Towards a Computational Model of Tutoring

Beverly Park Woolf

COINS Technical Report 91-38

April 1991

To appear as a chapter in the NATO Workshop volume *Foundations and Frontiers in Instructional Computing Systems*, P. Winne and M. Jones, eds., Springer-Verlag, NY.

Towards a Computational Model of Tutoring[1]

Beverly Park Woolf

Department of Computer and Information Science, University of Massachusetts
Amherst, Massachusetts 01003, U.S.A.

Abstract: This article addresses several issues around successful integration of instructional science and computer science. It addresses issues of building computational models of tutoring and incorporating instructional principles. The first barrier to overcome towards this integration is development of principled programs in which cognitive principles about learning and teaching are realized at a level of granularity consistent with building computational models. Such cognitive studies would facilitate fine-grained modeling of learning and teaching. The second barrier to overcome is the gap between the two disciplines, in terms of goals, motivations, literature, and even defining concepts. This situation suggests that a large effort should go into two areas: research on understanding basic principles behind learning and teaching and the establishment of clearer lines of communication between instructional and computer scientists. This article addresses both these issues.

1 Integration of Instructional Science and Computer Science

This article addresses several issues around successful integration of instructional science and computer science. Combining the efforts of these two disciplines offers exciting possibilities for the scientific study of human learning and the building of powerful teaching systems; yet, as a research community we are only at the very beginning of asking whether such an integration is possible and of knowing how to develop one. This integration would move beyond the paradigm of either separate discipline and would create results more powerful than those obtainable separately or in combination. To assume that such an integration is impossible

[1] Copyright © 1990, Beverly Park Woolf. Permission granted to copy for non-profit academic purposes.

reflects a narrow perspective and a lack of learning from the recent history of science which has seen many `impossible' achievements become reality.

This article describes conceptions about such an integration and addresses issues of building computational models of tutoring and incorporating instructional principles (e.g., low-level descriptions and analyses of learning and motivation). The first barrier to overcome towards this synthesis is development of principled programs in which cognitive principles about learning and teaching are realized at a level of granularity consistent with building computational models and these principles are integrated into computer systems. Such cognitive studies facilitate fine-grained modeling of learning and teaching and should ultimately provide the `zorch' that will make possible the building of powerful instructional systems. Other facets of advanced technology, e.g., multi-media or increased speed and power, including 200 MIPs (million instructions per second), will not significantly empower *tutoring* systems.

The second barrier to overcome is the gap between the two disciplines, in terms of goals, motivations, literature, and even defining concepts. Instructional scientists frequently work on issues and goals that are not addressed by computer scientists, and vice-versa. Instructional scientists do collaborate with computer programmers, but the latter are not typically versed in cutting-edge research developments in computer science. Computer scientists, on the other hand, do not strive to develop general design methodologies that will support participation of instructional designers in their efforts. Frequently computer scientists build tutoring systems in collaboration with domain experts; however, they rarely collaborate with practicing instructional scientists in the design of these systems.

This situation suggests that a large effort should go into two areas: research on understanding basic principles behind learning and teaching and the establishment of clearer lines of communication between instructional and computer scientists. This article addresses both these issues. The first issue discussed is the articulation of elements of cognitive principles for use in computational models. Cognitive modeling is directed at identifying computationally precise explanations of how cognition works, and, in this case, how, why, and when students learn. The true bottleneck in building powerful instructional systems today lies in our inability to fully understand and represent human learning and teaching processes.

The second issue discussed in this article is how computer scientists might clarify the process of building of knowledge-based tutors so that instructional designers and others might collaborate in their design and implementation. Researchers in computer science have not generally developed their technology with an eye toward including teachers and instructional scientists. Yet, the experience of learning via a knowledge-based learning environment is so novel for both student and teacher that neither computer scientist nor instructional designer acting independently can foresee all the crucial learning and teaching issues that might arise when these systems are actually used. Gleaning insights, principles, and rules about teaching that will be encoded within knowledge-based tutors must originate from a rich synthesis of learning and instructional theories, insights from practicing teachers, and on-line experimentation. Testbeds are needed for systematic integration of instructional design theory and advanced technology.

This article first examines some motivations for the development of knowledge-based systems and then describes salient success stories. Next, it looks at instructional research issues. Finally, it identifies contributions from researchers at the University of Massachusetts working to acquire, represent, and reason about tutoring knowledge.

2 Motivation for Building Knowledge-based Systems

Research into knowledge-based tutoring systems has created a great deal of interest on the part of academe, industry, and the military internationally. Tough educational problems, including outdated educational facilities, lack of public support for educational innovation, and in some cases inappropriate policies have left educators engaged in an uphill battle to reverse deficiencies in student learning and teacher training. Long distance learning, learning for the disabled, and a variety of non-traditional educational needs have spawned new educational research efforts.

Poor education in the United States has contributed to low-level entry abilities of industrial and military recruits and difficult retraining problems. Industry spends a tremendous amount on training. IBM, for instance, spends $900 million annually directed at training nearly 400,000 employees in 130 countries [Bowscher, 1989]. Industrial and military training is clearly faced with difficulties in the area of recruitment and selection of well prepared workers [Goldman, 1984] and the military has, additionally, a two-year turnover problem for new recruits.

These and other problems have focused attention on the possibilities offered by knowledge-based systems. The clear emergence of architectures for knowledge-based tutoring system and positive training results have produced the feeling that progress is being made. Indeed, several systems have achieved the two-sigma effect [Bloom, 1984], which is the same improvement in learning that results from one-on-one human tutoring over classroom tutoring. Several success stories have been described in which students using knowledge-based tutors learned knowledge and skills in one-third to one-half the time it took for a control group to learn the same material [Shute, 1990].

In one special case, students working with an Air Force electronics troubleshooting tutor for only 20 hours gained a proficiency equivalent to that of trainees with 40 months (almost 4 years) on-the-job training [Lesgold, Lajoie, Bunzo & Eggan, 1990]. In another example, students using a Lisp tutor at Carnegie-Mellon University [Anderson, 1990] completed programming exercises in 30% less time than those receiving traditional classroom instruction and scored 43% higher on the final exam. In a third study, students using a microworld environment learned general scientific inquiry skills and principles of basic economics in one-half the time required by students in a classroom setting [Shute, Glaser & Raghavan, 1989].

Given these results, one might ask why more intelligent tutors are not being used and why existing systems are not more effective. One reason relates to the lack of artificial intelligence development tools, such as shells and frameworks, similar to the shells used to build rapidly expert systems. Tools would facilitate large-scale development; and a simple tool, such as a simulation tied to an expert system or to a lock-step tutor, might be a practical way for a designer to get started on a path of incremental design through feedback from the user. Some researchers suggest that a teacher should interact with a variety of tools, much as a conductor might orchestrate a suite of instruments.

Other reasons for the slow adoption of new systems include the need to reduce cognitive task analysis to engineering practice and the difficulty of developing cognitive models, see Section 3.1. Additionally there is the need to make widely available certain new knowledge representations (i.e. qualitative simulations) which will be more effective than those offered by first-generation expert system tools. An additional barrier is the lengthy development cycle required before a system can move from research lab to a salable product.

3 Instructional Design Issues

The first major issue addressed by this article is the need to identify instructional science principles at a level of granularity that supports the building of computational models of learning and teaching. Instructional design issues present serious challenges to the current generation of researchers. This section discusses such challenges and describes possible *solutions* to the representation of student intentions, the support of knowledge reflection, and the engineering of tools for diagnosis.

3.1 Cognitive Modeling

Cognitive modeling is a methodology for developing computational explanations of cognition. It seeks a low-level description of cognitive processes -- in this case of learning and teaching -- which permits the encoding of rules and principles in a computer. Increased use of modeling is supported for the building of knowledge-based intelligent tutors and its use has strong advocates among psychologists and computer scientists.

