BUILDING A NEW STUDENT MODEL TO SUPPORT ADAPTIVE TUTORING IN A NATURAL LANGUAGE DIALOGUE SYSTEM

BY

YUJIAN ZHOU

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science
in the Graduate College of the
Illinois Institute of Technology

Approved_____
Advisor

Chicago, Illinois
May 2000

ACKNOWLEDGMENT

I would like to thank my advisor, Professor Martha W. Evens, who mentored and encouraged me to pursue my degree from the first day I met her.

I thank Dr. Joel Michael and Dr. Allen Rovick of Rush Medical College. Their insight and enthusiasm have made a huge impact on my thinking of tutoring systems.

I also want to thank my colleagues in Circsim-Tutor project. It was such a pleasure to working with them. Specifically, I would like to thank Michael Glass, Reva Freedman, Stefan Brandle, Jung Hee Kim, Feng-Jen Yang, Byung-In Cho, and Norman Jiang.

Finally thank you Wenhui. You are the one who is always there for me.

This work was supported by the Cognitive Science Program, Office of Naval Research under Grant No. N00014–94–1–0338, to the Illinois Institute of Technology. The content does not reflect the position or policy of the government and no official endorsement should be inferred.

Y.Z.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Term
IS	Inotropic State
HR	Heart Rate
TPR	Total Peripheral Resistance
SV	Stroke Volume
CO	Cardiac Output
CVP	Central Venous Pressure
MAP	Mean Arterial Pressure
RAP	Right Atrial Pressure

CHAPTER I

INTRODUCTION

This work attacks problems of student modeling and adaptive hinting in a natural language based Intelligent Tutoring System (ITS) — CIRCSIM-Tutor (CST). CST is an ITS that imitates the behavior of human tutors. So, we have also studied how to use machine learning techniques to discover tutoring rules from human tutoring transcripts.

1.1 Intelligent Tutoring System

Intelligent Tutoring Systems are educational software systems involving artificial intelligence. This field includes multidisciplinary research areas, such as artificial intelligence, education theory, psychology, cognitive science, and theories of human-computer interaction.

Intelligent computers are perfect for educational use since computers will never get tired, they are available all the time, and they are accessible by users all over the world (thanks to the rapid growth of the Internet). Specifically, they can be used one-on-one, and therefore they can adapt their tutoring to the needs of individual students. This will make an ITS a very good compliment to classroom teaching. One key component that makes an ITS more intelligent or more adaptive to the needs of individual students is that an ITS maintains a student model. This model is also the key difference between ITS and the traditional "Computer-Aided Instruction" Systems (CAI). Since a student model is so important to an ITS, this key component has been studied since the beginning of ITS research. It is a highly interesting area even to people outside the field because of its implications for user modeling for all types of computer systems, such as information

retrieval systems, adaptive multimedia systems, and so on. Several different types of models and theories about student models have been created in the field so far.

At the same time, many researchers also argue that the main purpose of a student model in the context of intelligent tutoring is to guide pedagogical decision-making. There is little need for a description of the student's knowledge unless there is some way for the system to make use of, or react to, that description [Ohlsson, 1986] [Woolf and Murray, 1994]. This functional view of the student model is very important when we try to build real systems. So it is very desirable that the study of student modeling should consider the pedagogical model. In particular, in an ITS where the tutor keeps control, the pedagogical model and the student model are very tightly related.

One important tutoring tactic in one-on-one dialogue is hinting. The contents of a hint are very sensitive to the student's response and the dialogue context. In order to build a sophisticated computational model of hinting that is pedagogically useful and context sensitive, we need a rich student model as well as rules that specify how to plan a hint in different situations. The combination of transcript analysis and machine learning gives the opportunity to discover the rules governing the behavior of human tutors from transcripts of real tutoring sessions.

The Internet is the most efficient medium to reach more and more students. It offers great opportunities and challenges to the ITS community. Benefits of Web-based tutoring systems are clear: classroom independence, platform independence, and easy deployment. But the World Wide Web also demands more adaptation from the ITS systems. As Brusilovsky [1998] pointed out, Web-based ITS are intended to be used by a much wider variety of students than any standalone systems and many students may be working "alone" with web-based ITSs. So, to improve tutoring quality, web-based ITSs need to further improve their student model and the adaptability of their tutoring. At the

same time, the Internet also offers the opportunity for ITSs to communicate with each other.

