Heffernan, Koedinger & *Razzaq* (2008) Expanding the model-tracing architecture: A 3rd generation intelligent tutor for Algebra symbolization. *The International Journal of Artificial Intelligence in Education. 18(2).* 153-178.

Expanding the Model-Tracing Architecture: A 3rd Generation Intelligent Tutor for Algebra Symbolization.

Neil T. Heffernan
nth@wpi.edu
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609

Kenneth R. Koedinger
koedinger@cmu.edu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Leena Razzaq
leenar@wpi.edu
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609

Abstract. Model-Tracing Tutors (MTTs) are intelligent tutoring systems that have been very successful at aiding student learning, but have not reached the level of performance of experienced human tutors. To that end, this paper presents a new architecture called ATM (for "Adding a Tutorial Model") which is an extension to the model-tracing architecture that allows these tutors to engage in a dialog that is more like what experienced human tutors do. Specifically, while MTTs provide *hints* toward *doing* the next problem-solving step, the ATM architecture adds the capability to *ask* questions towards *thinking* about the knowledge behind the next problem-solving step. We present a new tutor built in ATM, called *Ms. Lindquist*, which is designed to carry on a tutorial dialog about algebra symbolization. The difference between ATM and MTT is the separate *tutorial model* that encodes *pedagogical content knowledge* in the form of different tutorial strategies, which were partially developed by observing an experienced human tutor. Ms. Lindquist has tutored thousands of students at www.AlgerbaTutor.org. Future work will reveal if Ms. Lindquist is a better tutor because of the addition of her tutorial model.

Keywords. Intelligent tutoring systems, teaching strategies, model-tracing, student learning, algebra.

INTRODUCTION

This paper describes a step toward the next generation of practical intelligent tutoring systems. Let us say that CAI (Computer Aided Instruction) systems were 1st generation tutors (see Kulik, Bangert & Williams, 1983). They presented a page of text or graphics and, depending upon the student's answer, put up a different page. The 2nd generation of tutors was Model-Tracing Tutors (MTTs) (Anderson & Pelletier, 1991) that allow the tutor to follow the problem-solving steps of the student through the use of a detailed cognitive model of the domain. MTTs have had considerable success (Koedinger, Anderson, Hadley & Mark, 1997; Anderson, Corbett, Koedinger & Pelletier, 1995; Shelby et al., 2001) in improving student learning. MTTs have also had commercial success with more than 1% of American high schools now using MTTs sold by Carnegie Learning Incorporated (www.CarnegieLearning.com).

Despite the success of MTTs, they have not reached the level of performance of experienced human tutors (Anderson et al., 1995; Bloom, 1984) and instruct in ways that are quite different from human tutors (Moore, 1996). Various researchers have criticized model-tracing (Ohlsson, 1986; McArthur, Stasz, & Zmuidzinas, 1990). For instance, McArthur et al. (1990) criticized Anderson's et al. (1985) model-tracing ITS and model-tracing in general "because each incorrect rule is paired with a particular tutorial action (typically a stored message)…Anderson's tutor is tactical, driven by local student errors (p. 200)." They go on to argue for the need for a strategic tutor. The mission of the Center for Interdisciplinary Research on Constructive Learning Environments (CIRCLE) is 1) to study human tutoring and 2) to build and test a new generation of tutoring systems that encourage students to construct the target knowledge instead of telling it to them (VanLehn et al., 1998). The yet untested hypothesis that underlies this research area is that we can improve computer tutors (i.e., improve the learning of students who use them) by making them more like experienced human tutors.[1] A more specific assumption of this work is that students will learn better if they are engaged in a *dialog* to help them construct knowledge for themselves, rather than just being *hinted* toward inducing the knowledge from problem-solving experiences.

This paper is also focused on a particular aspect of tutoring. In particular, it is focused on what we call the *knowledge-search loop*. We view a tutoring session as containing several loops. The outermost loop is the *curriculum loop*, which involves determining the next best problem to work on. Inside of this loop, there is the *problem-solving loop*, which involves helping the student select actions in the problem solving process (e.g., the next equation to write down, or the next element to add to a free-body diagram in a physics problem). Traditional model-tracing is focused at this level, and is effective because it can follow the individual path of a student's problem solving through a complicated problem solving process. However, if the student is stuck, it can only provide hints or rhetorical questions toward what the student should do next. Model-tracing tutors do not ask new questions that might help students towards identifying or constructing relevant knowledge. In contrast, a human tutor might "dive down" into what we call the *knowledge-search loop.* Aiding students in knowledge search involves asking the student questions whose answers are not necessarily part of the problem solving process, but are chosen to assist the student in learning the knowledge needed at the problem solving level. It is this innermost *knowledge-search* loop that this paper is focused upon because is it has been shown that most learning happens only when students reach an impasse (VanLehn, Siler, Murray,

[1] Ideally, the best tutors should be chosen to model, but it is difficult to determine which are the best. This particular study is limited in that it is based upon a single experienced tutor.

Yamauchi & Baggett, 2003). In addition, VanLehn et al. suggested that different types of tutorial strategies were needed for different types of impasses.

The power of the model-tracing architecture has been in its simplicity. It has been possible to build practical systems with this architecture, while capturing some, but not all, features of effective one-on-one tutoring. This paper presents a new architecture for building such systems called ATM (for **A**dding a **T**utorial **M**odel) (Heffernan, 2001). ATM is intended to go a step further but maintain simplicity so that practical systems can be built. ATM incorporates more features of effective tutoring than model-tracing tutors, but does not aspire to incorporate all such features.

A number of 3rd generation systems have been developed (Core, Moore & Zinn, 2000; VanLehn et al., 2000; Graesser et al., 1999; Aleven & Koedinger, 2000a). In order to concretely illustrate the ATM architecture, this paper also presents an example of a tutor built within this architecture, called *Ms. Lindquist*. Ms. Lindquist is not only able to model-trace student actions, but can be more human-like in carrying on a running conversation with the student, complete with probing questions, positive and negative feedback, follow-up questions in embedded sub-dialogs, and requests for explanations as to why something is correct. In order to build Ms. Lindquist we have expanded the model-tracing paradigm so that Ms. Lindquist not only has a model of the student, but also has a model of tutorial reasoning. Building a tutorial model is not a new idea, (e.g., Clancey, 1982), but incorporating it into the model-tracing architecture is new. Traditional model-tracing tutors have an implicit model of the tutor; that model is that tutors keep students on track by giving (sometimes implicitly) positive feedback as well as making comments on student's wrong actions. Traditional model-tracing tutors do not allow tutors to ask new question to break steps down, nor do they allow multi-step lines of questioning. Based on observation of both an experienced tutor and cognitive research (Heffernan & Koedinger, 1997,1998), this tutorial model has multiple tutorial strategies at its disposal.

MTTs are successful because they include a detailed model of how students solve problems. The ATM architecture expands the MTT architecture by also including a model of what experienced human tutors do when tutoring. Specifically, similar to the model of the student, we include a tutorial model that captures the knowledge that a tutors needs to be a good tutor for the particular domain. For instance, some errors indicate minor slips while others will indicate major conceptual errors. In the first case the tutor will just respond with a simple corrective getting the student back on track (which is what model-tracing tutors do well), but in the second case, a good tutor will tend to respond with a more extended dialog (something that is impossible in the traditional model-tracing architecture).

We believe a good human tutor needs at least three types of knowledge. First, they need to know the domain that they are tutoring, which is what traditional MTTs emphasize by being built around a model of the domain. Secondly, they need general pedagogical knowledge about how to tutor. Thirdly, good tutors need what Shulman (1986) calls *pedagogical content knowledge*, which is the knowledge at the intersection of domain knowledge and general pedagogical knowledge. A tutor's "pedagogical content knowledge" is the knowledge that he or she has about how to teach a specific skill or content domain, like algebra. A good tutor is not simply one who knows the domain, nor is a good tutor simply one who knows general tutoring rules. A good tutor is one who also has content specific strategies (an example will be given later in the section "The Behavior of an Experienced Human Tutor") that can help a student overcome common difficulties. McArthur et al.'s (1990) detailed analysis of human tutoring concurred:

> *Perhaps the most important conclusion we can draw from our analysis is that the reasoning involved in tutoring is subtle and sophisticated. … First, … competent tutors possess extensive knowledge bases of techniques for defining and introducing tasks and remediating misconceptions. … [and] perhaps the most*

important dimension of expertise we have observed in tutoring involves planning. Not only do tutors appear to formulate and execute microplans, but also their execution of a given plan may be modified and pieces deleted or added, depending on changing events and conditions.

McArthur et al. recognized the need to model the strategies used by experienced human tutors, and that such a model could be a component of an intelligent tutoring system.

Building a traditional model-tracing tutor is not easy, and unfortunately, the ATM architecture involves only additional work. Authoring in Anderson & Pelletier's (1991) model-tracing architecture involves significant work. Programming is needed to implement a cognitive model of the domain, and ideally, this model involves psychological research to determine how students actually solve problems in that domain (e.g., Heffernan & Koedinger, 1997; Heffernan & Koedinger, 1998). The ATM architecture involves the additional work of first analyzing the tutorial strategies used by experienced human tutors and then implementing such strategies in a tutorial model. This step should be done before building a cognitive model, as it constrains the nature and level of detail in the cognitive model that is needed to support the tutorial model's selection of tutorial options.

In this paper, we first describe the model-tracing architecture used to build second-generation systems and then present an example of a tutor built in that architecture. Then we present an analysis of an experienced human tutor that serves as a basis for the design on Ms. Lindquist and the underlying ATM architecture. We illustrate the ATM architecture by describing how the Ms. Lindquist tutor was constructed within. The Ms. Lindquist tutor included both a model of the student (the research that went into the student model is described in Heffernan & Koedinger, 1997 & 1998) as well as a model of the tutor.

THE 2ND GENERATION ARCHITECHTURE: MODEL-TRACING

The Model-Tracing Architecture was invented by researchers at Carnegie Mellon University (Anderson & Pelletier, 1991; Anderson, Boyle & Reiser, 1985) and has been extensively used to build tutors, some of which are now sold by Carnegie Learning, Inc (Corbett, Koedinger, Hadley, 2001). These tutors have been used by thousands of schools across the country and have been proven to be very successful (Koedinger, Anderson, Hadley & Mark, 1995). Each tutor is constructed around a cognitive model of the problem solving knowledge students are acquiring. The model reflects the ACT-R theory of skill knowledge (Anderson, 1993) in assuming that problems solving skills can be modeled as a set of independent production rules. Production rules are if-then rules that represent different pieces of knowledge (A concrete example of a production will be given in the section on "Ms. Lindquist Cognitive Student Model".) Model-tracing provides a particular approach to implementing the standard components of an intelligent tutoring system, which typically include a graphical user-interface, expert model, student model and pedagogical model. Of these components, MTTs emphasize the first three.

Anderson, Corbett, Koedinger & Pelletier (1995) say that the first step in building a MTT is to define the interface in which the problem solving will occur. The interface is usually analogous to what the student would do on a piece of paper to solve the problem. The interface enables students to reify steps in their problem-solving performance, thus enabling the computer to be able to follow the problem-solving steps the student is using.

The main idea behind the model-tracing architecture, is that if you have a model of what the student might do (i.e., a cognitive model including different correct and incorrect steps that the student could take) then you will be able to offer appropriate feedback to students including positive feedback as well as hints to the student if they are in need of help. Each task that a student is presented with can be solved by applying different pieces of knowledge. Each piece of knowledge is represented by a production rule. The expert model contains the complete[2] set of productions needed to solve the problems, as well as the "buggy" productions. Each buggy production represents a commonly occurring incorrect step. The model-tracing algorithm uses the cognitive model to "model-trace" each step the student takes in a complex problem solving search space. This allows the system to provide feedback on each problem solving action as well as give hints if the student is stuck.