Cognitive modeling for building knowledge-based systems should be increased in three areas, primarily in (1) development of pedagogical and subject-matter theories, (2) design of instruction, and (3) delivery of instruction. Of these phases, the design of instruction is the one that seems to have achieved the most direct benefit for knowledge-based systems, including substantial benefits from modeling subject matter experts. For instance, Anderson et al. [1990] attribute much of the success of their tutors to the cognitive task analysis of experts in Lisp, geometry and algebra.

Work on modeling good teachers and tutors has only just begun (with the exception of a few early classics, such as the work of Stevens and Collins on Socratic tutoring [1977]). VanLehn expects this line of investigation to pay off at least as well, if not better, than the modeling of experts and learners [Woolf et al., 1991].

Of the three phases of pedagogical work, the actual delivery of instruction is the area where cognitive modeling has found the least fruitful application. Mostly, this is due to ahistorical accident. In most systems to date, teacher models have been weaker than expert models and student models. Although a good teacher model might compensate for an impoverished expert or student model, experience has shown that strong expert and student models require a decent teacher model for the system to be effective.

VanLehn underscores the fact that modeling is just good engineering practice, regardless of whether one is building a hydroelectric dam or a science course [Woolf et al., 1991]. With tongue in cheek, he suggests that if students could sue malfeasant instructional developers, cognitive modeling would be much more common since it is so obviously effective.

3.2 Representing Student Intentions

Great instructional leverage will come from implementing student and domain models based on cognitive modeling. Where problem solving is seen as a primary learning activity, additional power will come from understanding the cognitive processes necessary to complete the task and then reification of these processes in the domain and student models. Building such models and linking them to a powerful interface can only be accomplished after a thorough task analysis of the domain has been achieved.

Developing an effective instructional model of the domain implies that a designer has generated a conceptualization and structuring of the task and the learning process. This instructional model may lead to finding ways to represent abstract cognitive structures, such as student goals and plans, and representing6these structures as concrete visual or textual objects that can be made the focus of student action and discourse [Singley, 1990].

For example, a system might maintain a list of student subtasks. In a geometry environment, it might advise a student to first discover if two sides of a triangle are equal and then show that the triangle is isosceles. In a calculus environment, it might suggest use of the chain rule and indicate the two subtasks: differentiate the first function multiplying it times the second and then differentiate the second multiplying it by the first. In each environment, the system might post and display each subtask as boxed or shaded to indicate which subtasks are active and which have been satisfied, respectively [Singley, 1990].

Tools built into an interface should support a student in expressing his/her plan directly to the machine, and then these tools might become the structure of the tutorial discourse. Tools might be offered at varying instructional levels as in the case where an algebra tutor offers tools at a low level (e.g., commands to add, divide, or multiply terms) and also at a higher level (e.g., commands to group variables, simplify equations, or substitute values).

In this way, the system might infer the intentions of the student from expressed plans. To build such tools requires a deep understanding of the domain, and it may be difficult to find a transparent representation for goals and plans in some domains. Any system which requires that the student solve a problem by using a planning icon unrelated to the problem solving domain, might be burdening the student with additional complexity and forcing the learning of a formalism that is nearly as difficult as mastering the domain itself.

Well-crafted tools, on the other hand, will support a student in operating in the physical problem space and will be fully integrated with that space [Singley, 1990]. Excellent tools will also permit improved knowledge reflection or meta-cognitive activities involving conscious surveillance and self regulations. Instead of facilitating the use of a series of questions and answers, the availability of appropriate tools will support a tutor in making the student's thinking public for discussion. Questions from the tutor might then refer to student plans and expectations. Alternative high and low level languages might accommodate a user's shift in understanding [Dillenbourg, in this volume] and would facilitate a comparable shift in tutorial discourse.

Another unresolved instructional design issue is the effect of immediate feedback on learning. Corbett and Anderson [1990] note that students seemed more confident with less feedback. (They also note that this confidence, at least in the domain of Lisp programming, was unrelated to competence, specifically in performance on a post test.) Schooler and Anderson [1990] record several disadvantages associated with immediate feedback: 1) The student grows dependent upon feedback and hesitates to act without it; 2) The student does not develop error correction and detection abilities; and 3) The feedback competes for the student's limited short-term memory resources. The jury is still out on a final evaluation of the effect of immediate feedback.

Another instructional issue which requires attention is the identification of teaching goals for these systems. Difficult yet desirable goals include encouraging a student to gain confidence, passion, and ownership of their work. Yet current systems are a long way from achieving this goal. Another goal might be to support students in becoming researchers capable of independent exploration through networks of available knowledge.

3.3 Student Models and Error Diagnosis

Student modeling, an essential component of any tutoring system, is still incompletely understood. Effective student models should be usable, faithful, and runnable. They should provide descriptions of learning at a level of granularity that facilitates the encoding of principles and rules in a teaching system.

Current student models are shallow representations of topics a student does or doesn't know, or those topics about which the systems doesn't have enough information. In fact, a robust student model should record the role of context in student learning, student learning preferences, motivation, history while acquiring past learning, and meta-knowledge about learning. AI researchers will wait a long time for such descriptions to evolve; of course, they might participate with instructional designers in efforts to develop such computational models of learning.

Research into evaluating automatic diagnosis of common errors and plausible misconceptions is another instructional arena that currently receives insufficient attention. In limited domains we have accomplished diagnosis [Johnson & Soloway, 1984; Anderson & Reiser, 1986], yet we still do not know to use this diagnostic material in subsequent dialogue to empower students.

In general, researchers are moving away from building omniscient tutors capable of detecting all possible errors and misconceptions. Instead, research is now focused more on building empathetic partners that choose from among several forms of interaction based on the content of the communication and the needs of the student [Woolf, 1988]. Possible communication styles include didactic explanation, guided discovery learning, coaching or coaxing, and critiquing. Although no one style is preferred, different tutorial applications will be better addressed with a given primary style.

4 Representing and Reasoning about Tutoring Knowledge

In addition to trying to develop cognitive models of teaching and learning, this article also addresses the need for computer scientists to clearly define the structure and design of their systems so that instructional scientists can become their collaborators in building knowledge-based tutoring systems. This section describes efforts at the University of Massachusetts along this line; it clarifies how tutoring can be understood in terms of the artificial intelligence paradigm of knowledge and control and shows how to represent knowledge computationally and how to express it as strategies and rules. Architectures and tools are proposed for developing new systems and for supporting the knowledge representation and acquisition process. The tools are now used in a generic and consistent foundation which has enabled us to represent, acquire, and reason about tutoring knowledge across several domains and from within several sites. Our goal is to enhance this framework and ultimately to produce systems in which psychologists, instructional scientists, and domain experts can work as our colleagues to modify and upgrade tutors without the need for knowledge engineers.

This section describes tools that facilitate both identifying tutoring knowledge and representing it. *Knowledge representation* is explained in terms of modeling domain knowledge, human thinking, learning processes, and tutoring strategies. A uniform language is proposed for storing tutoring primitives, including lessons, topics, and presentations. *Knowledge acquisition* is described as a methodology for identifying and encoding the expertise used by teachers to reason about tutoring. *Control knowledge* is explained in terms of the machine's ability to select a topic or response for an individual student and then to customize its discourse and dynamically modify its examples, questions, or descriptions for that student.

4.1 Building a Tutoring System

We have evolved a generic and consistent foundation for representing, acquiring, and reasoning about tutoring knowledge. The big payoff has been that we can now apply the framework and evolving theory to several domains. We are not invested in promoting a particular tutoring strategy, nor do we advocate a specific knowledge-based tutoring system design. Rather, we build tools that allow for a variety of system components, teaching styles, and intervention strategies to be combined into a single framework. For example, Socratic tutoring, incremental generalizations, and case-based reasoning are just a few of the teaching strategies we have experimented with using this framework. Ultimately, we expect the machine to reason about its own choice of intervention method, to switch teaching strategies, and to use a variety of tactics and teaching approaches, while making decisions about the most efficacious method for managing one-on-one tutoring.