Researchers in the ITS area have already explored some of the opportunities to use the Web as the main medium to deliver tutoring systems [Brusilovsky, 1998, 1999].

1.2 Student Modeling and Adaptive Tutoring in Circsim-Tutor

CIRCSIM-Tutor (CST) is a language-based intelligent tutoring system designed to help first-year medical students learn to solve problems involving the reflex control of blood pressure. It is a one-on-one tutoring system, which attempts to adapt its tutoring to the student's learning state. There are several facts about CIRCSIM-Tutor that makes it special among other ITSs:

- The system is designed to imitate human tutors.
- The tutoring uses natural language and the tutoring dialogue is plan-based.
- The machine tutor takes the lead most of the time.
- The tutoring dialogue is handled as free text input and output.

So instead of only selecting available tutoring material, CST needs to interactively plan a conversation with the student. This ability to plan a dialogue influences the information needed for tutoring decisions and constrains the possible information that the tutor can collect to build a student model. Especially, multiple planners will demand different information from the student model.

A further step toward building a web-based ITS is to consider CST as an agent, which can communicate with other ITSs. Especially, since CST's language understanding ability limits the information that CST can collect from the student, it would be a very good complement to CST if it could communicate with an ITS, such as a Graphic User Interface-based ITS, which can collect more detailed and accurate student information. So, research on a centralized student database or a communicable distributed student model will be useful for the CST project.

1.3 Statement of the Problem

From this general introduction to ITS and our brief discussion of CST, we can see that it is very important for an ITS to construct a student model that explicitly represents the properties of a particular student. And it is also very important that an ITS can find a way to use this model, i.e., the ITS can adapt many aspects of its tutoring to each individual student by consulting this model. In this section, I explain in detail why we need to study student modeling in CST and why should we study how it coordinates with the adaptive tutoring mechanism.

First, a student model is particularly important to Circsim-Tutor. Hume [1995] showed that decisions on when and how to give a hint must be based on the student model. Brandle's [1998] study of acknowledgments demonstrated that issuing an acknowledgment is related to the system's assessment of the student. Khuwaja [1994] included student performance as one of the prerequisites to select problems in his curriculum planning rules. Freedman [1996a] described the student model as one of the important constraints for the tutorial planner.

A student model is even more important for Circsim-Tutor (CST) than other ITS, because CST is a natural language based dialogue system designed to simulate the behavior of human tutors. There are many aspects of the tutorial dialogue that the machine tutor needs to tailor to the individual student. This tailoring process depends on the information from the student model and other modules. The tutor needs to make decisions about a variety of aspects of tutoring, including tutoring strategy, style, content, and even the use of language, each of which may depend on different information about the student.

To construct a student model is a very difficult problem in CST too. Theoretically it is impossible to know what is a student's real mental state. Practically, the procedure to infer a student model may include too much uncertainty. So, how to build a student model that includes enough information to support the variety of decisions involved in a tutoring dialogue and at the same time responds fast (so it can support real time conversation) is a complex problem.

It is also important that we study the student model within the framework of the whole ITS, understanding where its inputs come from, and where its outputs go, because the intelligence of an ITS is distributed among all modules and the student model is only one of them. Specifically, what kind of intelligence we can put into a student model depends on other modules and this intelligence also influences other modules. Especially we need to study what other modules expect from the student model, i.e., how the information in the student model will be used by other modules. In Circsim-Tutor we hope the information in the student model can support a variety of decisions in each step of construction of the tutorial dialogue.

Circsim-Tutor Version 2 was developed by Woo and others in 1991 in Lisp [Woo, 1991]. Compared to our current needs, the original student model in CST Version 2 is too

simple. It cannot support new features of the tutorial dialogue and cannot be used in Circsim-Version 3, which will have a more sophisticated planner to plan multi-turn dialogue. So, it is necessary to build a sophisticated student model to update Circsim-Tutor Version 2 and study how to use it to support more adaptive tutoring in CST.

It is also important to study further how to use a student model to support different aspects of tutoring. For example, how are the contents of hints related to the student model; how can we choose different retry strategies according to the student model; and so on. In particular, CST is a system that imitates the behavior of human tutors. So, how to study their tutoring behavior and how to build computational models of this behavior are highly important. This research will use transcript analysis and machine learning to discover tutoring rules from human tutoring transcripts.