Specifically, when the student answers a question, the model-tracing algorithm is executed in an attempt to do a type of *plan recognition* (Kautz & Allen, 1986). For instance, if a student was supposed to simplify “7(2+2x) + 3x” and said “10+5x”, a model tracer might respond with a buggy message of “Looks like you failed to distribute the <u>7</u> to the <u>2x</u>”. (The underlined text would be filled in by a template so that message applies to all situations in which the student fails to distribute to the second term.) A model tracer is only able to do this if a bug rule had been written that is able to model that incorrect rule of forgetting to distribute to the second term. Note that model-tracing often involves firing rules that work correctly (like the rule that added the 2x +3x, as well as rules that do some things incorrectly).

More specifically, the model-tracing algorithm is given a set of production rules and an initial state, represented by what are called in the ACT-R community *working memory elements* but are referred to as *facts* in the AI community (e.g. JESS/CLIPS terminology). The algorithm does a search (sometimes this is implemented as an iterative deepening depth first search) to construct all possible responses that the model is able to produce and then tries to see if the student’s actual response matches any of model’s responses. There are two possible outcomes; either the search fails, indicating the student did something unexpected (which usually means they did something wrong), or the search succeeds (we say the input was "traced") and returns a list of productions that represent the thinking or planning the student went through. However, just because a search succeeds does not mean that the student's answer is correct. The student's input might have been traced using a buggy-production rule (possibly along with some correct rules) as the example above illustrated about failing to distribute to the second term.

One downside of the model-tracing approach is that because the model-tracing algorithm is doing an exponential search for each student’s action, model-tracing can be quite slow. A “pure” cognitive model will not make any reference to the student’s input and instead would be about to generate the student’s input itself. However, if the model is able to generate say a million different responses at a given point in time, the algorithm will take a long time to respond. Therefore, some modelers, we included, take the step of adding constraints to prevent the model from generating all possible actions, dependant upon the student’s input. Others have dealt with the speed problem differently by doing more computation ahead of time instead of in real time; Kurt Van Lehn’s approach seems to be to use rules to generate all the different possible actions and store those actions (in what he calls a *solution graph*), rather than use the rules at run time to generate all the actions.

An additional component of traditional model-tracing architecture is called *knowledge-tracing* which is a specific implementation of an "overlay"[3] student model. As students work

[2] The somewhat radical assumption of model-tracing tutors is that the set of productions needs to be **complete.** This requires the cognitive modeler to model all the different ways to solve a problem as well as all the different ways of producing the common errors. If the student does something that cannot be produced by the model, it is marked as wrong.

[3] An overlay student model is one in which the student's knowledge is treated as a subset of the knowledge

through a problem, the system keeps track of the probabilities that a student knows each production rule. These estimates are used to decide what is the next best problem to present to the student. The ATM architecture makes no change to knowledge tracing.

In summary, model-tracing tutors give three types of feedback to students: 1) *flag feedback*, 2) *buggy messages*, and 3) a *chain of hints*. Flag feedback simply indicates the correctness of the response, sometimes done by using a color (e.g., green=correct or red=wrong). A buggy message is a text message that is specific to the error the student made (examples below). If a student needs help, they can request a "Hint" to receive the first of a chain of hints that suggests things for the student to think about. If the student needs more help, they can continue to request a more specific hint until the "bottom-out" message is delivered that usually tells the student exactly what to type. Anderson & Pelletier (1991) argue for this type of architecture because they found

> “that the majority of the time students are able to correct their errors without further instructions. When students cannot, and request help, they are given the same kind of explanation that would accompany a training example. Specifically, we focus on telling them what to do in this situation rather than focus on telling them what was wrong with their original conception. Thus, in contrast to the traditional approach to tutoring we focus on re-instruction rather than bug-diagnosis.”

We agree that emphasizing bug-diagnosis is probably not particularly helpful, however simply "spewing" text at the student may not be the most pedagogically effective response. This point will be elaborated upon in the section describing Ms. Lindquist's architecture.

OTHER SYSTEMS

Murray (1999) reviewed the state of the art in authoring tools, and placed model-tracing tutors into a separate category (i.e., domain expert systems) as a different type of intelligent tutoring system. There has not been much work in bridging modeling tracing tutors with other types of systems. Many other systems have attempted to model the tutor but have not incorporated model-tracing of the student. This paper can be viewed as an initial attempt to do this coming from the model-tracing perspective.

The ATM architecture is our attempt to build a new architecture, from scratch, that will extend the model-tracing architecture to allow for better dialog capabilities. Other researchers (Aleven & Koedinger, 2000a; Core, Moore & Zinn, 2000; Freedman & Evens, 2000; Graesser et al., 1999; VanLehn et al., 2000) have built 3rd generation systems but ATM is the first to take the approach of generalizing the successful model-tracing architecture to seamlessly integrate tutorial dialog. Besides drawing on the demonstrated strengths of model-tracing tutors, this approach allows us to show how model tracing is a simple instance of tutorial dialog. Aleven and Koedinger (2000a & 2000b) have built a geometry tutor in the traditional model-tracing framework but have added a requirement for students to explain some of their problem-solving steps. The system does natural language understanding of these explanations by parsing a student's answer. The system's goal is to use traditional buggy feedback to help students refine their explanations. Many of the hints and buggy messages ask new "questions", but they are only rhetorical. For instance, when the student justifies a step by saying "The angles in an isosceles triangle are equal" and the tutor responds with "Are all angles in a isosceles triangle equal?" the student doesn't get to say "No, it’s just the base angles". Instead, the student is expected to modify

---

of the expert.

the complete explanation to say "The *base* angles in an isosceles triangle are equal." Therefore, the system's strength appears to be its natural language understanding, while its weakness is in not having a rich dialog model that can break down the knowledge construction process through new non-rhetorical questions and multi-step plans.

Another tutoring system that does natural language understanding is Graesser's et al. (1999) system called "AutoTutor". AutoTutor is a system that has a "talking head" that is connected to a text-to-speech system. AutoTutor asks students questions about computer hardware and the student types a sentence in reply. AutoTutor uses latent semantic analysis to determine if a student's utterance is correct. That makes for a much different sort of student modeling than model-tracing tutors. The most impressive aspect of AutoTutor is its natural language understanding components. The AutoTutor developers (Graesser et al.,1999) de-emphasized dialog planning based on the claim that novice human tutors do not use sophisticated strategies, but nevertheless, can be effective. Auto-tutor does have multiple tutorial strategies (i.e., "Ask a fill-in-the-blank question" or "Give negative feedback."), but these strategies are not multi-step plans. However, work is being done on a new "Dialogue Advancer Network" to increase the sophistication of its dialog planning.

The demonstrations systems built by Rickel, Ganeshan, Lesh, Rich & Sidner, (2000) are interesting due to the incorporation of an explicit theory of dialog structure by Grosz & Sidner (1986). However, both their pedagogical content knowledge and their student modeling are weak.

Baker (1994) looked at modeling tutorial dialog with a focus on how students and tutors negotiate, however this paper ignores negotiations.

The CIRCSIM-Tutor project (see Cho, Michael, Rovick, and Evens, 2000; Freedman & Evens, 1996) has done a great deal of research in building dialog-based intelligent tutors systems. Their tutoring system, while not a model-tracing tutor, engages the student in multi-step dialogs based upon two experienced human tutors. In CIRCSIM-Tutor, the dialog planning was done within the APE framework (Freedman, 2000). Freedman's approach, while developed independently, is quite similar to our approach for the tutorial model in that it is a production system that is focused on having a hierarchal view of the dialog.

VanLehn et al. (2000) are building a 3rd generation tutor by improving a 2nd generation model-tracing tutor (i.e., the Andes physics tutor) by appending onto to it a system (called Altas) that conducts multiple different short dialogs. The new system, called Atlas-Andes, is similar to our approach in that students are asked new questions directed at getting the student to construct knowledge for themselves rather than being told. Also similar to our approach is that VanLehn and colleagues have been guided by collecting examples from human tutoring sessions. While their goal and methodology are similar, their architecture for 3rd generation tutors is different. VanLehn et al. (2000) says that "Atlas takes over when Andes would have given its final hint. (p. 480)" indicating that the Atlas-Andes system is two systems that are loosely coupled together. When students are working in Atlas, they are, in effect, using a 1st generation tutor that poses multiple-choice questions and branches to a new question based on the response, albeit one that does employ a parser to map the student's response to one of the multiple-choice responses. Because of this architectural separation, the individual responses of students are no longer being model-traced or knowledge-traced. This separation is in contrast with the goal of seamless integration of model-tracing and dialog in ATM.

Carnegie Learning's Cognitive Algebra Tutor

We will now give an example of the sort of feedback traditional model-tracing tutors provide. We will look at the Carnegie Learning Inc.'s tutor called the "Cognitive Algebra Tutor". This software teaches various skills in algebra (i.e., problem analysis, graphing and equation solving), but the skill we will focus on here is the symbolization process (i.e., where a student is asked to write an equation representing a problem situation). Symbolization is fundamental because if students cannot translate problems into the language of algebra, they will not be able to apply algebra to solve them. Symbolization is also a difficult task for students to master. The two most relevant windows related to symbolizations are shown in Figure 1 and Figure 2. Figure 1 is a statement of a word problem, which poses multiple questions to the student. The student is expected to answer these questions by completing a table shown (partially filled in) in Figure 2.

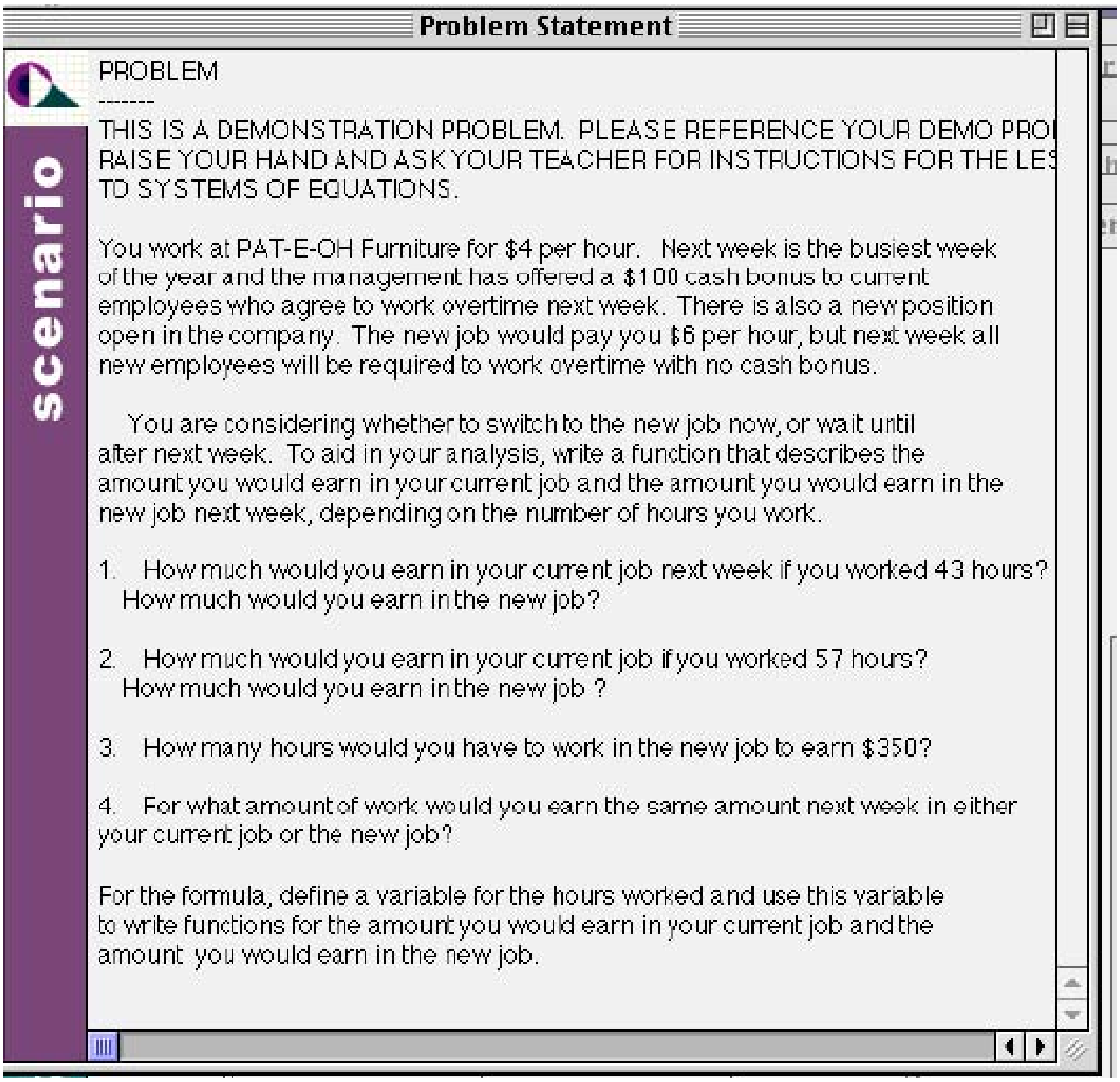

Figure 1: The Problem Statement window from the Carnegie Learning Inc's Cognitive Algebra Tutor.