Development Cycle for Artificial Intelligence Systems. Development of knowledge-based tutors, like development of any artificial intelligence system, requires several iterative cycles: computer scientists and instructional designers first collaborate on the design and development of the system, additional collaboration is required to test the system with students, and then the original implementation is modified and refined based on information gained through testing. This cycle is repeated as time permits.

For example, a professor at City College of San Francisco used the statics tutor (Section 4.2) in a classroom and noticed weaknesses in the simulation's ability to inform the student. She augmented the system with verbal discourse, adding examples or explanations, making diagnoses, and clarifying system response. She gave us a list of her additional discourse moves to be incorporated into the next version of the tutor.

Representation and Control. Artificial intelligence programs require that a teaching expert define the knowledge to be used along with the control structures which define the way an interpreter will traverse that knowledge. Knowledge representation refers to how such knowledge is stored by a system to allow it to model the domain, human thinking, learning processes, and tutoring strategies. Knowledge bases might store concepts, activities, relations between topics, and other quantities needed to make expert decisions. In tutoring, they might store a variety of lessons, topics, presentations, and response selections available to the tutor (see Figure 1). Control refers to passage of an interpreter through those knowledge bases and its selection of appropriate pieces of knowledge for making a diagnosis, a prediction, or an evaluation. For tutoring, control structures might be specified at the four levels indicated in Figure 1, separately defining control for selection of lesson, topic, presentation, and response selection.

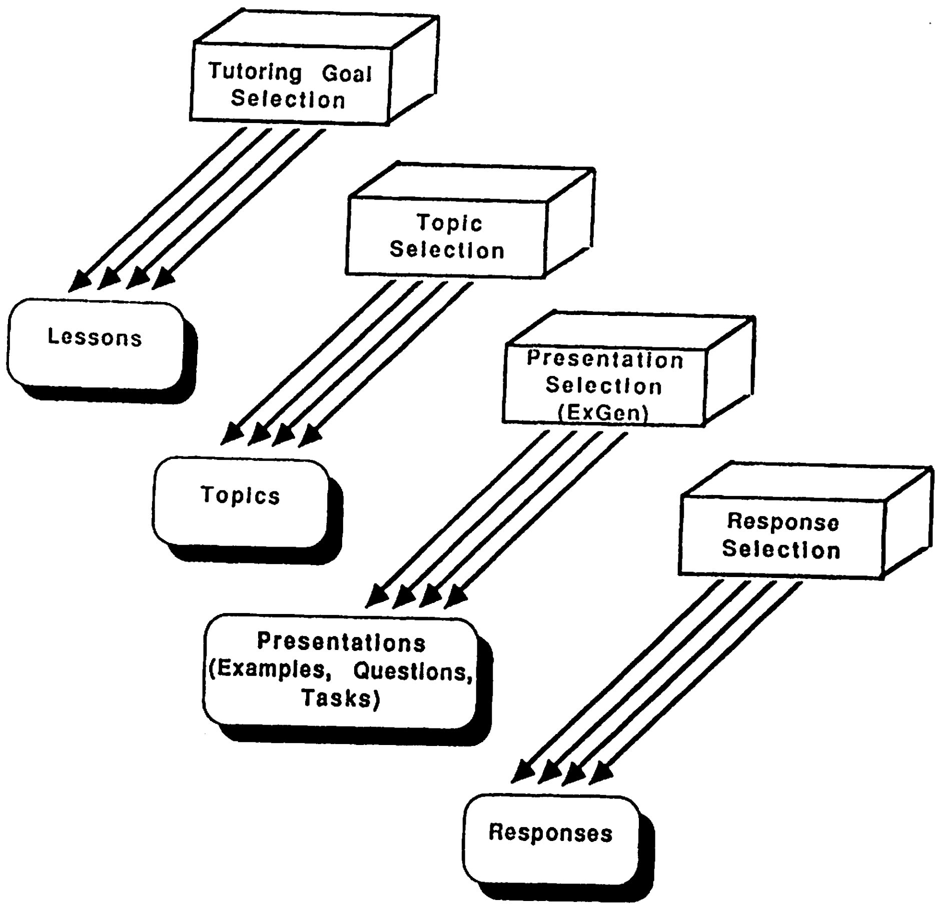


Figure 1: Representation and control in a tutoring system.

Currently, our control structures are motivated by specific instructional and diagnostic goals; thus, for example, one control structure produces a predominantly Socratic interaction and another produces interactions based on presenting incrementally generalized versions of new concepts or examples. Control structures are specific to a particular level of control and are used separately to define the reasoning to be used for selecting a lesson, topic, presentation, or response.

Acquiring and encoding this large amount of knowledge, or the knowledge acquisition process, is difficult and time consuming. We have built a number of tools that facilitate representing, acquiring, and reasoning about tutoring knowledge (see Figure 2). For each knowledge base (lessons, topics, presentation, or response) we consider the nature of the knowledge that must be accessed, such as the examples or questions (from the presentation knowledge base) or the activity the tutor must engage in, such as to motivate or teach a topic, or to provide follow-up. We have built tools, shown at the bottom of Figure 2, to support most activities listed in the figure. Only a few such tools will be described in this article, namely TUPITS, the Response Matrix, and DACTNs.

We divide the discussion into two parts, separately describing tools for representing tutoring primitives (lessons, topics, and presentations) and then tools for representing discourse knowledge.