Research on building a student model is of use to other projects as well as ours. Since any interactive dialogue system needs a user model, the study of how to build a sophisticated student model in a dialogue-based tutoring system will benefit other interactive dialogue systems too.

To achieve the goal of building a sophisticated student model to support intelligent tutoring, I have used several different approaches.

First, I use the original Circsim-Tutor Version 2 as a test bed. I find a problem in version 2, then propose a change to fix the problem, then implement the change, then find other problems and begin another testing cycle.

Second, I analyze human tutoring transcripts to find clues to possible solutions for the problems I find in the testing cycle. It is a tradition in the Circsim-Tutor project to analyze transcripts in order to discover the necessary tutoring knowledge. For me, it is important to know what kind of student model is used by our human tutors to support the variety of their tutoring behavior. So that we can build a computational model to simulate

their model. I want to know especially, what kind of information in the student's answer they should focus on. So my goal at this step is trying to find the possible solutions for the problems I discovered in the first step.

Furthermore, I apply Machine Learning techniques to find tutoring rules in human tutoring transcripts, since manual analyses can handle only a small portion of the transcripts.

Then I apply all the results from transcript analysis and from machine learning to improve Circsim-Tutor in the following areas: a new student model, an adaptive hinting mechanism, the retry strategies, and other planning improvements. This new version has been tested in three large-scale experiments with medical students.

Finally I built a student modeler for Circsim-Tutor Version 3 by combining results from the previous steps. This modeler actually includes four different models in order to support different levels of the tutoring process and at the same time respond fast. I have evaluated part of my approach by using simulation since version 3 is still under development.

1.4 The Goals of This Research

In this work I tried to:

1) Identify the problems in the student model in the original version and figure out where we could improve it.
2) Find out how human tutors' decisions are related to the student's learning state and what kind of information is necessary for the decisions of the planners.

3) Use Machine Learning techniques to automatically obtain the possible rules.
4) Implement these rules as much as possible in CIRCSIM-Tutor.
5) Use results from these studies to update Version 2, including a new student model, an adaptive hinting mechanism, the retry strategies, and other planning improvements.
6) Test the updated version with medical students.
7) Build a student model for Version 3.
8) Study the implications of distributed ITS.

1.5 Organization of This Thesis

In Chapter II, I introduce the Circsim-Tutor project, describe earlier work on the student model and mention the possible uses of the student model. Chapter III discusses related issues in designing a student model and analyzes the student model components and their usage in several successful Intelligent Tutoring Systems. In Chapter IV I present how to extract a possible student model from analysis of human tutoring transcripts. I show a detailed analysis based on a specific tutoring model. Chapter V applies Machine Learning techniques to automatically find tutoring rules and possible student modeling information from transcripts. Chapter VI discusses how to apply the results from Chapter IV and V in real systems – the updated Circsim-Tutor Version 2, especially a new hinting mechanism for tutoring dialogue. In Chapter VII I describe the student model framework for Circsim-Tutor Version 3. In Chapter VIII, I extend my study to distributed tutoring systems and discuss the advantages of a Web-based multi-tier architecture.

The last chapter summarizes this thesis and discusses the significance of this research.

CHAPTER II

THE CIRCSIM-TUTOR PROJECT

2.1 Introduction to Circsim-Tutor

CIRCSIM-Tutor (CST) is a language-based intelligent tutoring system designed to help first-year medical students learn about the reflex control of blood pressure. It is a one-on-one tutoring systemthat attempts to adapt its tutoring to the student's learning state.

When students use CST, they are asked to predict the value of seven parameters at three points in time: the DR or direct response stage, the RR or reflex response stage, and the SS or new steady state stage. After the student finishes each prediction phase, the tutor and the student conduct a dialogue. In this tutoring dialogue stage the tutor develops its tutoring goals based on information collected from the prediction table and implements its goals by selecting different tutoring methods. Each tutoring method is executed as a series of tutoring tasks, and each task is implemented using questions, hints, explanations or other strategies.

The main user interface screen is illustrated in Figure 1.

There are roughly two steps in building this system: 1) characterizing human tutoring strategy and tutoring rules; 2) implementing those rules as much as feasible in the computer tutor.