In Figure 2 we see that the student has already identified names for three quantities (i.e., "hours worked", "The amount you would earn in your current job", and "the amount you would earn in the new job"), as well as having identified units (i.e., "hours", "dollars" and "dollars" respectively) as well as having chosen a variable (i.e., "h") to stand for the "hours worked" quantity. In the bottom four rows of the table, the student will answer the four concrete questions specified in the problems statement window (Figure 1).

One of the most difficult steps for students is generating the algebraic expression and Figure 2 shows a student who is currently in the middle of attempting to answer this sort of problem, as shown by the fact that that cell is highlighted. The student has typed in "100-4*h" but has not yet hit return. The correct answer is "100+4*h".

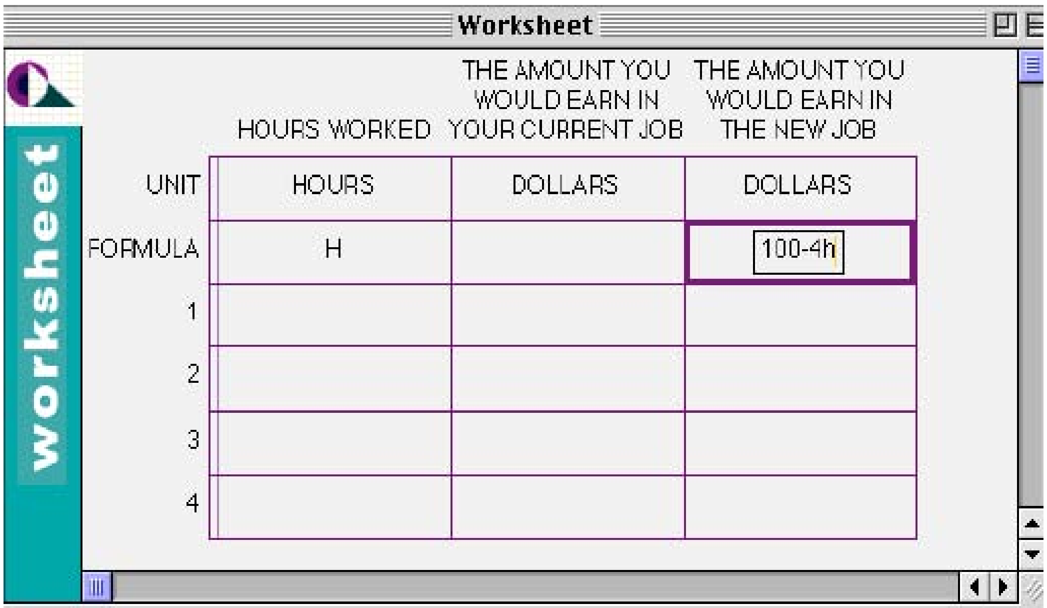

Figure 2: The worksheet window from the Carnegie Learning tutor. The student has already filled in the column headings as well as the units, and is working on the formula row. The student has just entered "100-4h" but has not yet hit the return key.

Once the student hits return, the system will give flag feedback, highlighting the answer to indicate that the answer is incorrect. In addition, the model-tracing algorithm will find that this particular response can be modeled by using a buggy rule, and since there is a buggy template associated with that rule, the student is presented with the buggy message that is listed in the first row of Table 1. Table 1 also shows three other different buggy messages.

Table 1: Four different classes of errors, and associated buggy-message that are generated by Carnegie Learning's Cognitive Algebra Tutor. The third column shows a hypothetical student response, but unfortunately, the questions are only rhetorical. The ATM is meant to address this.

	Example Errors	**The buggy message generated in response to those errors**	**Possible response by the student**
1	100-4*h - 4*h+100	Does the money earned in your current job increase or decrease as the number of hours worked increases?	It increases.
2	4*h 10+4*h	How many dollars do you start with when you calculate the money earned in your current job?	100 dollars

3	100+h 100+3*h	How much does the money earned in your current job change for each hour worked?	Goes up 4 dollars for every hour
4	4+100*h 100h+4	Which number should be the slope and which number should be the intercept in your formula?	The 4 dollars an hour would be the slope.

Notice how the four buggy messages are asking questions of the student that seem like very reasonable and plausible questions that a human tutor would ask a student. The last column in Table 1 shows possible responses that a student might make. Unfortunately, those are only rhetorical questions, for the student is not allowed to answer them, as such, and is only allowed to try to answer the original question again. This is a problem the ATM architecture solves by allowing the student to be asked the question implied in this buggy message. In this hypothetical example, when the student responds "It increases" then the system can follow that question up with a question like "And 'increases' suggests what mathematical operation?" Assuming the student says "addition" the tutor can then ask "Correct. Now fix your past answer of 100-4*h". We call this collection of questions, as well as the associated responses in case of unexpected student responses, a *tutorial strategy*. The ATM architecture has been designed to allow for these sorts of tutorial strategies that require asking students new questions that foster reasoning before doing, rather than simply hinting towards what to do next.

Table 2: The list of hints provided to students upon request by the Carnegie Learning's Cognitive Algebra Tutor.

Text of Hint
Enter an expression to calculate the money earned in your current job using the hours worked.
First, consider the initial value of the money earned in your current job. Next, consider how the money earned in your current job will change for each hour.
Write an expression that means the same thing as the value of the money earned in your current job plus the change in the money earned in your current job for each hour times the hours worked.
Write an expression that means the same thing as 100+4 times the number of hours worked.
Enter 4.00H + 100.00.

Table 2 shows the hint sequence for this same symbolization question. Notice how the hints get progressively more explicit until finally the student is told what to type. One of the problems with model-tracing tutors is that sometimes students keep asking for a new hint until they get the last most specific hint (Gluck, 1999). However, maybe this is a rational strategy to use when the hints do not efficiently focus on the student's difficulty. Take a moment to consider if the chain of hints in Table 2 is likely to help the student above who just tried "100-4*h"? The 1^{st} and 2^{nd} hints certainly do not address this student's difficulty, and the later hints only do so very obliquely. This lack of sensitivity to the student's cognitive state is an architectural limitation that the ATM architecture is designed to overcome by creating tutors that aim to aid learning by asking the student questions which are focused on the portions that they got wrong. We call this *dynamic scaffolding* and will define this in the next section.

THE BEHAVIOR OF AN EXPERIENCED HUMAN TUTOR

We developed the ATM architecture to be able to build tutors that model the tutorial strategies like we observed in the behavior of an experienced tutor. An example of the behavior of this tutor is shown in the left column of Table 3. This example was collected and transcribed from a one-on-one tutoring session with an eighth grade student working on the “bike-trip” problem (a problem we use as one of our running examples). The right hand side of Table 3 shows a corresponding interaction with Ms. Lindquist and will be discussed later in the section on Ms Lindquist.

Table 3: An experienced tutor's transcript and Ms. Lindquist's attempt to model it.

Experienced Human Tutor Exact Transcript	**The Ms. Lindquist Tutor Similar simulated input.**
240. Student: [reads the "bike-trip" problem] Cathy took a "m" mile bike ride. She rode at a speed of "s" miles per hour. She stopped for a "b" hour break. Write an expression for how long the trip took.	T1: [The "bike-trip" problem was presented] Please write an expression for the total time the trip took.
241. S: uhm [Writes "s/m+b"]	S1: s/m+b
242. Tutor: How do you calculate the amount of time it takes you? If you're, if you're, if you're riding at, let's make it simple. If you are riding at 20 miles per hour, OK, and you go 100 miles, how many hours did that take you?	T2: No. Let me ask you an easier question. Please compute the time actually on the bikes if the speed was 20 miles per hour, and the distance ridden was 100 miles?
243. S: Umh 5	S2: 5
244. T: 5. And how did you get that 5? How did you use the numbers 100 and ?[trails off]	T3: Correct. What math did you do to get that five?
245. S: 100 miles divided by miles per hour.	S3: 100/20
246. T: So you took the miles and divided it by the [garbled, but probably "speed"]	T4: Correct. Now write your answer of 100/20 using the variables given in the problem (i.e., put in 'm' and 's')?
247. S: Miles divided by 's' plus 'b' equals time [writes m/s+b]	S4: b+m/s
248. T: Right.	T5: Correct.

The tutor in the above dialog appears to have done two things to help the student with the problem. First, the tutor focused on the problem of calculating the time actually on the bikes (i.e., the m/s part) by decomposing what was a problem with two arithmetic operators (i.e., addition and division) into a problem that had only one math operator. Presumably, this is because the student indicated he understood that the goal quantity was found by adding for the amount of the break (“b”) to the time actually on the bikes. This is an example of what we call **dynamic scaffolding**, by which we mean focusing the dialog on an area where the student has had difficulty.

The second way this tutor helped the student was to apply what we call a **tutorial**

strategy (similar to what McArthur et al. (1990) called *micro-plans* and what VanLehn et al. (2000) called *knowledge construction dialogs*). The particular tutorial strategy the tutor used is the one we call the *concrete articulation strategy* (Gluck, 1999, Koedinger & Anderson, 1998[4]), which involves three steps. The first step is the *compute question* which involves asking the student to suppose one, or more, of the variables is a concrete number and then to compute a value (i.e., asking the student to calculate the time actually on bikes using 100 and 20 rather than "m" and "s".) The second step is the *articulation question*, which asks the student to explain what math they did to arrive at that value (i.e., "How did you get that 5?"). The final step is the *generalization question*, which asks the student to generalize their answer using the variables from the problem (i.e., line 246). We observed that our experienced human tutor employed this concrete articulation strategy often (4 of 9 problems).

THE ATM ARCHITECTURE

We believe that dynamic scaffolding and tutorial strategies are two pieces that current model-tracing framework does not deal with well, and thus motivate extending the model-tracing architecture by adding a separate tutorial model that can implement these new features and the ATM architecture. Figure 3 shows a side-by-side comparison of the traditional model-tracing architecture.