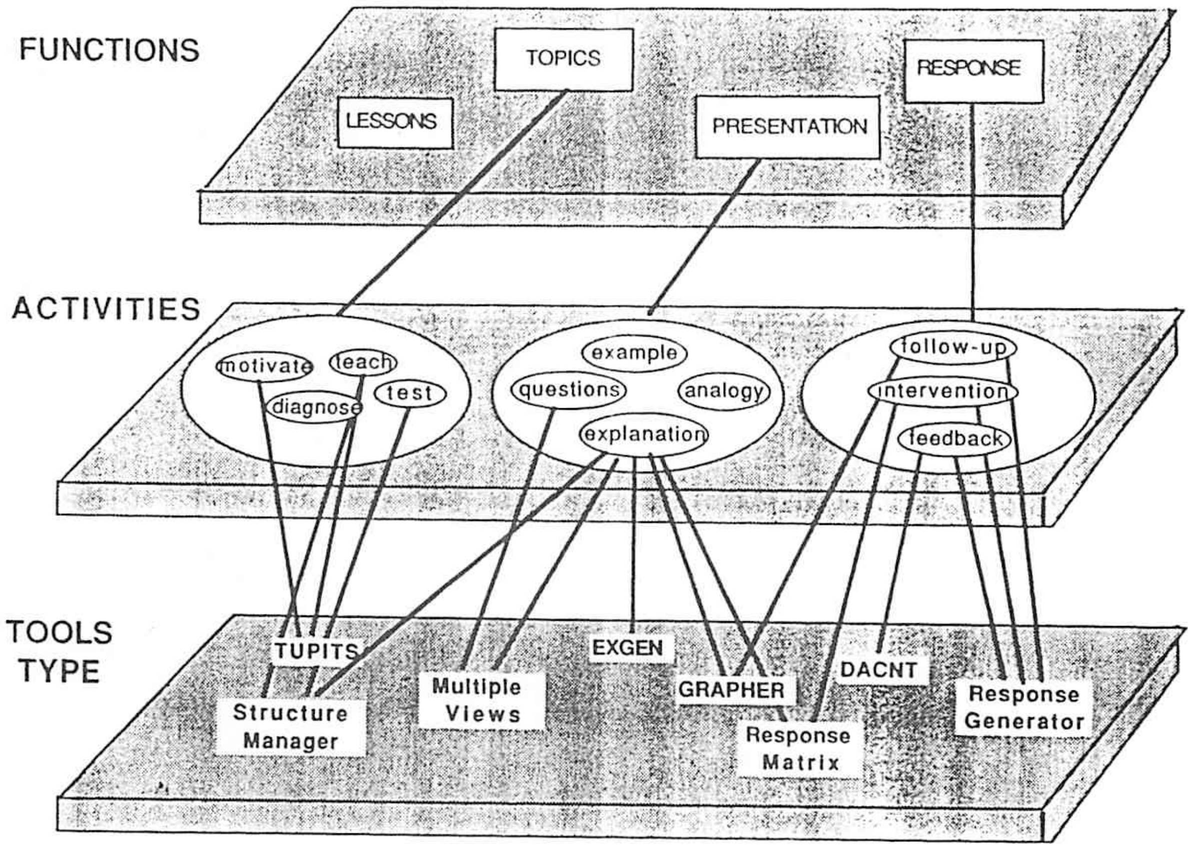


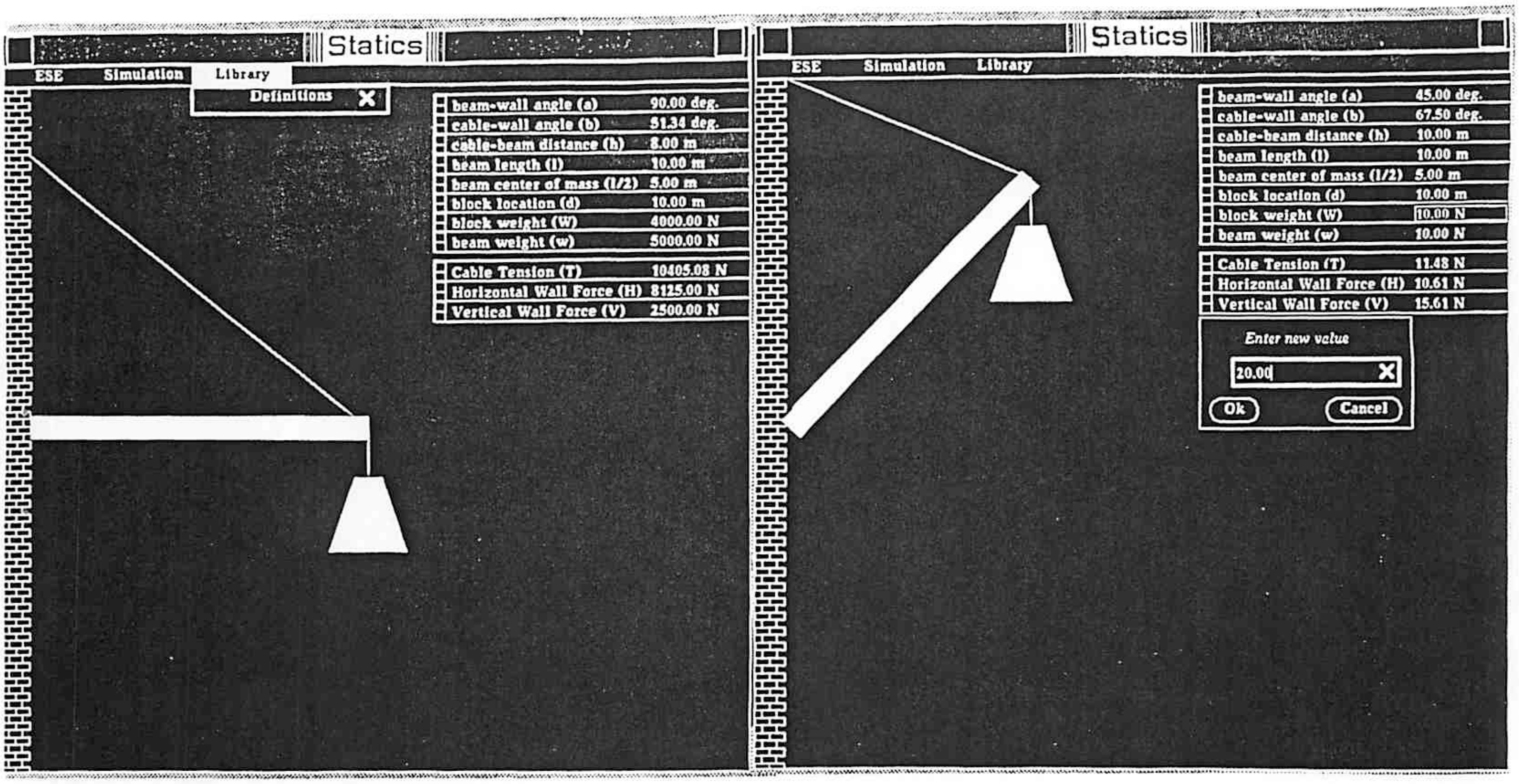
Figure 2: Tools for the representation and control of tutoring knowledge.

4.2 Tutoring Primitives

We define *tutoring primitives* as basic elements needed for communicating knowledge, such as topics to be taught, specific tutoring responses, and possible student errors. Our knowledge bases hold a variety of examples, knowledge types, tasks to be given to the student, and discourse states describing various human-machine interactions.

As an example of how tutoring primitives are used, we describe a tutor built in conjunction with the Exploring Systems Earth (ESE) Consortium [Duckworth, Kelley & Wilson, 1987]. This tutor is based on interactive simulations that encourage students to work with "elements" of physics, such as mass, acceleration, and force. The goal is to help students generate hypotheses as necessary precursors to expanding their own intuitions. We want the simulations to encourage students to "listen to" their own scientific intuition and to make their own model of the physical world before an encoded tutor advises them about the accuracy of their choices. This tutor has been described elsewhere [Woolf & Cunningham, 1987; Woolf & Murray, 1987] and will only be summarized here.

Figure 3 shows a simulation for teaching concepts in introductory statics. Students are asked to identify forces and torques on the crane boom, or horizontal bar, and to use rubber banding to draw appropriate force vectors directly on the screen. When the beam is in static



beam-wall angle (a)	90.00 deg.
cable-wall angle (b)	51.34 deg.
cable-beam distance (h)	8.00 m
beam length (l)	10.00 m
beam center of mass (l/2)	5.00 m
block location (d)	10.00 m
block weight (W)	4000.00 N
beam weight (w)	5000.00 N
Cable Tension (T)	10405.08 N
Horizontal Wall Force (H)	8125.00 N
Vertical Wall Force (V)	2500.00 N

beam-wall angle (a)	45.00 deg.
cable-wall angle (b)	67.50 deg.
cable-beam distance (h)	10.00 m
beam length (l)	10.00 m
beam center of mass (l/2)	5.00 m
block location (d)	10.00 m
block weight (W)	10.00 N
beam weight (w)	10.00 N
Cable Tension (T)	11.48 N
Horizontal Wall Force (H)	10.61 N
Vertical Wall Force (V)	15.61 N

Figure 3: Statics tutor.

equilibrium there will be no net force or torque on any part of it. Students are asked to solve both qualitative and quantitative word problems.

If a student were to specify incorrect forces either by omitting force lines or by including the wrong ones, the tutor makes a decision about how to respond. There are many possible responses depending on the tutorial strategy in effect. The tutor might present an explanation or hint, provide another problem, or demonstrate that the student's analysis leads to a logical contradiction. Still another response would be to withhold explicit feedback concerning the quality of the student's answer, and to instead demonstrate the consequence of omitting the "missing" force; i.e., the end of the beam next to the wall would crash down. Such a response would show the student how his/her conceptions might be in conflict with the observable world and to help him/her visualize both an internal conceptualization and the science theory.

Representing and Reasoning about Tutoring Primitives. The four knowledge bases described in Section 4.1 are used to represent topics, examples, explanations, and possible misconceptions. We use a network of Knowledge Units frames to explicitly express relationships between topics such as prerequisites, corequisites, and related misconceptions (Figure 4). An important notion about the network is that it is declarative --- it contains a structured space of concepts, but does not mandate any particular order for traversal of this space.