The goal of the system is to remedy the student's misconceptions about the cardiovascular system and to help the student learn problem solving skills. In this system we assume that the student has already attended the regular lectures and knows most of the necessary facts.

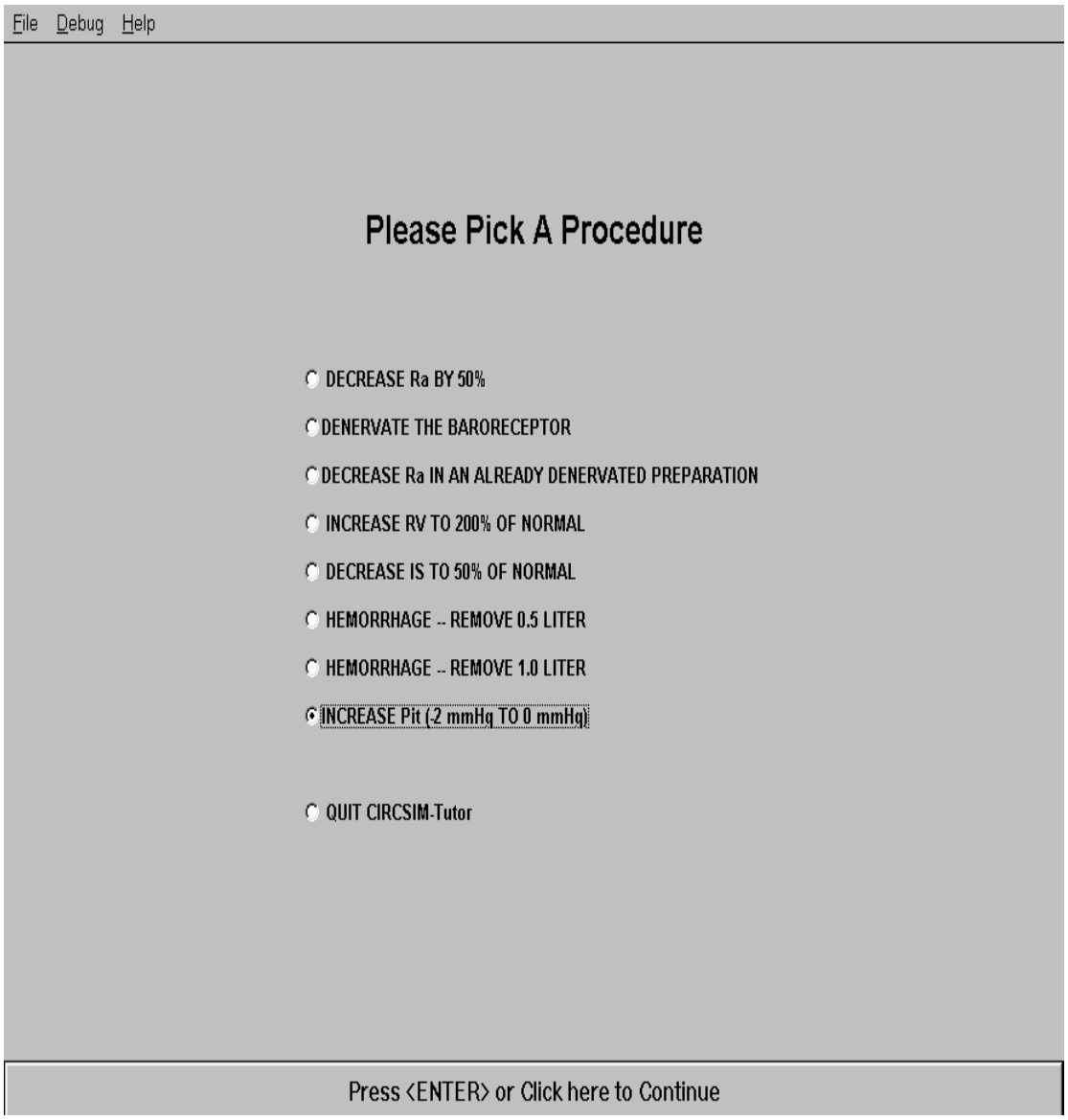

Figure 1 The Procedure Selection Window

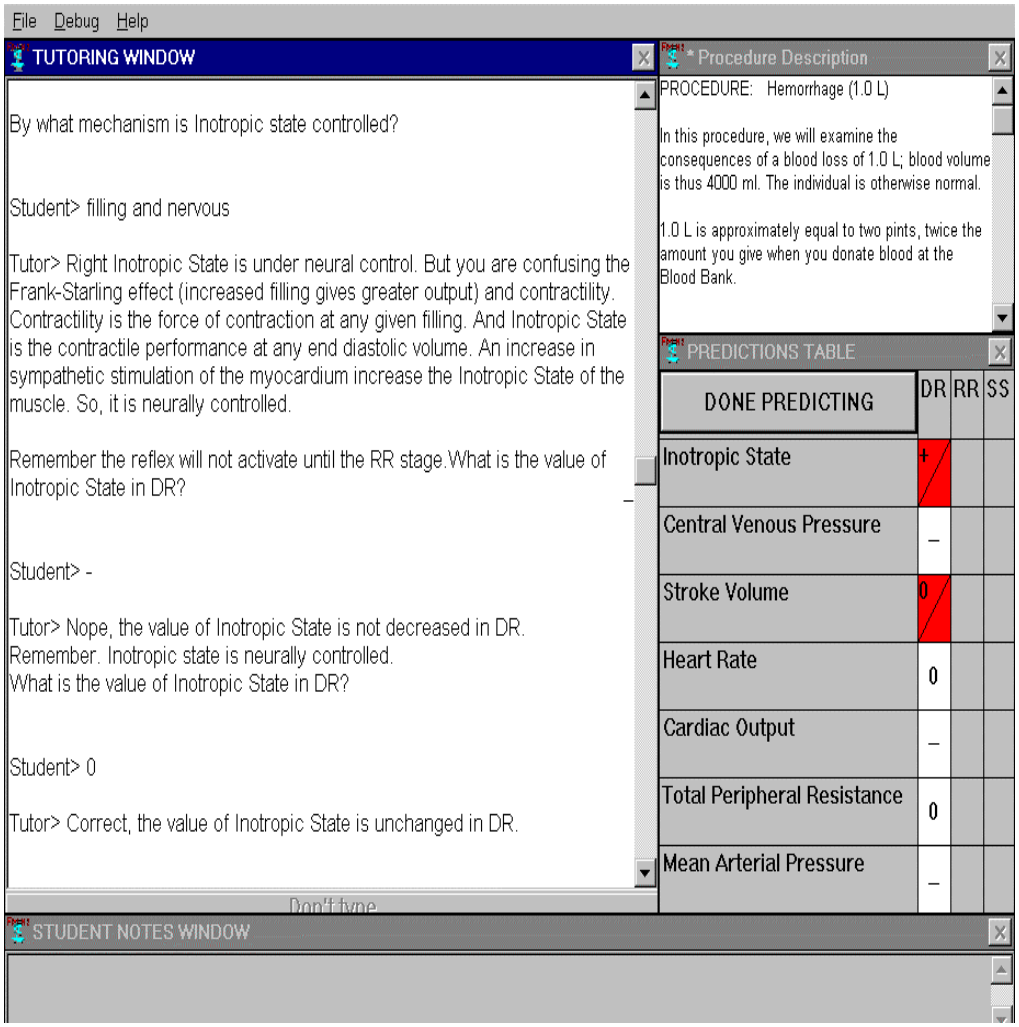

Figure 2. The interface of Circsim-Tutor Version 2.

A prototype of Circsim-Tutor was implemented in Prolog by Kim [1989].

The current working version is an extension of Circsim-Tutor Version 2, developed by Woo and others in 1991 in Lisp [Woo, 1991]. We keep improving this version so it can be used by the students at Rush Medical College. Glass has written a new input understander, which is more robust and can handle many more phenomena [Glass, 1997]. He also made some other improvements for the instructional planner. Brandle built a new screen manager, so the current version can be run on both PCs and Macintoshes [Brandle, 1998]. I have updated the lesson planner and discourse planner, added a dynamic student model during the tutoring dialogue, and added new hinting mechanisms to deliver more meaningful and context sensitive hints. This updated version has been used by Rush students in April and November 1998, and in November 1999.

