topics. The planning process satisfies three high level constraints: 1) the simulated patient's underlying physiology generated by the system must be physiologically and clinically realistic; 2) a limited number of events can happen, given any state in the simulation, subject to alterations caused by the bias, and given that each outcome is determined by a known probability;

and 3) the system integrates into the simulation any medical intervention applied by the student in a medically reasonable way.

Student actions are outside the control of the tutoring system and frequently upset the system's plans to achieve a goal or create opportunities to achieve other goals. Thus, the planning process dynamically reorders its goal priorities in response to both changes in the user model and the current state of the simulation.

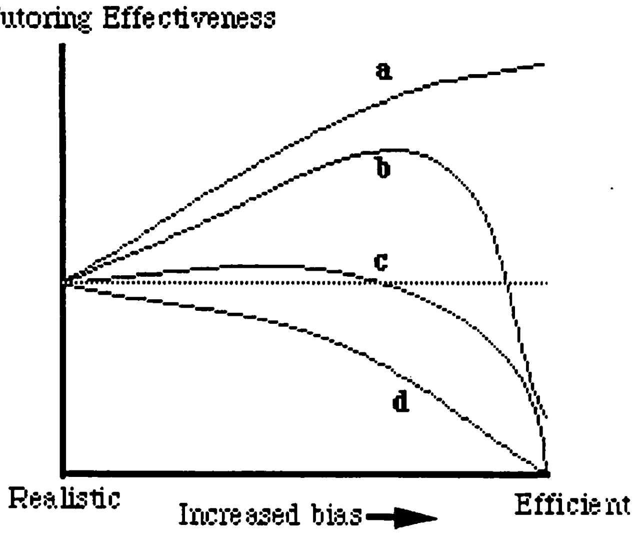


Figure 3: Simulation Bias and Tutoring Effectiveness

Future Work

The tutor has been successfully used by half a dozen physicians and nurses, and formative evaluation is proceeding. We intend to explicitly evaluate the relationship between tutoring effectiveness and simulation bias. Currently this relationship is unknown, and may be directly related, inversely related or be related through more complex formulae positive, see Figure 3. Curve a shows steady improvements in the overall tutoring effectiveness as the simulation is increasingly biased. Curves b and c show improvements up to some limit, followed by declines. A minuscule improvement in tutoring effectiveness as seen in curve c may not be enough to justify the expense and complexity of the biasing mechanism. Even worse adding bias may simply cause a decrease in tutoring effectiveness as shown in curve d. A steady improvement as shown in curve a would be ideal but is not expected. The expected result is that the relationship between overall tutoring effectiveness and increased simulation bias will be similar to curve b. Results similar to curves c or d would refute the hypothesis behind this research.

Many other research questions remain, including several psychological questions. Central among these is how a student perceives biasing the simulation. Potentially, we anticipate several effects that might diminish or eliminate the theoretical advantages of the controlled simulation. For example, the simulated patient might seem "unreal" to the student. Even worse, the student might develop erroneous expectations about the likelihood of various events in the domain. During the next few months, the tutor will be tested with numerous physicians, nurses, EMT, and other emergency room personnel to provide a quantitative evaluation of the relationship.

Previous Work

Current computational models of reasoning in tutor systems focus on generalized stereotypes, a kind of "dead reckoning" to assess student learning or a bug library to anticipate procedural bugs and generate appropriate instruction [Van Lehn, 1988]. These models are both too simplistic and too static to reason effectively about human learning, which is uncertain, inconsistent, and highly variable. Such results are often formulated in a deterministic manner based on experience with long term protocols of human activities [Spada, 1992]. User modeling is still an area of active debate [Clancy, 1989; Brown et al., 1988, Katz et al., 1992]. Katz et al. distinguish between researchers who are "model-builders" committed to explicit representations of user models and "model-breakers" who doubt the possibility or usefulness of such models. We concur with Katz [Katz et al., 1992] that a user model should be used "without interfering with the opportunities that the learning environment affords students for experimentation, reflection, and self-diagnosis." For example, Sherlock II uses a substantial lattice of fuzzy variables to model the student [Katz et al., 1992]. The aim of this representation is to explicitly represent uncertainty in the user model. A vague description of the student, it was argued, would be a more truthful representation, and would better support the tutor's interactions. However, this approach ignores the central issue; what to do about erroneous assumptions made about the student at least some of the time.

There are many parallels between this work on biasing the simulation and research into customized explanation. In both cases the goal is to present "correct" information to a user based upon the system's understanding of what the user wants and needs to know [Suthers et al., 1992]. It is possible to generate explanations that are correct relative to any given set of facts about a user (perhaps) but this is of no great value since much of what we need to know cannot be determined with confidence and precision. In both cases the system needs to be designed with a "fail soft" principle in mind; unavoidable errors in the user model must not cause egregious misbehavior as a result. Early results with medical reasoning systems provides the clearest indication of how difficult is the presentation of coherent explanations [Szolovitz, 1979]. Canned-text and simple translations of execution traces do not provide high quality explanations. Work on knowledge representation [Michner, 1978] and reasoning [de Kleer, 1979; Patil, 1981] illustrate how multidimensional representations are needed for full use of a system's knowledge knowledge.