The network describes tutorial strategies in terms of a vocabulary of primitive discourse moves such as teach, motivate, contrast, and summarize. It is implemented in a language called TUPITS[2] which was built as a framework to facilitate development of numerous tutors. It is an object-oriented representation language that provides a framework for defining primitive components of a tutorial discourse interaction. These components are then used by the tutor to reason about its next action.

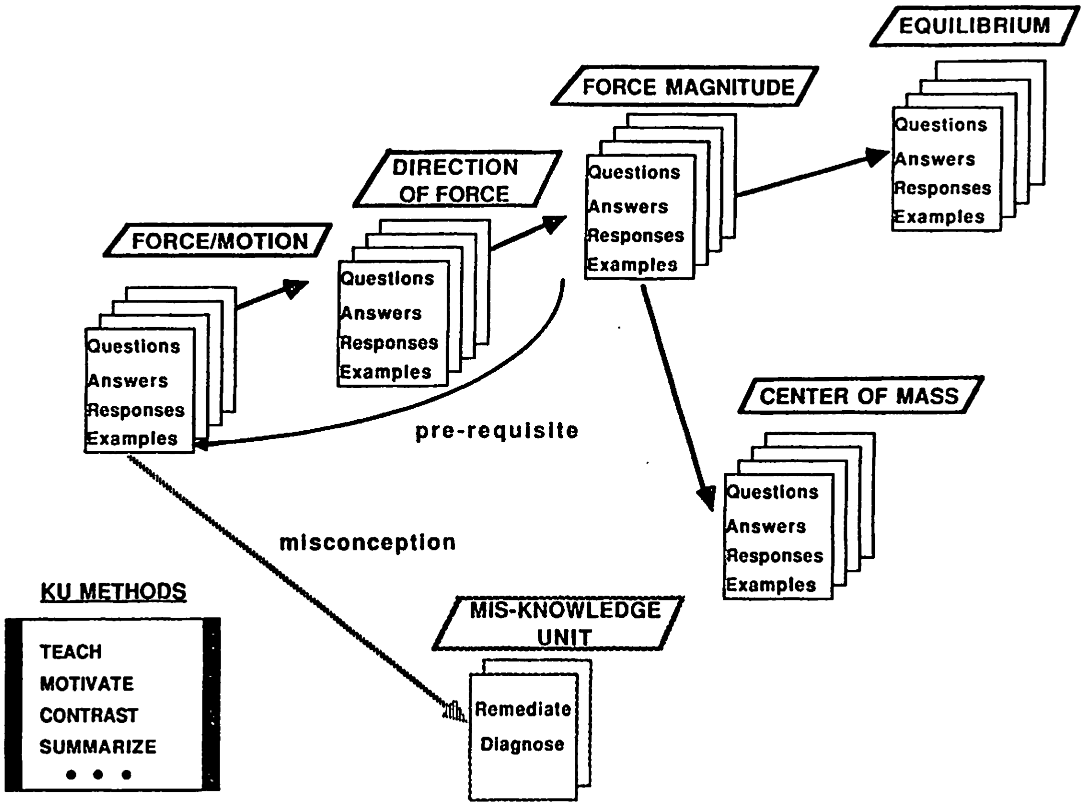


Figure 4: Hierarchy of frames.

As shown in Figure 4, each object in TUPITS is represented as a frame and each frame is linked with other frames representing prerequisites, corequisites, or triggered misconceptions. The primary objects in TUPITS are:

- Lessons which define high-level goals and constraints for each tutoring session;
- Knowledge Units (KUs);
- MIS-KUs, which represent common misconceptions, wrong facts or procedures, and other types of "buggy" knowledge;
- Examples, which specify parameters that configure an example, diagram, or simulation to be presented to the student;
- Questions, which define tasks for the student and how the student's behavior during the task might be evaluated; and
- Presentations, which bind an example together with associated questions.

[2] TUPITS (Tutorial discourse Primitives for knowledge-based Tutoring Systems) was developed by Tom Murray and runs on both Hewlett-Packard Bobcats and Apple Macintosh IIs.

MIS-KUs, or "Mis-Knowledge Units," represent common misconceptions or knowledge "bugs" and ways to remediate them. Remediation is inserted opportunistically into the discourse. The tutoring strategy parameterizes this aspect of Knowledge Unit selection by indicating whether such remediation should occur as soon as the misconception is suspected, or wait until the current Knowledge Unit has been completed.

Control is achieved through information associated with each object which allows the system to respond dynamically to new tutoring situations. For instance, Knowledge Units, or topics represented as objects, have procedural "methods" associated with them that:

- teach their own topic interactively;
- teach their own prerequisites;
- explain knowledge didactically;
- test students for knowledge of that topic;
- summarize themselves;
- provide examples of their knowledge (an instantiation of a procedure or concept);
- provide motivation for a student learning the topic; and
- compare this knowledge with that of other Knowledge Units.

A specific tutoring strategy manifests itself by parameterizing the algorithm used to traverse the knowledge primitives network based on classifications of and relations between knowledge units. Several major strategies have thus far been implemented. For example, the tutor might always teach prerequisites before teaching the goal topic. Alternatively, it might provide a diagnostic probe to see if the student knows a topic. Prerequisites might be presented if the student doesn't exhibit enough knowledge on the probe. These prerequisites may be reached in various ways, such as depth-first and breadth-first traversal. An intermediate strategy is to specialize the prerequisite relation into "hard" prerequisites, which are always covered before the goal topic, and "soft" prerequisites, taught only when the student displays a deficiency.

4.3 Discourse Analysis

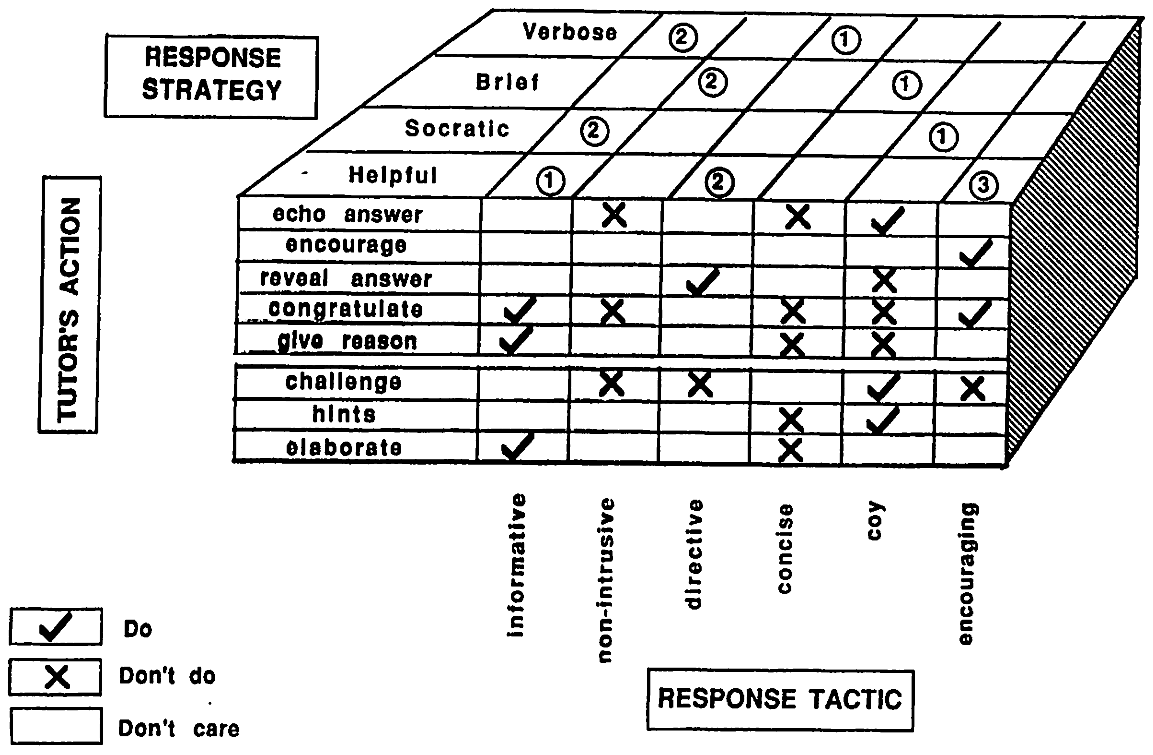
Our tutors are beginning to represent and reason about alternative responses to the student. Choices are concerned with how much information to give and what motivational comments to make. For instance, the machine must decide whether or not to:

- talk *about* the student's response;
- provide *motivational* feedback about the student's learning process;
- say *whether* an approach is appropriate, *what* a correct response would be, and *why* the student's response is correct or incorrect;

• provide hints, leading questions, or counter-suggestions.