Circsim-Tutor Version 2 has seven modules: the instructional planner, the student modeler, the input understander, the text generator, the screen manager, the problem solver, and the knowledge base [Woo, 1991]. Most intelligent tutoring systems have four major components: an instructional modeler, a student modeler, a user interface, and a domain knowledge base [Sleeman and Brown, 1982]. In order to carry on a natural language dialogue the user interface in our system has been divided into three modules: the input understander, the text generator, and the screen manager.

The function of each component is described below.

Instructional planner: responsible for deciding what to do next at each point during a tutoring session. It interacts with all other modules in order to carry out tutorial activities. It includes the lesson planner and the discourse planner.

Student modeler: responsible for inferring and representing the student's current knowledge state and misconceptions. This model is the key that the tutor can give individualized instruction to the student.

Input understander: responsible for understanding the student's natural language inputs. It is a very important component in giving the tutoring system a good natural language interface.

Text generator: responsible for executing the tutor's logic output into a natural language sentence. It is another important component for the tutoring system to have a natural language interface.

Screen manager: responsible for general interaction between the student and the system.

Problem solver: responsible for solving the problem presented to the student or questions asked by the student. It includes two modules: the main problem solver and the sub-problem solver.

Knowledge base: responsible for storing the expert's knowledge in order to solve a problem and to give causal explanations.

Circsim-Tutor Version 3 is now under construction. It has a new planner -- ATLAS, a new text generator, and some other new features. The new version will plan at four levels:

Curriculum Planning

Discourse Planning with Multiturn Dialogue

Turn Planning

Sentence Planning - surface generation - lexical choice

ATLAS is a reactive planner for tutorial planning and discourse planning developed by Freedman [1999] as a part of her research at the University of Pittsburgh. ATLAS selects and executes operators that contain goals for each task the planner is trying to accomplish. Each operator contains goals, as well as preconditions that must be satisfied before the operation can be performed, a set of steps for the operation. The basic knowledge representation for the discourse planner is a sophisticated form of schema that allows static and dynamic preconditions, recursion and full unification. The most common form of schema encompasses three levels:

Tutoring strategy

Topics within the strategy

Text generation primitives (inform or elicit) for each topic

If the student answers a question incorrectly, ATLAS allows the system to backtrack at each of these levels.

My contribution to Version 3 is a practical student model and some specific heuristic rules that will use the student model to support the planning implemented as ATLAS planning operators.

2.2 Keyboard to Keyboard Tutoring Experiments

CST is designed to imitate the behavior of human tutors. In order to discover the rules used in human tutoring, we carried out human tutoring sessions keyboard to keyboard to collect transcripts. The original Computer Dialogue System only captured keyboard to keyboard text [Li et al., 1992]. I have developed a new system that provides the same prediction table as CST, along with its text dialogue window. So the tutoring screen in the keyboard to keyboard experiments is similar to CST. Joel Michael and Allen Rovick of Rush Medical College tutored fifty students using the original version; they have now used the new version to tutor another twenty five. Each of their conversations was recorded as a text file along with the student's predictions. We call the text files our tutoring transcripts and use them to analyze human tutoring behavior.

2.3 Student Modeling in Circsim-Tutor

The student model is an important module of Circsim-Tutor. It stores information about the student's current knowledge state and serves as one of the inputs that the tutoring planner uses to determine the next tutoring step. Though it is quite simple in the early versions it has been studied from the very beginning in this project.

2.3.1 Student Model in Version 2. The student model in Version 2 is an explicit overlay model of the seven core variables and their relationships plus an implicit overlay model of other domain knowledge. It does not include any overall student assessment and bug model, and it does not update the explicit overlay model during the dialogue.

Since Version 2 only includes a static and simple explicit overlay model of the seven core variables, its planner cannot dynamically update its original plan according to the student's state. So the ability to provide adaptive tutoring during the tutoring dialogue in Version 2 is very limited.

As Khuwaja pointed out, Version 2 does not put emphasis on the misconceptions of the student, and the major goal of the tutor is to find out the missing chunks of knowledge that are responsible for the student's sub-optimal behavior and develop lessons so that the student fills in the gaps in his or her knowledge [Khuwaja, 1994].

2.3.2 Shim's Student Model. Shim proposed a student modeling framework for Circsim-Tutor Version 2. His framework combines the overlay model and the bug library model [Shim, 1991]. He argued that to model a student better we should take into account earlier responses as well as immediate ones [Shim et al., 1991]. Unfortunately, only the top level overlay model was implemented in Version 2.