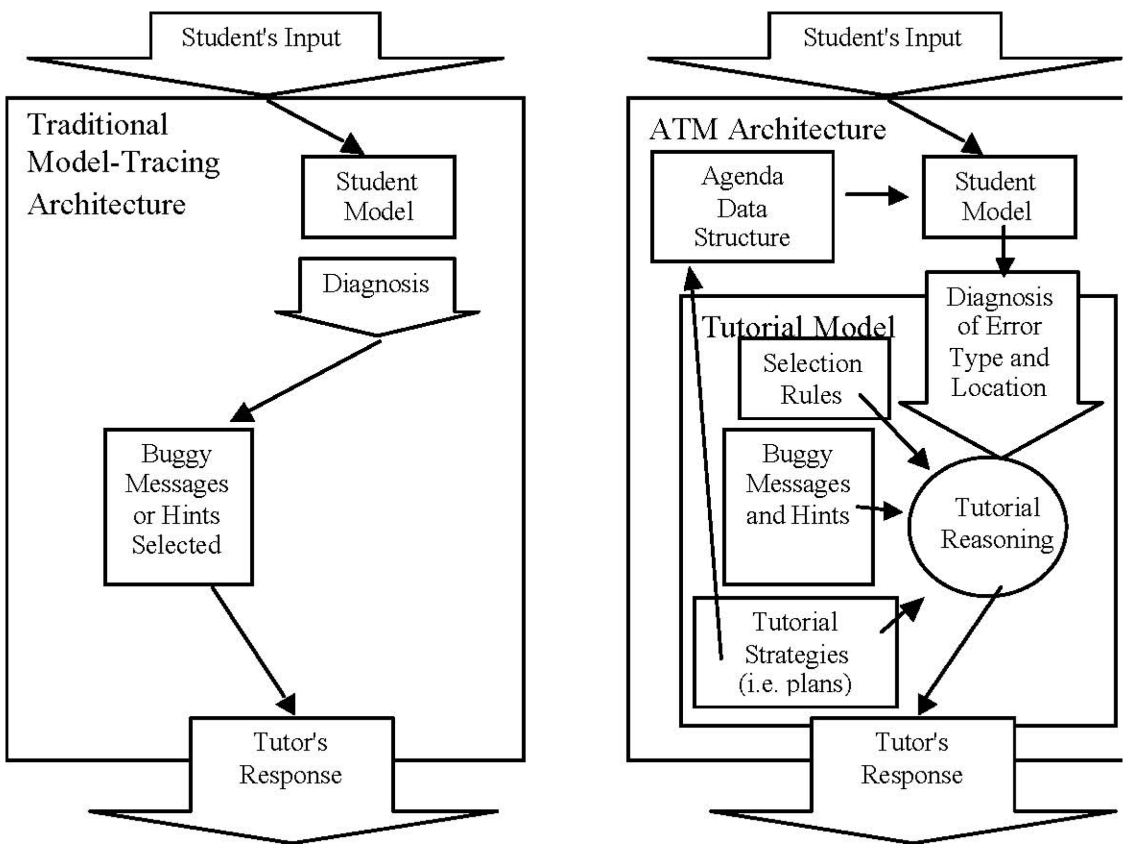

Figure 3: A comparison for the old and the new architectures.

[4] Called the *inductive support* strategy in this prior work.

The traditional model-tracing architecture feeds the students response into the model-tracing algorithm to generate a message for the student but never asks a new question, and certainly never plans out a series of follow-up questions (as we saw the experienced human tutor appear to do above with the concrete articulation strategy). A key enhancement of the ATM architecture is the agenda data structure that allows the system to keep track of the dialog history as well as the tutor's plans for follow-up questions. Once the student model has been used to diagnose any student errors the tutorial model does the necessary reasoning to decide upon a course of action. The types of responses that are possible are to give a buggy message, give a hint or use a tutorial strategy. The *selection rules*[5] shown in Figure 3 are used to select between these three different types of responses. For instance, there is a rule that forces the system to use a tutorial strategy, when possible, as opposed to a buggy message. Another selection rule can cause the system to choose a particular tutorial strategy in response to a certain class of error.

Whereas buggy messages and hints are common between both architectures, the use of tutorial strategies triggered by selection rules makes the ATM more powerful than the traditional architecture, because the tutor is now allowed to ask new questions of the student.

<<< Insert, about here, Figure 4 with the following caption

"**Figure 4**. Flowcharts comparing the ATM Architecture (labeled as the "Ms Lindquist's Architecture") with the traditional model-tracing architecture."

Figure 4 is at (in full resolution) http://nth.wpi.edu/neil/flowchart600.jpg

or as a jpg (reduced file size) as http://nth.wpi.edu/neil/flowchart600.png

We need to crop the right hand side that has a caption and page number and instead put the caption

End insert>>>

The overall algorithm ATM uses is shown in Figure 4, and contrasted with traditional model tracing tutors. The traditional model-tracing architecture includes only buggy feedback and hints. On the other hand, the ATM architecture also includes new elements, as shown by the extra boxes in the flowchart (KCD and KRD are two types of tutorial strategies that will be discussed in the section below on "Tutorial Strategies"). The ATM architecture begins by posing the question that is at the top of the agenda structure, and waits for the student to attempt an answer. Sometimes the student's answer will reveal more information than what was asked for, as in Table 3, response S4, in which the system was expecting an answer of "m/s" but instead received an answer of "b+m/s". Strictly speaking, the student's answer of "b+m/s" is wrong for the question that was asked, however, the tutor would appear pedantic if it said "no"

[5] It should be noted that currently the selection rules used in Ms. Lindquist are very simple. However, selection rules can model complex knowledge, such as when to use a particular tutorial strategy for a particular student profile, or a particular student's error, or a particular context in a dialog. Research will be needed to know what constitutes **good** selection rules, so we have currently opted for simple selection rules.

because "b+m/s" is an answer to a question that is lower down on the tutorial agenda. Therefore, the system treats "b+m/s" as a correct answer to the original question asking for "b+m/s". Having this mechanism in place is part of ensuring reasonable conversational coherence.

The flow diagram shows that if the student gave an answer that is correct for the question at the top of the agenda, the system pops that question off the agenda and proceeds to pose any remaining questions. However, if the student's answer is not correct, the system says "No" and then tries to add any positive feedback before entering the dynamic scaffolding subroutine. That routine tries to come up with the best plan for each error the student might have made for each subgoal. Once the system has planned a response to the first subgoal that had an error, the system will try to do the same for any remaining subgoals that have errors. The integration of model-tracing and dialog is shown in Figure 4. As Figure 4 illustrates, ATM generalizes the functionality of model-tracing (the added boxes on the right) without eliminating any of it (boxes appearing on both sides). We will now describe each of the components of the ATM architecture (Figure 3) with reference to the Ms. Lindquist tutor.

Ms. Lindquist’s Cognitive Student Model

Ms Lindquist's student model is similar to traditional student models. We used the Tertl (Anderson & Pelletier, 1991) production system, which is a simplification of the ACT (Anderson, 1993) Theory of Cognition. As mentioned above, a production system is a group of if-then rules operating on a set of what are called *working memory elements*. We use these rules to model the cognitive steps a student could use to solve a problem. Our student model has 68 production rules. Our production system can solve a problem by being given a set of working memory elements that encode, at a high level, the problem.

To make this concrete, we now provide an example. Figure 5 shows initial working memory encoding the "Anne in a lake" problem. We see that the problem has 5 quantities and two relations that link the quantities together in what we call a *quantitative network*. Our 68 productions can be broken up into several groups. Some productions are responsible for doing a search through the quantitative network to connect the givens with the goal. Other productions are used to retrieve the operator to use (e.g., +, -, *, /). Other productions are used to order the arguments (e.g., 800-40m versus 40m-800). Still other productions are used to add parenthesis when needed. For example, an English version of a production that does the search:

If

You are trying to find a symbolization for an unknown quantity,
And that quantity is involved in a relation

Then

Set goals to try to symbolize the two other quantities connected to that relation,
And set a goal to retrieve the operator to use.

For example, in conjunction with the working memory elements shown in Figure 5, this production could be used to symbolize "the distance Anne has left to row" by setting goals to symbolize 1) "the distance she started from the dock" and 2) "the distance rowed so far", as well as setting a goal to retrieve the correct operator to use.

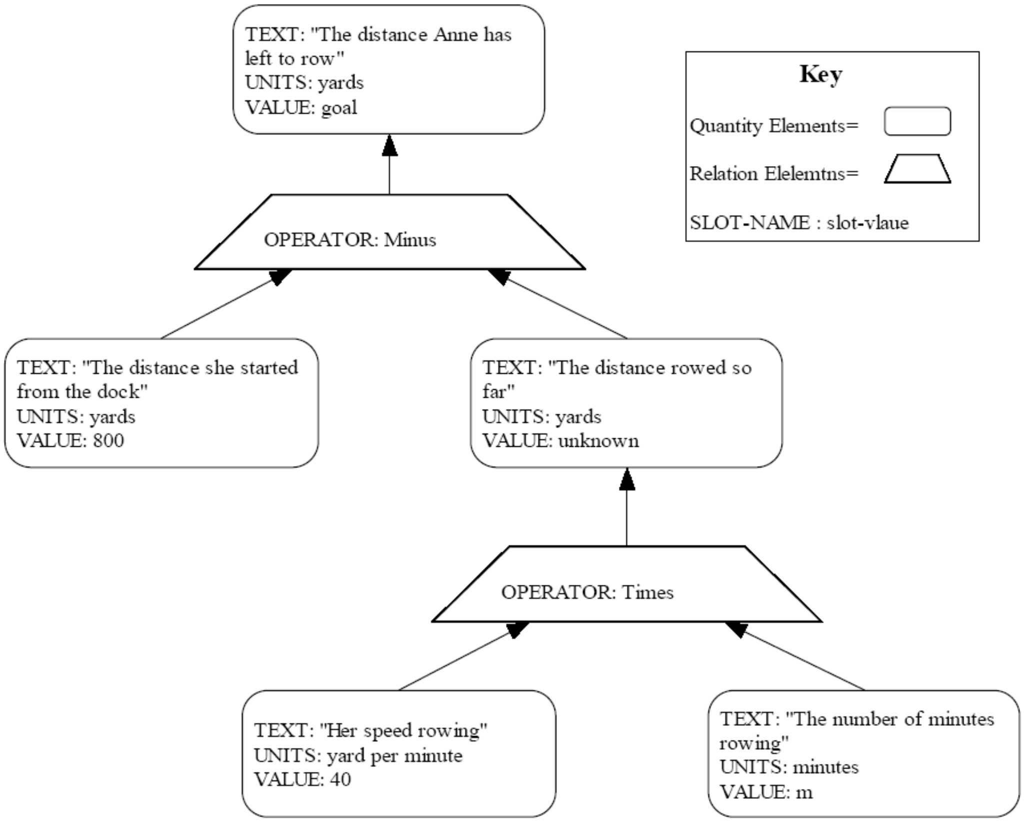

Figure 5: The initial working memory elements for the following problem: *Ann is in a rowboat in a lake. She is 800 yards from the dock. She then rows for "m" minutes back towards the dock. Ann rows at a speed of 40 yards per minute. Write an expression for Ann's distance from the dock.* Answer=800-40m.

We model the common errors that students make with a set of "buggy" productions. From our data, we compiled a list of student errors and analyzed what were the common errors. We found that the following list of errors was able to account for over 75% of the errors that students made. We illustrate the errors in the context of a problem, which has a correct answer of "5g+7(30-g)".

1) Wrong operator (e.g., "5g-7(30-g)")
2) Wrong order of arguments (e.g., "5g+7(g-30)")
3) Missing parentheses (e.g., "5g+7*30-g")
4) Confusing quantities (e.g., "7g+5(30-g)")
5) Missing a component (e.g., "5g+7g" or "g+7(30-g)" or "5g+30-g")
6) Omission: correct for a subgoal. (e.g., "7(30-g)" or "5g")
7) Any combinations of errors (e.g., "5/g+7*g-30" has three errors;1) the wrong order for "g-30", 2) is missing parenthesis around the 30-g, and 3) the "5/g" uses the division instead of multiplication.)

Consider what a good human tutor would do when confronted with a student who wrote what is listed in the 7^{th} item above. Perhaps the tutor would realize that there are multiple errors in the student's answer and decide to tackle one of them first, and plan to deal with the other ones after finishing the first. In contrast, a traditional model-tracing tutor could fire three different bug rules that would generate three different bug messages and then display all three to the student. This seems to make the tutor appear more like a compiler spitting out error messages. ATM deals with each of the errors separately. Dealing with more than one error occurring at the same time (such as the 7th item in the list above), is something that Anderson's traditional model-tracing tutors do not do well, and that is probably due to the fact that the pedagogical response of such tutors is usually a buggy message. This is not to say that model-tracing tutors have *never* dealt with more than one student error occurring simultaneously; some cognitive modelers have tried to compensate for the architecture's lack of support for more than one error at a time, by writing single rules that will model two errors occurring at the same time. However, this makes the modeling work even harder.

Ms. Lindquist's Tutorial Model

Now we will look at the components of the tutorial model shown in Figure 3. A fundamental distinction in the intelligent tutoring system is between the student model, which does the diagnosing, and the tutorial model, which does everything else. The tutorial model is implemented with 77 production rules.[6] Some of these production rules are the selection rules shown in Figure 3, that do the selection of what type of response to make. Other rules do different things. For instance, some rules specify how to implement a particular tutorial strategy while others know when to splice in positive feedback.