Motivational feedback may include asking questions about the student's interest in continuing or providing encouragement, congratulations, challenges, and other statements with affective or prelocutionary content. Control is modulated by which tutoring strategy is in effect, which in turn places constraints on what feedback or follow-up response to generate. The strategy may also specify that system action be predicated on whether the student's response was correct, or whether any response was given.

Reasoning about Discourse Level. As a start to this process we have defined the Response Matrix, which now encodes several high-level response strategies and tactics (see Figure 5). For example, we have designated an informative response tactic as one in which the machine will elaborate, give reasons, and congratulate the student. For each concept represented in the machine, some of these primitive responses are available and the machine will generate the requested tactic. However, we also advise the system about strategies such as Socratic tutoring, being brief, and being verbose. Here we indicate a priority ordering; thus to be Socratic, the machine must place highest priority on the tactic called coy and secondary rating on the tactic to be informative. If there is a conflict between the checks and the crosses in the model model shown in Figure 5, that notation with the highest priority will win.



RESPONSE STRATEGY	informative	non-intrusive	directive	concise	coy	encouraging
Verbose	②		①			
Brief		②		①		
Socratic	②				①	
Helpful	①		②			③

TUTOR'S ACTION	informative	non-intrusive	directive	concise	coy	encouraging
echo answer		✗		✗	✓	
encourage						✓
reveal answer			✓		✗	
congratulate	✓	✗		✗	✗	✓
give reason	✓			✗	✗	
challenge		✗	✗		✓	✗
hints				✗	✓	
elaborate	✓			✗		

✓	Do
✗	Don't do
	Don't care

Figure 5: The Response Matrix: Reasoning about discourse

4.4 Managing Discourse

We realize that a more flexible and responsive discourse management technique is critical to a tutoring or consultant system. By discourse management, we mean the system's ability to maintain interactive discourse with a user and to custom-tailor its responses beyond the generalized discourse levels suggested above. Ideally, the system should tailor its response to the idiosyncrasies of a particular user. Machine discourse and response need not be in natural language to be effective [Servan-Schreiber, 1983].

For example, the system should ensure that an intervention relates directly to an individual's personal history, learning style, and on-line experience with the system. It should dynamically reason about a user's actions, the curriculum, and the discourse history. In doing this the tutor should make each user feel that his/her unique situation has been responded to appropriately and sensitively. In this way the system simulates one-on-one human tutoring behavior. The mechanism we use to do this is called a DACTN, *Discourse ACtion Transition Network*,[3] which represents and controls human-machine dialog. Figure 6 shows a DACTN for responding to a user about a question he/she asked.

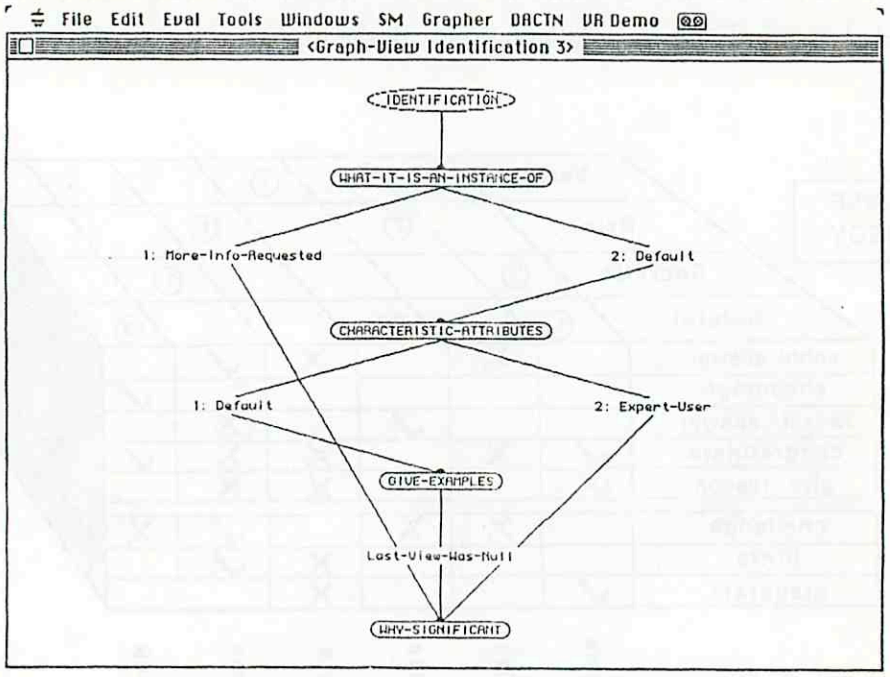

File Edit Eval Tools Windows SM Grapher DACTN VR Demo
<Graph-View Identification 3>
IDENTIFICATION
WHAT-IT-IS-AN-INSTANCE-OF
1: More-Info-Requested
2: Default
CHARACTERISTIC-ATTRIBUTES
1: Default
2: Expert-User
GIVE-EXAMPLES
Last-View-Was-Null
WHY-SIGNIFICANT

Figure 6: Discourse ACtion Transition Network: DACTN.

Sometimes the intervention steps designated by a DACTN are based on a taxonomy of frequently observed discourse sequences which provide default responses for the tutor [Woolf et al., 1987]. The discourse manager reasons about local context when making discourse *decisions*. Here local context is an aggregate of the user profile and response history.

[3] Rhymes with ACT-IN.

The DACTN represents the space of possible discourse situations: Arcs track the state of the conversation and are defined as predicate sets while nodes provide actions for the tutor. The discourse manager first accesses the situation indicated by the arcs, resolving any conflicts between multiply-satisfied predicate sets, and then initiates the action indicated by the node at the termination of the satisfied arc.

Arcs represent discourse situations defined by sets of predicates over the user profile and the state of the system. For instance, the value of the a particular arc in Figure 6 is determined by inferring over the current state of the profile and recent user responses. Placing actions at the nodes rather than on the arcs, as was done in the ATN [Woods, 1970], allows nodes to represent abstract actions which can be expanded into concrete substeps when and if the node is reached during execution of the DACTN.

Each user response causes the user model, or in this case the personality profile, to be updated, which in turn affects the interpretation and resolutions of subsequent interactions. DACTNs allow discourse control decisions to be based on a dynamic interpretation of the situation. In this way the mechanism remains flexible, domain-independent, and able to be dynamically rebuilt---decision points and machine actions are modifiable through a graphics editor. DACTNs have been implemented in three domains, the first is statics, described in Section 4.2, a second supports development of time management skills [Slovin & Woolf, 1988], and a third explains concepts and processes in elementary electrical network theory and is briefly described in the next section.

4.5 A Computational Model of Explanation

We have focused on developing computational theories of explanation and their application to computer-based "knowledge communication systems" [Wenger, 1988], e.g., advisory and tutoring systems which serve as a medium for the communication of knowledge. In this context, *explanation* is "the act or process of making plain or comprehensible; elucidation; clarification" (American Heritage Dictionary). Explanation research accounts for the content and structure of explicit communications, directed towards specific informative goals.