Many tutoring systems present the student with a realistic simulation of a working environment and the tools normally used to manipulate that environment. Other systems use a simulation but do not attempt to provide a realistic one. For example, the CIRCSIM-TUTOR [Khuwaja, 1990] provides a "prediction table" which is filled by a student and compared with results generated by a circulation simulation. Filling this table is not a natural domain task but forces the student to develop abilities that are thought to be crucial in real domain tasks. Recently there has been interest in combining artificial intelligence and simulation techniques [Elzas, 1986].

Classical AI planning solves the problems they were designed to handle, but never found much practical use, and the field was largely dormant until the mid 1980's. The fundamental problem with these earlier systems is the assumption that the world is static. This assumption is obviously false. Many aspects of the world change dynamically, and a planner should include alternative reasoning about plans and be capable of making changes in existing plans. People constantly make plans and predictions that are useful without being able to fully understand the consequences of their actions. Developing ways to program this ability, at least in part, was one goal of this research.

References

Broverman, C., "Constructive Interpretation of Human-Generated Exceptions During Plan Executions," (Ph.D. Thesis), Computer Science, University of Massachusertts, Amherst, available as Technical Report 91-9 (1991).

Broverman, C. A. and Croft, W. B., "Reasoning About Exceptions During Plan Execution Monitoring", *Proceedings of the Sixth National Conference on AI* (AAAI-87), Seattle, Washington. Morgan-Kaufman: Los Altos, Calif. (1987).

Brown, J. S., Collins, A., Duguid, P., "Situated Cognition and the Culture of Learning," Institute for Research on Learning report no. IRL 88-0008 (1988).

Chapman, D., "Planning for Conjunctive Goals," *Artificial Intelligence*, Vol. 32, pp. 333-377 (1987).

Clancey, W. J., "Viewing Knowledge Bases as Qualitative Models," IEEE-Expert (1989).

Cohen, P. and Gruber, T. "Reasoning about Uncertainty, a Knowledge Representation Perspective," Computer Science Technical Report 85-24, University of Massachusetts at Amherst (1985).

de Kleer, J., "Causal and Teleological Reasoning in Circuit Recognition", MIT-AI-TR 529, Cambridge (1979).

Elzas, M. S., Oren, T. I. and Zeigler, B. P. (editors), *Modeling and Simulation Methodology in the Artificial Intelligence Era*, North Holland: Amsterdam (1986).

Fennema, C.L., Jr. (1991). "Interweaving Reason, Action and Perception." Ph.D. Dissertation, Computer Science Dept., University of Massachusetts at Amherst. Available as Technical Report 91-56.

Katz, S., Lesgold, A., Eggan, G. & Gordin, M., "Modeling the Student in Sherlock II", *AI and Education* Vol. 3, No. 4 (1992).

Khuwaja, R. A., Evens, W. E., Rovick, A.A., and Michael, J.A., "Knowledge Representation for an Intelligent Tutoring System based on a Multilevel Causal Model," (1990).

Kiviat, J. P. C., Villanueva, R., and Markowitz, H. M., "Simscript II.5 Programming Language", *CACI*, Los Angeles, California (1973).

Michner, E. Rissland., The Structure of Mathematical Knowledge, AI-Tech-Report No 472, (1978).

Murray, T. and Woolf, B. (1991). "A Knowledge Acquisition Framework for Intelligent Learning Environments." Feature article in *SIGART Bulletin*, 2: 1-13.

Murray, T. and Woolf, B. (1992). "Tools for Teacher Participation in ITS Design." *Proceedings of the Second International Conference on Intelligent Tutoring Systems*. Montreal, Quebec, June 1992, pp. 593-600.

Patil, R. S., "Causal Representation of Patient Illness for Electrolyte and Acid-Base Diagnosis", (Dissertation) MIT/LCS/TR-267 (1981).

Self, J. (1987). "User modeling in open learning systems." In *Tutoring and Monitoring Facilities for European Open Learning Environments,* J. Whiting & D. Bell, Eds. Elsevier: Amsterdam.

Spada, H. (1992). "Empirical Validation of Student Models," *Journal of Artificial Intelligence and Education,* Vol. 3 (4).

Suthers, D., Woolf, B. and Cornell, M. (1992). "Steps from Explanation Planning to Model Construction Dialogues," *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92),* pp. 24-30.

Szolovits, P, "Artificial Intelligence and Clinical Problem Solving," MIT/LCS/TM-140 (1979).

Van Lehn, K. (1988). "Student Modeling," in *Foundations of Intelligent Tutoring Systems,* Erlbaum: Hillsdale, NJ.