Since using a tutorial strategy involves asking a series of questions, we will first state the questions Ms. Lindquist currently knows how to ask a student.

Tutorial Questions

Ms Lindquist currently has the following tutorial questions:[7]

1) Q_symb: Symbolize a given quantity ("Write an expression for the distance Anne has rowed?")
2) Q_compute: Find a numerical answer ("Compute the distance Anne has rowed?")
3) Q_articulate: Write a symbolization for a given arithmetic quantity. This is the articulation step. ("How did you get the 120?")
4) Q_generalize: Uses the results of a Q_articulate question ("Good, Now write your answer of 800-40*3 using the variables given in the problem (i.e., put in 'm')")
5) Q_represents_what: Translate from algebra to English ("In English, what does 40m represent?" (e.g., "the distance rowed so far"))
6) Q_articulate_verbal: Explain in English how a quantity could be computed from other quantities. (We have two forms: The reflective form is "Explain how you got 40*m" while the problem solving form is "Explain how you would find the distance rowed?")
7) Q_decomp: Symbolize a one-operator answer, using a variable introduced to stand for a sub-quantity. ("Use A to represent the 40m for the distance rowed. Write an expression for the distance left towards the dock that uses A.")
8) Q_substitute: Perform an algebraic substitution ("Correct, that the distance left is given by

---

[6] Our use of a production system for tutorial modeling is similar to Freedman's (2000).

[7] Each example is illustrated in the context of the student working on the following problem: "Ann is in a rowboat in a lake. She is 800 yards from the dock. She then rows for "m" minutes back towards the dock. Ann rows at a speed of 40 yards per minute. Write an expression for Ann's distance from the dock."

800-A. Now, substitute “40m” in place of A, to get a symbolization for the distance left.”)

You will notice that questions 1, 3, 4, and 8 all ask for a quantity to symbolize. Their main difference lies in when those questions are used, and how the tutor responds to the student’s attempt. Questions 5 and 6 ask the student to answer in English rather than algebra. To avoid natural language processing, the student is prompted to use pull down menus to complete this sentence “The distance rowed is equal to <noun phrase> <operator> <noun phrase>.” The noun phrase menu contains a list of the quantity names for that problem. The operator menu contains “plus”, “minus”, “times” and “divided by.” Below we will see how these questions can be combined into multi-step tutorial strategies.

Tutorial Agenda

The tutorial agenda is a data structure that operates somewhat like a stack. It is used to keep track of the current focus. It includes the questions that have been asked already of the student but are still awaiting a correct response, as well as questions that the tutor plans to ask but has not yet done so. The question at the top of the agenda represents the current question that the student was just asked. If the tutor invokes a tutorial strategy, it places the new question on the agenda to be asked. As students answer questions, they are removed from the agenda.

Tutorial Reasoning: Dynamic Scaffolding

A diagnosis is passed from the student model to the tutorial model. If the student's response is correct, the system pops that question off the agenda. However, if it is not, the dynamic scaffolding procedure requires that for each error the student made, the system come up with a plan to address it. Dynamic scaffolding is based upon the fact that human tutors tend to ask questions related to incorrect aspects of the student's answer. This *error localization* communicates valuable information to the student by focusing the student's attention on a single aspect of what might have been a complicated problem-solving process. The dynamic scaffolding procedure can also give positive feedback on correct aspects of the student's reasoning when appropriate. The dynamic scaffolding procedure does the error localization and then passes responsibility to the selection rules to determine what is the most pedagogically effective tutorial strategy to employ for the given situation. The next section details the options Ms. Lindquist has.

Tutorial Strategies

This section will show several different tutorial strategies that Ms. Lindquist can use. Some strategies we observed that the human tutor used seemed to apply only if the student made a particular type of error and we call such strategies *Knowledge Remediation Dialogs* (KRD). Other strategies the tutor used were more broadly applicable and we call such strategies *Knowledge Construction Dialog*[8] (KCD) Both KCD and KRD invoke multi-step plans to deal with particular errors, however the KRD is only applicable if the student has made a particular type of error. For instance, a dialog about the role of order of operations shown in Figure 6, would be a KRD, because it applies only in the case the student's error was to forget parentheses. However, the concrete articulation strategy is a KCD, because it can be used no matter which specific error type might have occurred. Since KRDs apply in fewer situations, we have first

---

[8] We borrow the term *knowledge construction dialog* from VanLehn.

focused on authoring KCDs, and have implemented only one of the KRDs we observed the experienced tutor use. That KRD is applicable when the student has made an *error of omission*, by which we mean that the student correctly symbolized only a piece of the problem. For example, suppose the student was supposed to say “800-40m” but instead said “40*m”, the tutor would digress using the one-step KRD that asks the student to identify what the “40*m” represents, and once the student has clarified what the 40*m represents, the student is then asked to symbolize the whole quantity again. ATM has four different types of responses. Each of these is shown in Table 4, illustrated with an example from the Ms Lindquist tutor.

Table 4: The four different types of tutorial responses: hints, buggy-messages, KRD and KCD.

	Two types of Tutor Reponses	
	Tell the student something	**Ask a New Question(s)**
Always Applies	Traditional model-tracing hint: e.g., "You need to use 40 and 'm' to find the distance rowed."	Use one of the four KCDs such as the three-step *Concrete Articulation Strategy*
Applies selectively	Traditional model-tracing buggy-feedback message: e.g., "You are missing parentheses around the '30 G'."	Use a KRD such as shown in Figure 6 that only applies when the student has forgotten parentheses.

Student: **The answer is 5*g+7*30-g.**

Tutor. The 5*g is correct for the amount bagging groceries, but the amount delivering newspapers is not 7*30-g. Let me ask you this about the rules for order of operations. Which do you do first, multiplication or subtraction.

Student: **multiplication**

Tutor: Correct. In this problem, do you want the multiplication done first or the subtraction in the expression 7*30-g?

Student: **The subtraction.**

Tutor: Good- so what do you need to use to indicate that the subtraction should be done first.

Student: **Parentheses.**

Tutor: Good. So try this question again, but this time use parentheses. …

Figure 6: An example of a knowledge remediation dialog. Note that the dialog is focused on the location of the error, as well as the type of error.

We have been using the term *selection rule* to describe the rules that determine what the best tutorial response to make is, given the entire context so far. Because this field is so new, and tutoring is so complicated, our selection rules are currently simple heuristics, which will need to be refined by further research. For instance, when the system has multiple different responses to choose between, its selection rules will try to put them in the following order; KRD, Buggy-Message, KCD, and finally hint. The heuristic for ordering them in this manner is to respond with the response that takes into account as much context as possible (KRD & Buggy Message). The second heuristic is to use a tutorial strategy (KRD or KCD) before using a buggy message or hint, because we would rather ask a question than give a hint. These heuristics are examples of selection rules.

We will now look at four different tutorial strategies (all KCDs) that Ms Lindquist uses;

1) "Concrete Articulation", 2) "Introduced Variable", 3) "Explain in English First" and finally 4) "Convert the Problem into an Example to Explain."

Concrete Articulation Strategy

Our experienced human tutor used the concrete articulation strategy on 44% of the problems she tutored. This strategy uses three questions of type Q_compute, Q_articulate and Q_generalize. The right-hand side of Table 3 shows an example dialog generated by Ms. Lindquist, which closely parallels the experienced human tutor dialog. We will walk through what Ms. Lindquist did here and illustrate both dynamic scaffolding as well as the particular tutorial strategy that Ms. Lindquist used. First, the student model diagnosed the student's response (“s/m+b”) as being partially correct (i.e., the student understood that the total trip time was the time of the break, “b”, plus the time actually on the bikes) and partially wrong (i.e., the system noted the bug of reversing the order of the arguments (i.e., "m/s" vs. "s/m"). As part of the dynamic scaffolding routine, the system needed to come up with a plan to address the error related to the time on bikes. (This is similar to the way the human tutor also ignored the part of the problem related to the time for the break and instead focused on the goal of getting the time actually on bikes.) In this example, the tutorial model chose to use the *concrete articulation* strategy. Therefore, the tutorial model generated questions for the *compute, explain/articulation* and *generalization* steps and placed these three questions on the agenda.

After the student answers the compute question correctly (at S2), the tutor decides to continue with its plan. However, if the student had typed 100/20 instead of 5, then both the compute and explain questions would be removed from the agenda and the tutor would skip to the generalize step. An example of this sort of flexibility to interpret the student's response broadly is demonstrated by the tutor's response to "b+m/s" (at S4). The question at S4 was actually asking for the student to answer "m/s". Only a pedantic tutor would not accept that answer as it indicates an understanding that subsumes the understanding required for the posed question.

To show you an example of how the previous situation could have come out differently, let’s look at the following dialog.

T1: Hello. [Doing the “bike-trip” problem] Please write an expression for the total time the trip took. [Q_symb]

S1: **help!** [Student hit the help button]

T2: Let me ask you an easier question. Please compute the time actually on bikes if instead the speed was 20 miles per hour, the time of the break was 2 hours, and the distance ridden was 100 miles? [Q_compute]

S2: 7

T3: Correct. What math did you do to get that 7? [Q_articulate]

S3: 2+5

T4: Okay, but how did you get the 5? [Q_articulate again but this time for the 5.]

S4: **100/20**

T5: Correct. Now, let’s try this question again. What math did you do to get the 7? [Q_articulate]

S5: **2+100/20**

T6: Correct. Now write your answer of ‘2+100/20’ using the variables given in the problem (i.e., put in ‘m’, ‘b’ and ‘s’)? [Q_generalize]

S6: **b+m/s**

T7: Correct.

This example is interesting in the way Ms. Lindquist responds at T4 with what we call an embedded sub-dialog. In this case, the student model reported that the student's answer of "2+5" is a partially complete articulation but that the student didn't spell out where the "5" came from. The tutorial model is able to "dig-down" into the student's reasoning with an embedded sub-dialog asking for an explanation of where the 5 came from. Once that was accomplished, the tutor could proceed with the rest of the plan (As indicated by saying "Now, let's try this question again.")

Croteau, Heffernan & Koedinger (2004) and Freybeger, Heffernan & Ruiz (2004) looked at tutorial log files from hundreds of students who were tutored by Ms. Lindquist and tried to find the best fitting cognitive models to explain students' responses. Croteau et al (2004) was interested in the question of whether there is evidence that an *articulate* skill is what makes writing expressions in the language of algebra difficult. This study found statistically significant evidence that students were finding "articulating" more difficult than "comprehending" word problems.

Introduced Variable Strategy

A second example of a tutorial strategy is what we call the *Introduced Variable Strategy*, which we observed our human tutor employ on close to half of the problems she tutored. This strategy uses both Q_decomp and Q_substitute questions as shown below.

T1: Hello. Adam, Stan and Doug spent the day picking strawberries. Adam picked twice as much as Stan and Doug picked 7 more kilograms than Adam. Suppose Stan picked "s" kilograms of strawberries. Stan and Doug cooperated to make a pie. Please write an expression for the combined number of kilograms for Stan and Doug.

S1: 7+(2*s) [{201 seconds[9]} This answer is partially correct but should be s+7+(2*s)]

T2: No. But, "7+(2*s)" is correct for the number of kilograms Doug picked. We will come back to the combined number of kilograms for Stan and Doug. Let me try to break this down for you. If "C" were the number of kilograms Doug picked then what would be the combined number of kilograms for Stan and Doug? [This is the Q_decomp question]

S2: c+s {77 seconds}

T3: Correct. Now, let's try this question again: You are correct that "s+C" is the combined number of kilograms for Stan and Doug. Now, can you state this in terms of the numbers given in the problem? [This is the Q_substitute question]

S3: 7+(2*s)+s { 44 seconds}

T4: Correct.