We claim that the most pressing research issues in explanation occur at an epistemological level of analysis. Such an analysis examines how the selection, organization, and presentation of the content of explanations are guided by three *epistemological sources of constraints,* namely:

- the internal structure of a domain's shared body of knowledge,

- the role of an individual's knowledge in understanding new concepts and situations, and
- the ways in which individuals are willing or able to transform their knowledge.

People won't understand an explanation, let alone retain, retrieve, and use it, if it does not make some contact with what they already know. Furthermore, there are restricted ways in which people are willing or able to transform the knowledge structures by which they understand the world. This means an explainer should identify pedagogically exploitable relations between the relevant knowledge and whatever guesses are available about the hearer's knowledge state, and structure the explanation to follow epistemological "gradients" along which he or she is likely to comprehend and integrate the new knowledge it contains. Thus, a theory of how to reason about the characteristics of knowledge when selecting and organizing the content of an explanation is necessary if we are to construct knowledge communication systems based on multi-use declarative knowledge bases.

The first application domain for our analysis of explanation is elementary electrical network theory. We emphasize the communication of an understanding of concepts such as "electricity," "current," and "resistance" within the context of reasoning about the operation of simple circuits and their components. We are building an explanation generation system to produce complex explanations and have described the system and the issues encountered elsewhere [Suthers, 1990; Suthers & Woolf, 1990].

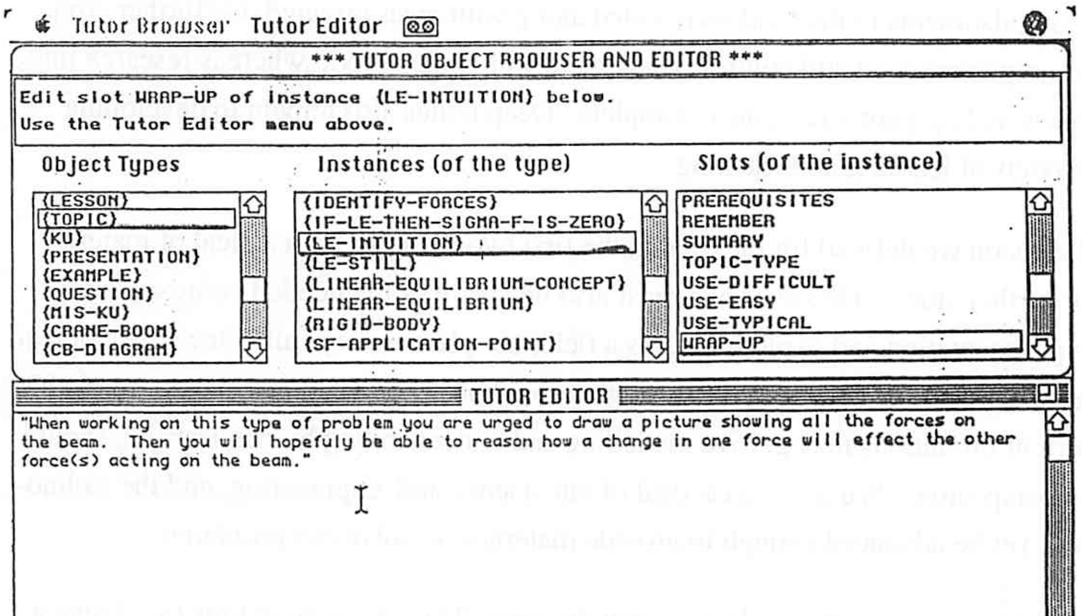
4.6 Knowledge Acquisition: Involving Teachers in the Process

As research into knowledge-based tutoring continues to produce increasingly more sophisticated systems and diverse representational paradigms, the gap between the computer science community and the instructional design community has widened. Educators' understanding, acceptance, and use of this research has been much slower than expected. Yet, as discussed in the first section of this article, educators are vital participants in the research collaborative to build knowledge-based tutors. Otherwise this research becomes increasingly academic and unconnected to the pragmatic aspects of teaching and learning. Clancey [1988] says: "...the reality today is that the endeavor is one that only experienced programmers (or experts trained to be programmers) can accomplish. Indeed, research of the past decade has only further increased our standards of knowledge representation desirable to teaching, while the tools for constructing such programs lag far behind or are not generally available."

There are several concerns about how and when to involve instructional designers in designing and testing tutoring strategies. Practicing teachers do not have well-articulated theories of learning or instruction. In addition, relevant instructional and cognitive theories are not operationalized to a level easily implemented in a computer, nor do these theories anticipate practical factors and domain related idiosyncrasies.

As a small step towards including teachers and educational researchers in this work, we have developed a knowledge acquisition interface for the statics tutor described in Section 4.2. The tutor conveys a qualitative understanding of Newton's laws and an intuitive grasp of the relationship between forces and force components in static systems [Woolf & Murray, 1987, Woolf et al., 1988]. Part of its curriculum centers around a learning environment called the "crane boom" in which the student manipulates a simulated physical system and observes the resulting forces and force components. The knowledge acquisition interface was developed in part as a response to the combined need for more teacher participation and more collaboration in developing tutoring strategies [Murray & Woolf, 1990]. The framework incorporates instructional design paradigms and facilitates rapid creation and manipulation of multiple tutoring strategies. Currently, teachers use this framework to incrementally add to and modify existing knowledge and to debug, update, and expand knowledge "pieces" for topics, examples, hints, prerequisites, etc. We also expect teachers to be able to create or modify tutoring strategies which determine how the tutor responds to the student.

The knowledge acquisition interface provides a view of the sum of the object types, attributes, relationships, and decision levels available to the tutor. This is a conceptual vocabulary for describing "what to teach" and "how to teach" it. The interface is designed to reify this framework for the teachers and others working with the system making it possible



```
*** TUTOR OBJECT BROWSER AND EDITOR ***

Edit slot WRAP-UP of instance {LE-INTUITION} below.
Use the Tutor Editor menu above.

Object Types         Instances (of the type)              Slots (of the instance)

{LESSON}             {IDENTIFY-FORCES}                    PREREQUISITES
{TOPIC}              {IF-LE-THEN-SIGMA-F-IS-ZERO}         REMEMBER
{KU}                 {LE-INTUITION}                       SUMMARY
{PRESENTATION}       {LE-STILL}                           TOPIC-TYPE
{EXAMPLE}            {LINEAR-EQUILIBRIUM-CONCEPT}         USE-DIFFICULT
{QUESTION}           {LINEAR-EQUILIBRIUM}                 USE-EASY
{MIS-KU}             {RIGID-BODY}                         USE-TYPICAL
{CRANE-BOOM}         {SF-APPLICATION-POINT}               WRAP-UP
{CB-DIAGRAM}

TUTOR EDITOR
"When working on this type of problem you are urged to draw a picture showing all the forces on
the beam.  Then you will hopefuly be able to reason how a change in one force will effect the other
force(s) acting on the beam."

USER| Use the Tutor Editor Menu
```

Figure 7: The Knowledge Acquisition Interface

for them to organize and encode their knowledge. Figure 7 shows a screen dump of the interface. The three menu bars at the top allow a teacher to select an object, instances of that object, and slots within the instance. The teacher might test, review, edit, or play any `object' in the system including any topic, summary, example, sound or graphic. Buttoning on one of these objects causes an editor to open along the bottom of the screen, which can be used by the teacher to change any text issued by the system. All changes are automatically entered into the Lisp code of the system. The teacher can then run a portion of the tutor to test his/her correction. In the figure, the wrap-up statement of the topic `intuition about linear-equilibrium' is being modified.

The interface facilitates both knowledge acquisition (in which a computer scientist interacts with the teacher to formalize his/her knowledge) and data entry and testing (in which the teacher adds to or modifies the knowledge base). This interface is further described in [Murray & Woolf, 1990].