Our previous research (Heffernan & Koedinger, 1998) uncovered evidence that suggested that this strategy could be effective. Because students have difficulty writing expressions that have multiple variables, by introducing the new variable (i.e., C) at T2, the tutor reduces the number of operators the student has to write. Only after the student has gotten confirmation (that you need to add the amount Doug picked and Stan picked, using "c+s") do we ask the student to put it all together by substituting in the correct expression for "C" (i.e., T3).

---

[9] This example is from a 10th grader using Ms. Lindquist in our lab study. The "{}" brackets indicate the seconds the student took to respond.

"Explain in English" Strategy

Mathematical communication is increasingly seen as a goal for students, and in particular translating between an algebraic representation, an English verbal representation, and a concrete representation of numerical instances (Schoenfeld et al., 1992, Nathan & Koedinger, 2000). Other researchers are also interested in wanting an ITS to elicit explanations from students (e.g., Conati et al., 1997.) Our experienced human tutor would often ask the student to explain how to compute a quantity in English and only afterward, ask the student to say it in symbols (Heffernan, 2001). Therefore, we have implemented a strategy that is based on the hypothesis that students will learn more if they are first asked to explain in English how to compute a quantity, before trying to symbolize it. The following example shows this strategy:

T1: Hello. [Doing the "bike-trip" problem] Please write an expression for the total time the trip took. [Q_symb]
S1: **m/s**
T2: What does "m/s" represent (e.g., the speed on the bike?)[Q_represents_what]
S2: **The total time of the trip.**
T3: No, "m/s" represents the actual time on the bikes. Can you explain how you would find the total time of the trip using the time on the bikes? [Q_articulate_verbal]
S3: **The total time of the trip is equal to [1st menu] "the amount of time for the break" [2nd menu] "plus" [3rd menu] "the actual time on the bikes". [Composed using three pull down menus.]**
T4: Good, now say what the total trip time is in symbols [Q_symb]
S4: **b+m/s**
T5: Correct

"Convert the Problem into an Example to Explain" Strategy

This next strategy was not based on either our own empirical work, nor was it based on what the experienced tutor did. Instead, this strategy was inspired by literature and the suggestions that many other researchers have made about what makes for effective tutoring. For instance, Zhu & Simon (1987) suggest a similar strategy of presenting students with worked out examples. See also Renkl (1997), VanLehn (1998), Sweller & Cooper (1985) for related studies of student use of worked examples. Catrambone (1996) also suggests that learning from examples can be improved by asking questions that focus on the goal structures of the domain. Also, to the degree that this strategy focuses on asking students to explain the quantitative structure in English (using pull-down memus), rather then simply focusing on problem solving, the literature suggests that the *self-explanation effect* (Chi, 1996) should make this an effective strategy. For instance, Chi (1996), Conati, Larkin & VenLehn (1997), Mwangi & Sweller's (1998), and Aleven, Koedinger, & Cross (1999) have all proposed, and in some cases demonstrated that, asking students to explain can be beneficial. Most concretely of all, this strategy also gets at the goal of practicing translation from the algebraic representation to the verbal representation, which Corbett McLaughlin, Scarpinatto, & Hadley (2000) showed to be effective for learning symbolization. Here is an example dialog:
T1: [Doing the "bike-trip" problem] Please write an expression for the total time the trip took.

[Q_symb]
S1: **m*s-b**
T2: Hmm. No. The answer is actually "b+m/s". Let me ask you some questions to help you understand it. Can you explain where the "m/s" came from [Q_articulate_verbal]
S2: **The "m/s" is equal to the distance ridden divided by the speed ridden.**
T3: Correct. What does "m/s" represent? [Q_represents_what]
S3: **The time actually on bikes**
T4: Correct. Can you explain where the "b+m/s" came from? [Q_articulate_verbal variant]
S4: **The "b+m/s" is equal to the hours for break plus the time actually on bikes.**
T5: Correct. What does "h+m/s" represent? [Q_represents_what]
S5: **The total time the trip took**
T6: Correct.

Notice how at T2, the answer to the original question is given, and then the student is asked to explain the answer by translating the components back into English.

This completes the review of the strategies Ms. Lindquist has. We ran some experiments to determine which are most effective and under what conditions.

EXPERIMENTAL EVALUATIONS

In this section we review some of the experimental evaluations that have been done with Ms Lindquist. Unfortunately, none of the evaluations we present compare ATM versus model-tracing. We would have liked to compare Ms Lindquist to Cognitive Learning's model tracing tutor, however this was not possible as the Cognitive Learning tutor had been licensed and was not available for a study. (Razzaq and Heffernan (2004) did compare a model-tracing tutor to one that incorporated tutorial dialog for solving linear equations and found favorable results for the tutor with dialog.) Therefore, we report on some evaluations of the different tutoring strategies in Ms. Lindquist. The measures of interest include measures of how much learning is occurring and under what conditions, as well as measures of motivation.

Experiment 1

The first experiment was designed to replicate the results of Heffernan (2001), which showed that if the number of problems was controlled for, rather than time on task, students learned more from a dialog with Ms. Lindquist, than if they were simply told the answer. After collecting data for several months, we analyzed 3800 individual log files. About 2000 of them did not get beyond the 10-15 minute tutorial, and therefore never began the curriculum. Another 500 more did not do more than a single problem in the curriculum.[10] These groups were dropped from the analysis.

Hundreds of students were thrown out of the analysis if they got the first two problems correct, and therefore did not receive any tutoring. We were left with 623 student files for analysis. Our goal was to find out which of the tutorial strategies let to the greatest learning. We used a mastery-learning algorithm that for the first curriculum section pushed them onto the next

[10] Many individuals skip the demonstration, and then realize that this tutor does not address the skills they are interested in, such as symbolic manipulation. Many students are disappointed that the web site does not allow them to submit their own problems, such as their homework problems.

section after getting 2 problems correct in a row. The results we report on relate to the first curriculum section. Once a student reached the mastery criterion of two problems correct in a row, the student was given a two-item posttest.

There were actually three different experimental conditions in this experiment, with each condition being represented by one of the tutorial strategies mentioned in the introduction. The control condition was the one described in the introduction that told students the answer when they got it wrong and then went on to do more problems.

Results for Experiment 1

While doing the analysis for this experiment, we encountered a problem that we should have anticipated. Students that were placed into the control condition used the system for a shorter period of time than those in the experimental condition. This "drop-out" rate was significantly higher in the control condition than in any of the experimental conditions. Of the 623 individuals analyzed, 47% of the 225 that received the control condition dropped out, while only 28% of the other 398 dropped out. This difference was statistically significant at the $p<.01$ level. There was no statistically significant difference between the drop-out rates of the three experimental conditions.

Because of this massive selection effect, we do not bother to report any detailed analysis of the learning results[11]. We will note that the "Explain in English First" tutorial strategy seemed to be the most effective for the first curriculum section, while the "Concrete Articulation" strategy appeared to be the most effective for the second curriculum section. These were merely "suggestions" and not to be taken too seriously, due to the potentially serious threat to the validity of this experiment because of the selection effect related to dropouts. We did however use these as guesses in picking which of the tutorial strategies to use in our more refined experiments (Experiments 2 & 3).

We conclude that, as far as from a motivational point of view, the intelligent feedback was superior at getting students to persist in tutoring. We now move on to report Experiment 2 and 3.

Experiment 2

After Experiment 1, we made several changes to the system including coming up with a way to deal with the drop-out problem, by focusing our analysis only to those students that were doing the tutor as part of a class assignment. A student entering the Ms Lindquist tutoring site was asked if they were students as well as if they were being required to do Ms. Lindquist by their teacher. If they were being required, they were asked for their teacher's name. Over a period of a few months, we collected over a hundred such files, most of them from 5 different teachers. The teacher that sent the largest number of students to the site, whom we will call "Mr. X", sent about 76 students. We decided to analyze just these students.

We know little about Mr. X, but we can infer some things. From the time-stamps on these files, it appears the students used the system during two classes (One on a Monday, and the other a Friday), and did not return to the system for homework (which is possible since it is running over the internet). Every student clicked on the button indicating they were "In 7th or 8th grade". Furthermore, it appears that Mr. X's students were from three different classes. We can only guess that Mr. X is a math teacher. This person took three groups of students to a computer lab (as indicated by time stamps), and supervised them while they worked through the tutor.

[11] Because we did not have a pretest, we could not determine if it was weaker or stronger students that were responsible for the increased drop-out rates. In part, to help to deal with this issue, in the version used in Experiments 2 & 3 we included a pretest.

There is a mechanism for students to request the system to send a progress report to their teacher, but this facility was only used by a few students, so it seems likely that Mr. X did not grade his students according to the measures the systems provides. We also have no idea if this teacher was walking around the lab helping students, or even if the teacher was present. Regardless of whether students were being given assistance by their teacher[12], we have no reason to believe that he would be helping students in the control condition any differently than those students in the experimental condition, so we therefore believe these results to be worth considering. As an experimenter used to conducting studies in classrooms, this sort of information is often important to understand why the experiment turned out the way it did, and of course, it would be nice to have that sort of information for this experiment.

These results were also collected using a slightly different version of the software in which we added a pretest that matched the posttest, thereby allow us to look at learning gains. Another change was the fact that this version controlled for time on task by giving the posttest after a set period of time (that varied according to the curriculum section but was somewhere between 6 minutes to 15 minutes). After the posttest, students were moved onto the next pretest if they had already reached the mastery criterion, or were given more practice if they had not yet reached the mastery criterion.

We report results from the experiment that was run on the first curriculum section as Experiment 2 and will report the results from the second curriculum section as Experiment 3. There were not enough students who finished the third curriculum section to analyze. The experimental condition in Experiment 2 received the "Explain in English" tutorial strategy, while in Experiment 3 the experimental condition received the "Concrete Articulation" strategy.

Table 5: Learning Gain and # Problems Completed within time limit, showing students did not learn much in the first curriculum section. This appears to be due to ceiling effects.

		Gain (Pre-Post) in # Probs. (p=.54)			# Problems Done. (p=.0003)	
	N	Mean	Std.Dev.	P(Mean=0)	Mean	Std.Dev.
Control	33	.121	.545	.21	8.364	4.656
Experiment	29	.034	.566	.75	4.621	2.678
Total	62	.081	.552	.25	6.613	4.267

Results for Experiment 2

Mr. X had 76 students to begin with. We excluded 14 of them because they got every practice problem correct, and therefore received no corrective feedback of either the experimental type (i.e., dialog) or of the control type (i.e., being told the answer to type in). The results are summarized in Table 5.

Not surprisingly, since engaging in a dialog takes more time then simply being told the answer, students in the control condition solved a statistically significant larger number of (8.3 problems versus 4.6 problems, p<.0003) in the same period of time.

Unlike Experiment 1, (where there was a confound caused by more drop-outs in the control group) all of Mr. X's students completed the first section. We refer to the difference between the pretest score and the posttest score as the *learning gain*. Between pretest and posttest, a period of time lasting 6.5 minutes, students learning gain was an average of .081 problems (which is a 4% gain). This difference was not statistically significant for any of the individual conditions (i.e., meaning the hypothesis that the mean was significantly different than zero was not supported), nor overall. The reason students did not appear to learn in this section is

[12] Student might have been helping each other, but the fact that the problems the students saw were randomly ordered helps to mitigate against cheating by making it harder for a student to just copy answers from each other.

probably due to the fact that students came in already knowing this skill rather well (pretest scores=1.58, or 79%, with 40 of 62 students getting both pretest problems correct, evenly split between conditions). Given that there is no evidence of learning, it is not surprising that there was no statistically significant effect of condition upon learning gain (p=.54). We now turn to the results of the second curriculum section where we will see that there was no problem of students entering with too much knowledge.

Experiment 3

After completing the first section, Mr. X's students were either moved onto section 2 or given more practice on Section 1, if they had yet to demonstrate mastery by getting two problems correct in a row. Two students did not even get to the second section, due to this requirement[13].

The time between pretest and posttest was 10 minutes. The students went on to the second curriculum section that involved writing expressions that had two-operators (e.g., 800-40*m). This is what we report as Experiment 3. The control condition was the same as in Experiments 1 and 2. Students in the experimental condition received the Concrete Articulation Strategy for feedback.

Results for Experiment 3

The problems that students solved during this experiment were harder than those of Experiment 2, as measured by the fact that of the 74 students who completed the posttest, their average score on the three items was 1.068 correct (or 36% correct). Therefore there was much less chance of a potential "ceiling effect" than in Experiment 2. During the tutoring session, students got 39% of the problems correct on the first try and therefore received tutoring on the remaining 61% of problems (those in the control condition were again simply told the answer).

61 of Mr. X's students went on to complete section 2. Three of them never made any errors, so were dropped from the analysis since they received neither the control nor the experimental feedback. Unlike in Experiment 1, there was no reliably different drop-out rate due to condition (8 in the control condition did not finish, while 7 in the experimental condition did not finish). This lack of an interaction between conditions and drop-out rate suggests that the method of looking at students who were required by their teacher to work on Ms. Lindquist appears to be a nice way to avoid the confound of differential drop-out rates between conditions.

Table 6: Students learned more even while doing fewer problems.

		Gain (Pre-Post) in # Probs. (p=.12)			# Problems Done. (p=.0001)	
	N	Mean	Std. Dev.	P(Mean=0)	Mean	Std. Dev.
Experiment	29	.483	.688	.0008	3.483	1.902
Control	29	.138	.953	.44	6.897	3.063
Total	58	.310	.842	.007	5.190	3.058

Given that time was controlled for, it is not surprising that the average number of problems done differed significantly (p<.001) by condition (Control=6.9 problems, Experiment=3.5) (See Table 6).

Averaged over both conditions, the average learning gain of 0.31 problems (or 10% for the 3 problem pre-post test) was statistically significant (p<.007 when compared with the null hypotheses of the learning gain being equal to zero). Interestingly, the learning gain in the control condition was 0.138 problems, while in the experimental condition it was 0.483 problems.

[13] One student did 34 one-operator problems in a row, never getting two correct in a row. This probably suggests a student who was not reading the problems, and was simply typing in the answers provided by the computer. The student did happen to be in the control condition, where this is possible.

This difference in learning gain between conditions approached statistical significance (F(1,56=2.5),p=.12). The effect size[14] was a respectable .5 standard deviations. Figure 6 shows that even though students in the experimental condition solved about half as many problems, they learned more while doing so.

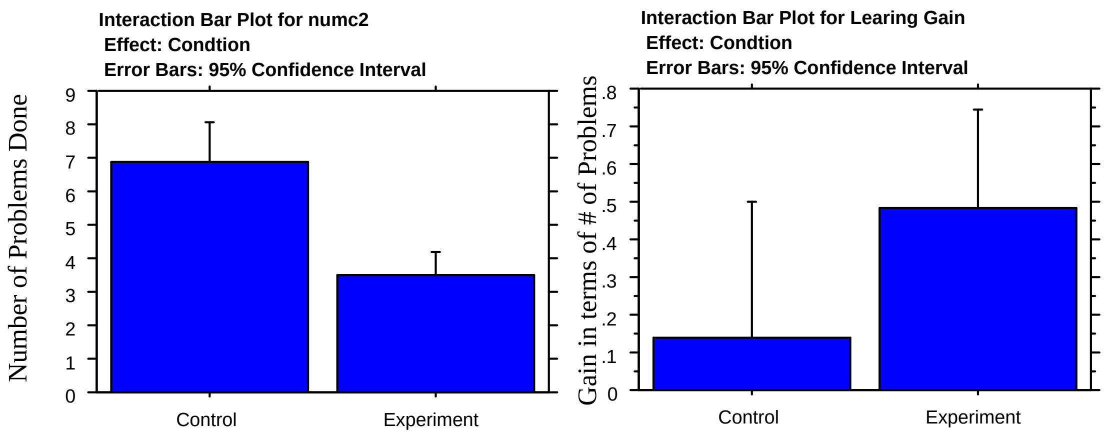

Figure 6: Students did almost half as many problems in the experimental condition (left), but had higher learning gains (right) between pretest and posttest of close to ½ a problem out of a 3-item test, for a gain of about 16%.

Table 7: Student's learning gain (or loss) broken down by condition.

Learning Gain	Experiment	Control	Total
-1	2	7	9
0	12	14	26
1	14	6	20
2	1	1	2
3	0	1	1
Total	29	29	58

Table 7 shows how the average learning gain of 0.31 problems, reported above, is broken down by condition. We see that the students in the experimental group tested to learn more on average. There was one student that had a learning gain of 3 problems and this person was in the control group. Upon inspection of this student's file, we found that the student did not complete two of the pretest items (probably just hit the return key instead of answering them). Furthermore, this student did only two practice problems before getting to the posttest. On the second practice problem this student got the wrong answer and then was told the answer, however he/she refused to type that answer in and instead typed "garbage" answers for 30 consecutive turns. It seems reasonable to consider how sensitive the results reported above are to the presence of this one student that appears to be an "outlier" student who is over three standard deviations from mean for all students. It turns out that if this student is excluded, then the average learning gain in the control becomes a very small .04 problems. Our tests of statistical significance tell us that this small number is not statistically significantly different than zero (p=.55) leading us to reject the hypotheses that student's pretests and posttests results differ significantly. Furthermore, the

[14] Effect size is defined as the difference between the two groups divided by the standard deviation of the control group.

interaction between condition and learning gain switches from marginal significance to become statistically significant (from p=.13 to p=.03). The effect size goes from .35 to .56 standard deviations. This further supports the hypothesis that students really did learn more in the experimental condition, even though they did fewer problems.

Hypothesis: Dialog encourages learning because it is viewed as a penalty

Looking for instances that seemed to suggest where a learning event might have happened, we read over all the student transcripts that showed a learning gain in the experimental condition in Experiment 3. We failed to find examples of what appeared to be clear examples of what looked like great tutoring. (The students themselves seemed to show lots of "sloppiness"). Because of this, we wondered why did the experimental condition show higher learning gains than the control condition? One alternative hypothesis to explain these results is that students in the experimental condition were more motivated to get an answer correct because they perceived the ensuing dialog as a penalty. In the control condition students can take a guess at a problem, and if they are wrong they are simply told the answer, but in the experimental condition, they will get asked new questions which they might view negatively.

To guard against this hypothesis explaining our results, we looked to see if students in the experimental condition of Experiment 3, spent more time composing their response on the posttest than those students in the control. It turned out that both groups took the same amount of time (The experimental group took 58 seconds while the control group took 60 seconds, a difference that was not statistically significant (p=.8)). It is also true that both groups took the statistically significant same amount of time to compose their initial response for each new problem during the practice period. (The experimental group took 71 seconds while the control group took 70 seconds, a difference that was not statistically significant (p=.6)). Therefore, the hypothesis that students might learn more from dialog because they view the dialog as a penalty, and consequently concentrate more, seems not to be supported by the data.

Experiment 4

In Mendicino & Heffernan (submitted) Ms Lindquist was compared to both: 1) classroom instruction and 2) Computer Aided Instruction (CAI). This work tried to quantify the value added of CAI over classroom instruction, versus the value-added of ITS on top of CAI..

Results for Experiment 4

One result was that both computer based versions out-performed the classroom teachers, replicating Kulik (1994) studies showing benefits for computer instruction compared to traditional classroom controls. (Leena-lets cite. We hypothesize that this is mainly due to the benefit of immediate feedback.

A second result of Mendicino & Heffernan’s study found that the value-added of the intelligent tutoring on top of CAI was substantial (measured in terms of effect size was about .4 standard derivations) suggesting that the more intelligent version was more effective at promoting learning. Mendicino & Heffernan also did an experiment trying to replicate the motivational results reported in Experiment #1 above, by randomly assigning students into two homework conditions; either the CAI condition or the Concrete Articulation “intelligent” strategy. They again found a motivational benefit in that students getting the more intelligent version would persist longer. However, given the short length of the experiment this benefit might quickly evaporate over time.

DISCUSSION

It is interesting to note that in the last few years there has been an increase in interest in building dialog-based systems. However, dialog systems are not new; Carbonell (1970) built one of the early computer tutors over 30 years ago and it was dialog-based. Since that time, many educational technologies have instead relied on elaborate graphical user interfaces (GUI) that reify parts of the problem solving process (e.g., the reification of subgoals by Corbett & Anderson, 1995). One possible benefit of dialog-based systems, is that students do not have to spend time learning a new interface. This seems particularly important if the tutoring system has multiple different tutorial strategies that encourage different ways of solving problems. Therefore, the student does not have to learn multiple different GUIs for each different method.

We have released Ms. Lindquist onto the web at www.AlgebraTutor.org, where she has been used by thousands of students and teachers. Ms. Lindquist has also won various industry awards from teacher related web sites (e.g., the National Council of Teachers of Mathematics). So far, we have learned that the dialogs that Ms. Lindquist has with students do lead to better learning, compared to simply telling students the answer as well as the fact that student appear to get motivated. Future work will focus on examining if the benfit of this type of tutoring is worth the additional time these dialogs require.

While Anderson's model-tracing development system was designed to allow the tutor to **tell** students how to get back on track, the ATM architecture is designed to **ask** students questions, which is more like what human tutors do. However, it remains to be seen if the ATM architecture will enable the building of tutors that are more effective than model-tracing tutors. We plan to address this question by comparing the Ms. Lindquist tutoring system to a control version that uses only the traditional model-tracing forms of feedback (buggy messages and hints). We are also currently running experiments comparing the effectiveness of the different tutorial strategies Ms. Lindquist has. We are also interested in generalizing this architecture further by building a set of authoring tools for content experts to be able to author similar intelligent tutoring systems.

We are currently using the web site to run experiments in which each condition of the experiment uses one of the four tutorial strategies. These experiments will tell us which one strategy is most effective (if you are only going to have a single strategy). Later, we want to learn "Under what conditions is it best to use tutorial strategy X versus tutorial strategy Y?" For example, it might be best to use the concrete articulation strategy for problems that include only a few arithmetic operations. Alternatively, maybe there is utility in using multiple different strategies. Answers to these questions can be found by systematically experimenting with the selection rules used by the system. Arroyo et al. (2000) provides a nice example of a selection rule; students who score low on a Piagetian test perform better if given instruction that is more concrete, while high scoring students learn better with instruction that is more formal. Arroyo et al. (2001) have also found evidence suggesting boys are less likely to read hint messages and benefit from less interactive hints. We plan to use Ms. Lindquist to discover progressively more detailed selection rules. As we run more experiments, refining our selection rules and adding new tutorial strategies, we will be creating a concrete theory of tutoring for symbolization that makes specific recommendations. Some of the tutor's behaviors will be shown to be more helpful than others. Of course, we will never reach the perfect tutoring model, but by making our theories about tutoring concrete, we accumulate a body of useable knowledge about what makes for good tutoring.

CONCLUSION

McArthur et al. (1990) criticized the model-tracing architecture "because each incorrect rule is paired with a particular tutorial action (typically a stored message)" and argued for a more strategic tutor. The ATM architecture and the Ms. Lindquist tutor meet this criticism. The main difference between ATM and Traditional Model-Tracing is the incorporation of a tutorial model. Whereas traditional model-tracing tutors generate all their feedback from text templates that are inside the rules in the cognitive model, the ATM architecture generates a plan (usually involving multiple new questions to ask the student) for each error the student made. The model-tracing architecture does not have a way of encoding new general pedagogical knowledge, beyond that inherent in the architecture (such as giving feedback in response to errors). In summary, The ATM architecture allows Ms. Lindquist to combine the student modeling of traditional model-tracing tutors with a model of tutorial dialog based on an experienced human tutor including such features as positive and negative feedback, multiple tutorial strategies, with embedded sub-dialogs, as well as traditional buggy messages and hints.

ACKNOWLEGDEMENTS

This research was supported by NSF grant number 9720359 to CIRCLE and the Spencer Foundation.

REFERENCES

Aleven, V., and Koedinger, K. R., (2000a) The need for tutorial dialog to support self-explanation. In the Proceedings of the AAAI 2000 Fall Symposium, Building Dialog Systems for Tutorial Applications. Technical Report FS-00-01. AAAI Press. Menlo Park, CA.