5 Conclusion and Discussion

This article has documented some issues to explore enroute to creating an integration of instructional and computer science in service to developing knowledge-based tutoring systems. Several accomplishments in the field were noted along with areas targeted for further work. For instance, representation and control issues seem well understood, whereas research into student models and diagnosis remains incomplete. Deep issues also remain in developing cognitive models of learning and teaching.

Given the vision we defined for the field in the first two sections, a great deal of material remains for further study. However, we must also be aware of the needed context -- i.e., repeated experimentation and exploration. As a field, our plate is very full. One small defeat, whether in a classroom or with regard to a single box, should not be considered a remarkable failure. Current limitations in cognitive modelling and understanding human learning should beplaced in perspective. We ask a great deal of our science and engineering, and the technology may not yet be advanced enough to provide materials to solve our problems.

Winning or loosing in a single subarea is not the point. Thus if a system fails to achieve a powerful breakthrough we need not abandon the search. Rather, we should realize that a research effort might need to be redirected until such time as learning and teaching are more completely understood, or until technology permits a solution to problems currently beyond our ability. A more principled view of the field requires development of several broad-based research approaches in which success and failure are simply a part of the game. Coubertin, the founder of modern-day Olympics (of which Calgary was a recent host, as well as the host of this NATO conference) expressed it this way: the goal is "not to win, but to take part, just as the most important thing in life is not the triumph, but the struggle."

References

1. Anderson, J.R.: Analysis of student performance with the Lisp tutor. In: *Diagnostic Monitoring of Skill and Knowledge Acquisition,* (N. Frederickson et al., eds.), Lawrence Erlbaum Associates, Hillsdale, NJ., 1990.
2. Anderson, J. and B. Reiser: The Lisp tutor, *Byte* 10, 159-175, 1986.
3. Anderson, J.R., Boyle, C.F., Corbett, A.T. & Lewis, M.W.: Cognitive modeling and knowledge-based tutoring. *Artificial Intelligence,* Vol. 42, 7-50, 1990.
4. Bloom, B.S.: The 2-Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. Educational Researcher, Vol. 13., No. 6, pp. 4-16, 1984.
5. Bowscher, J.E.: *Educating America: Lessons Learned in the Nation's Corporations,* Wiley & Sons, New York, 1989.
6. Clancey, W.J.: A practical authority shell for apprenticeship learning. Proceedings of the International Conference on Intelligent Tutoring Systems (ITS-88). University of Montreal, Canada, June, 1988.
7. Corbett, A. and Anderson, J.: The effect of feedback control on learning to program with the Lisp tutor. Proceedings of the 12th Annual Meeting of the Cognitive Science Society, pp. 702-708. Cambridge, MA, 1990.
8. Dillenbourg, P.: The language shift: A mechanism for triggering metacognitive activities. (This volume.) In : *Foundations and Frontiers in Instructional Computing Systems* (P. Winne & M. Jones, eds.), Springer-Verlag, 1991(?).
9. Duckworth, E., Kelley, J., & Wilson, S.: AI goes to school. *Academic Computing,* November, 1987, pp. 6-10, 38-43, 62-63.
10. Goldman, H.D.: Instructional Systems Development in the United States Air Force. In: *Instructional Development: The State of the Art II,* (Bass, R. And Dills, C., eds.), Dubuque: Kendall/Hunt, 1984.
11. Johnson, L. and E.M. Soloway: Intention-based diagnosis of programming errors. Proceedings of the National Conference of Artificial Intelligence, pp. 369-380. Austin, Texas, 1984.
12. Lesgold, A., Laijoie, S.P., Bunzo, M., and Eggan, G.: A coached practice environment for an electronics troubleshooting job. In: *Computer Assisted Instruction and Intelligent Tutoring Systems* (J. Larkin, R. Chabay and C. Shefic, eds.), Lawrence Erlbaum Associates, Hillsdale, NJ, 1990
13. Lewis, M., McArthur, D., Stasz, C., & Zmuidzinas, M.: Discovery-based tutoring in mathematics. In: Working Notes Artificial Intelligence Spring Symposium, AAAI, Palo Alto, CA, 1990.

14. Murray, T. and Woolf, B.: A knowledge acquisition framework facilitating multiple tutoring strategies, Working Notes: Symposium on Knowledge-based Environments for Learning and Teaching, AAAI: Menlo Park, CA, 1990, pp. 120-127.

15. Schooler, L. and Anderson, J.: The disruptive potential of immediate feedback. Proceedings of the 12th Annual Meeting of the Cognitive Science Society, pp. 796-803. Cambridge, MA, 1990.

16. Shute., V.: Rose garden promises of intelligent tutoring systems: Blossom or thorn? Paper presented at the Space Operations, Applications and Research (SOAR) Symposium, 1990. Contact V. Shute, AFHRL, Brooks Air Force Base, TX 78235-5601.

17. Shute, V.J., Glaser, R., and Raghavan, K.: Inference and discovery in an exploratory laboratory. In : *Learning and Individual Differences* (P.L. Ackerman, R.J., Sternberg, and R. Glaser, eds.), W.H. Freeman, New York, pp. 279-326, 1989.

18. Servan-Schreiber, D.: From intelligent tutoring to computerized psychotherapy. Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87). Los Altos, CA: Morgan Kaufmann. pp. 66-71, 1987.

19. Slovin, T., & Woolf, B. P.: A consultant tutor for personal development. In: Proceedings of the International Conference on Intelligent Tutoring Systems (C. Frasson, ed.), Dept. of Information, University of Montreal, Canada, pp. 162-169, 1988.

20. Singley, K.: The reification of goal structures in a calculus tutor: Effects on problem-solving performance. *Interactive Learning Environments,* Vol. 1 (2), 102-123, 1990.

21. Stevens, A., & Collins, A.: The goal structure of a Socratic tutor. Proceedings of the Association for Computing Machinery Annual Conference, 1977. Also available as BBN Report No. 3518 from Bolt Beranek and Newman Inc., Cambridge, Mass., 02138.

22. Suthers, D.: The epistemological structure of explanation. In: Proceedings of AAAI 1990 Workshop on Explanation. Boston, July 1990.

23. Suthers, D. and Woolf, B.: Accounting for the epistemological structure of explanation. In: Working Notes of the AAAI 1990 Spring Symposium on Knowledge-based Environments for Learning and Teaching. AAAI, Menlo Park, CA, 1990, pp. 46-51. Also available as Technical Report 90-36, Computer and Information Science Dept., University of Massachusetts, Amherst.

24. Wenger, E.: *Artificial Intelligence and Tutoring Systems.* Los Altos, CA: Morgan Kaufmann Publishers, 1988.

25. Woods, W.: Transition network grammars for natural language analysis. *Communications of the ACM,* 13(10), 591-606, 1970.

26. Woolf, B.: Intelligent tutoring systems: A survey. In: *Exploring Artificial Intelligence* (H. Shrobe and the American Association for Artificial Intelligence, eds.), Morgan Kaufmann Publishers, Palo Alto, CA, pp. 1-43, 1988.

27. Woolf, B., and Cunningham, P.: Multiple knowledge sources in intelligent tutoring systems. *IEEE Expert,* Los Alamitos, CA, Summer, pp 41-54, 1987.

28. Woolf, B., and Murray, T.: A framework for representing tutorial discourse, International Joint Conference in Artificial Intelligence (IJCAI-87), Morgan Kaufmann, Inc., Palo Alto, CA, pp. 189-192, 1987.

29. Woolf, B., Murray, T., Suthers, D. and Schultz, K.: Knowledge primitives for tutoring systems. Proceedings of the International Conference on Intelligent Tutoring Systems (ITS88), University of Montreal, Canada, pp. 491-498, 1988.

30. Woolf, B., Soloway, E., Clancey, W.J., VanLehn, K. and Suthers, D.: Knowledge-based environments for teaching and learning, *AI Magazine,* Vol. 11 (5), 74-77, 1991.