Aleven, V., and Koedinger, K. R (2000b). Limitations of student control: Do students know when they need help? In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, ITS 2000, edited by G. Gauthier, C. Frasson, and K. VanLehn. Berlin: Springer Verlag.

Aleven, V.,Koedinger, K. R., & Cross, K. (1999). Tutoring Answer Explanation Fosters Learning with Understanding. In S. P. Lajoie & M. Vivet (Eds.), Artificial Intelligence in Education, Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration, proceedings of AIED-99 (pp. 199-206). Amsterdam: IOS Press.

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Boyle, D. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science*, 228, 456-462.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995) Cognitive tutors: lessons learned. *The Journal of the Learning Sciences*, 4 (2), 167-207.

Anderson, J. R. & Pelletier, R. (1991) A developmental system for model-tracing tutors. In Lawrence Birnbaum (Eds.) *The International Conference on the Learning Sciences.* Association for the Advancement of Computing in Education. Charlottesville, Virginia (pp. 1-8).

Arroyo, I., Beck, J., Woolf, B.,. Beal, C., & Schultz, K. (2000) Macroadapting Animalwatch to gender and cognitive differences with respect to hint interactivity and symbolism. *Proceedings of the Fifth International Conference on Intelligent Tutoring Systems*. Edited by G. Gauthier, C. Frasson, and K. VanLehn. Berlin: Springer Verlag.

Arroyo, I., Beck, J., Beal, C., Wing, R. & Woolf, B. (2001) Analyzing students' response to help provision in an elementary mathematics Intelligent Tutoring System . Help Provision and Help Seeking in

Interactive Learning Environments. Workshop at the Tenth International Conference on Artificial Intelligence in Education. San Antonio, TX. May 2001.

Baker, M. (1994). A model for negotiation in teaching-learning dialogues. *International Journal of Artificial Intelligent in Education.* 5(2), 199-254.

Bloom, B. S., (1984) The 2-sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher* June , (pp. 4-16).

Carbonell, J. R. (1970) AI in CAI: Artificial-intelligence approach to computer assistance instruction. *IEEE Transactions on Man-Machine Systems 11(4)*:190-202

Catrambone, R. (1996). Transferring and Modifying Terms in Equations. In Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society Hillsdale, NJ: Erlbaum. (pp. 301-305).

Chi., M. T. H. (1996) "Constructing Self-Explanations and Scaffolded Explanations in Tutoring" Applied Cognitive Psychology, Vol 10,S33-S49.

Cho, B., Michael, J., Rovick, A., Evens, M. (2000) The analysis of multiple tutoring protocols. Appeared in Intelligent Tutoring Systems: 5th International Conference (Eds Gauthier, Frasson VanLehn) Springer, Lecture Notes in Computer Science no. 1839, pp. 212-221.

Clancey, W. J., (1982) Tutoring rules for guiding a case method dialog. In D. Sleeman & J. S. Brown (Eds.) *IntelligentTutoring Systems* London: Academic Press. (pp. 201-226.)

Conati, C., Larkin, J. and VanLehn, K. (1997) A computer framework to support self-explanation. In : du Bolay, B. and Mizoguchi, R.(Eds.) Proceedings of AI-ED 97 World Conference on Artificial Intelligence in Education. Vol.39, pp. 279-276, Amsterdam: IO Press.

Corbett, A. T., and Anderson, J. R., (1995) Knowledge decomposition and subgoal reification in the ACT programming tutor. in *Proceedings of Artificial Intelligence in Education* (pp. 469-476)

Corbett, A.T., Koedinger, K.R., & Hadley, W.H. (2001) Cognitive Tutors: From the research classroom to all classrooms. In Goodman, P.S. (Ed.) *Technology Enhanced Learning: Opportunities for Change*. Mahwah, NJ: Lawrence Erlbaum Associates.

Corbett, A. T., McLaughlin, M., Scarpinatto, C., & Hadley, W. (2000) Analyzing and generating mathematical models: an algebra II cognitive tutor design study. In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, ITS 2000, edited by G. Gauthier, C. Frasson, and K. VanLehn. Berlin: SpringerVerlag. Lecture Notes in Computer Science no. 1839.

Core, M. G., Moore, J. D., and Zinn, C.. (2000) Supporting constructive learning with a feedback planner. In the *Proceedings of the AAAI 2000 Fall Symposium, Building Dialog Systems for Tutorial Applications*. Technical Report FS-00-01. AAAI Press.Menlo Park, CA.

Croteau, E., Heffernan, N. T. & Koedinger, K. R. (2004) Why Are Algebra Word Problems Difficult? Using Tutorial Log Files and the Power Law of Learning to Select the Best Fitting Cognitive Model. Proceedings of 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil.

Freedman, R. (2000) Using a reactive planner as the basis for a dialogue agent. In *Proceedings of the Thirteenth Florida Artificial Intelligence Research Symposium* (FLAIRS '00), Orlando.

Freedman, R. & Evens, M. W. (1996) Generating and revising hierarchical multi-turn text plans in an ITS. In C. Frasson, G. Gautheir and A. Lesgold (Eds.), *Intelligent Tutoring Systems: Proceedings of the 1996 Conference* (pp. 632-640). Berlin: Springer.

Freyberger, J., Heffernan, N., & Ruiz, C. (2004). Using Association Rules to Guide a Search for Best Fitting Transfer Models of Student Learning. In Beck, Baker, Corbett, Kay, Litman, Mitrovic & Rigger (Eds.) *Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes.* Held at the 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil. Lecture Notes in Computer Science. ISBN 978-3-540-22948-3.

Gluck, K. (1999). Eye movements and algebra tutoring. Doctoral dissertation. Psychology Department, Carnegie Melon University.

Graesser, A.C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R., & the TRG (1999). AutoTutor: A simulation of a human tutor. *Journal of Cognitive Systems Research*, 1, 35-51.

Grosz, B. J. & Sidner, C. L. (1986) Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175-204.

Heffernan, N. T. (2003). Web-Based Evaluations Showing both Cognitive and Motivational Benefits of the Ms. Lindquist Tutor In F. Verdejo and U. Hoppe (Eds) *11th International Conference Artificial Intelligence in Education*. Sydney, Australia. IOS Press. pp. s 115-122.

Heffernan, N. T. (2001). *Intelligent Tutoring Systems have Forgotten the Tutor: Adding a Cognitive Model of an Experienced Human Tutor*. Dissertation & Technical Report. Carnegie Mellon University, Computer Science, http://www.algebratutor.org/pubs.html.

Heffernan, N. T. & Croteau, E. (2004). Web-Based Evaluations Showing Differential Learning for Tutorial Strategies Employed by the Ms. Lindquist Tutor. In James C. Lester, Rosa Maria Vicari, Fábio Paraguaçu (Eds.) *Proceedings of 7th Annual Intelligent Tutoring Systems Conference,* Maceio, Brazil. Springer Lecture Notes in Computer Science. pp. 491-500.

Heffernan, N. T., & Koedinger, K. R. (1997) The composition effect in symbolizing: the role of symbol production versus text comprehension. *Proceeding of the Nineteenth Annual Conference of the Cognitive Science Society,* 307-312. Hillsdale, NJ: Erlbaum.

Heffernan, N. T., & Koedinger, K. R. (1998) A developmental model for algebra symbolization: The results of a difficulty factors assessment. *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society,* 484-489. Hillsdale, NJ: Erlbaum.

Kautz, H. and Allen, J.F. (1986) Generalized plan recognition. In *Proceedings of AAAI National Conference on Artificial Intelligence,* Philadelphia, PA.

Koedinger, K. R., Anderson, J.R., Hadley, W.H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8,* 30-43.

Koedinger, K. R., & Anderson, J. R. (1998). Illustrating principled design: The early evolution of a cognitive tutor for algebra symbolization. In *Interactive Learning Environments*, 5, 161-180.

Kulik, J.A. 1994. "Meta-Analytic Studies of Findings on Computer-Based Instruction."
In Eva L. Baker and Harold F. O'Neil, Jr., eds., *Technology Assessment in Education and Training,* pp. 9-33. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kulik, J. A., Bangert, R. L., & Williams, G. W. (1983) Effects of computer-based teaching on secondary school students. *Journal of Educational Psychology, 75*, 19-26.

McArthur, D., Stasz, C., & Zmuidzinas, M. (1990) Tutoring techniques in algebra. *Cognition and Instruction*. 7 (pp. 197-244.)

Mendicino, M. & Heffernan, N. T. (Submitted) Comparing the learning from intelligent tutoring systems, non-intelligent computer- based versions, and traditional classroom instruction. *Journal of Interactive Learning Research (JILR). http://nth.wpi.edu/pubs_and_grants/papers/journals/MendicinoSubmitted.doc*

Moore, J. D. (1996) Discourse generation for instructional applications: Making computer-based tutors more like humans. *Journal of Artificial Intelligence in Education*, 7(2), 118-124

Mwangi, W. & Sweller, J. (1998). Learning to solve compare word problems: The effect of example format and generating self-explanations. *Cognition & Instruction 16*(2): 173-199.

Murray, T. (1999). Authoring intelligent tutoring systems: an analysis of the state of the art, International Journal of Artificial Intelligence in Education, 10, 98-129.

Nathan, M. J. & Koedinger, K. R. (2000). An investigation of teachers' beliefs of students' algebra development. *Cognition & Instruction 18*(2): 209-237.

Ohlsson, S. (1986) Some principles for intelligent tutoring. *Instructional Science, 17,* 281-307.

Razzaq, L. & Heffernan, N. T (2004) Tutorial dialog in an equation solving intelligent tutoring system. In J.C. Lester, R.M. Vicari, & F. Parguacu (Eds.) Proceedings of 7th Annual Intelligent Tutoring Systems Conference, Maceio, Brazil. 851-853.

Renkl, A. (1997) Learning from worked examples: A study of individual differences. *Cognitive Science, 21(1),* 1-30.

Rickel, J., Ganeshan, R., Lesh, N., Rich, C. & Sidner, C. L. (2000). Task-oriented tutorial dialogue: issues and agents.AAAI Fall Symposoum on Building Dialogue Systems for Tutorial Applications, Cape Cod, MA, AAAI Press.

Schoenfeld, A., Gamoran, M., Kessel, C., Leonard, M., Or-Bach, R., & Arcavi, A. (1992) Toward a comprehensive model of human tutoring in complex subject matter domains. *Journal of Mathematical Behavior*, 11, 293-319

Shelby R, Schulze K, Treacy D, Wintersgill M, VanLehn K, & Weinstein A. (2001). An assessment of the Andes tutor. In the Proceedings of the Physics Education Research Conference, July 21-25, Rochester, NY.

Shulman, L. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher, 15*, 4-14.

Sweller, J. & Cooper, G.. (1985) Use of worked examples as a substitute for problem solving in algebra learning. *Cognition and Instruction*, 2, 58-89

VanLehn, K. (1998). Analogy events: How examples are used during problem solving. *Cognitive Science, 22* (3), 347388

VanLehn, K, Anderson, J., Ashley, K., Chi. M., Corbett, A., . Koedinger, K., Lesgold, A., Levin, L., Moore, M., and Pollack, M., (1998) NSF Grant 9720359. *CIRCLE: Center for Interdisciplinary Research on Constructive Learning Environments.* NSF Learning and Intelligent Systems Center. January, 1998 to January, 2003.

VanLehn, K. Freedman, R., Jordan, P., Murray, C., Osan, R., Ringenberg, M., Rose, C., Schulze, K., Shelby, R., Treacy, D., Weinstein, A. and Wintersgill, M. (2000) Fading and deepening: The next steps for Andes and other model-tracing tutors. In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems* , ITS 2000, edited by G. Gauthier, C. Frasson, and K. VanLehn. Berlin: Springer Verlag.

VanLehn, K., Siler, S., Murray, C, Yamauchi, T. & Baggett, W. B. (2003). Human tutoring: Why do only some events cause learning? *Cognition and Instruction.21(3)*, 209-249.

Zhu, X. & Simon, H. A. (1987). Learning mathematics from examples and by doing. *Cognition & Instruction* *4*(3): 137-166.