His overlay model consists of frames for every parameter and every causal relationship in the concept map. To overcome possible inconsistencies each slot shows three pieces of information about the student's cognitive status: the history string of "c" and "w" for correct and wrong responses, the confidence factor calculated from this string, and the point at which the response is made. His bug library consists of a frame for each known bug. Similar kinds of bugs are grouped together.

2.3.3 Hume's Student Model. Hume created a student model for Version 3 before the planner had been built [Hume, 1995]. In his model he used surface errors in the prediction table to identify the error patterns, and made a list of error patterns associated with each concept. To each error pattern he attached two slots:

1. Prediction history
2. Tactical history

His tactical history is based on a string of Hs (for hint) and Es (for explanation) while our current research is constructing tutoring goals, methods and topics in a different kind of hierarchy [Kim et al., 1998]. So the process of implementing the tactical history must be different.

Hume also used the tactical history to reveal sensitized error patterns and proposed a list of sensitized classes of these patterns.

His idea of global and local assessment has been implemented but using a different method. It has been broken into specific performance measures since we now have changed the categories of student answers and the tutoring goal hierarchy.

2.3.4 Using Student Assessment to Decide When and How to Give Hints. Hume's study showed that when and how to give a hint is based on the student model [Hume, 1995]. He also suggested that tutorial tactics are dependent on the student's response [Hume et al., 1996a]. He has several rules like:

If an error is made, provide a hint. The exception is when the global assessment is very low. In other words,

If
(a) the student has not discovered the concept and
(b) the global assessment continues to be sufficiently high and
(c) the number of hints in a string is sufficiently low,
try subsequent hints.

2.3.5 Issuing Acknowledgments Based on the Student Assessment. Brandle's [1998] study of acknowledgment demonstrates that issuing an acknowledgment is related to the student's assessment. His theoretical approach is based on Clark's [1996] theory of joint actions.

He used Machine Learning methods to produce several rules for how to issue an acknowledgment. His results showed that issuing acknowledgments is mostly dependent on the student's answer category and the local assessment of the student's performance.

2.3.6 Using the Student Model to Support Curriculum Planning. Khuwaja included the student's performance as one of the prerequisites to select problems in his curriculum planning rules [Khuwaja, 1994]. Here is an example of his rules:

If
a student's performance is not good in a procedure from category "A"
Then
make category "A" an option for student, after PSR1 is satisfied.

We can see that basically he needed the student's global performance to decide what should be the next procedure for the student.

Cho is working on the curriculum plan now and he is trying to use both the local and the global performance assessment to implement the curriculum planner.

2.3.7 Using the Student Model to Support Adaptive Tutoring. Different researchers in the Circsim-Tutor group have discovered different uses of the student model from their own area as described in the above sections. Basically the role of the student model is to support any tutoring decisions, from the high level curriculum planning decisions, to the lowest level of detailed response strategy decisions, and even the terminology choices of the surface language generation. But for decisions at different levels made by different components, the expected information about the student may be different. So I have included several kinds of student information in different student models to support different needs.

CHAPTER III

STUDENT MODELING: THEORIES AND PRACTICAL PROBLEMS

3.1 Overview and Theories

A student modeling component is an essential part of an ITS. There are lots of models and theories about student models in the field so far. Many researchers have tried to classify them and formalize them in a unified framework. For example, VanLehn [1988] used three dimensions -- bandwidth, target knowledge type, and differences between the student and the expert – to construct the space of student models and classify them within this space. Ragnemalm [1996] regards the student modeling problem as the process of bridging the gap between the student's input to the tutoring system, and the system's conception and representation of correct knowledge. Self [1990b] tried to provide a theoretical, computational basis for student modeling, which is psychologically neutral and independent of applications.

In the Circsim-Tutor project we are more interested in a view that relates to issues of designing an ITS. So the classification given by Ohlsson [1986], which emphasizes the construction of tutoring systems, is presented here. In his paper, he calls the student modeling problem "cognitive diagnosis." To analyze the various types of diagnosis he proposed three criteria:

1. the kinds of empirical observations they operate on and the kinds of knowledge structures they need to carry out the diagnosis (i.e. their "inputs"),
2. the procedures by which they infer the cognitive state of the student, and

3. the types of descriptions they generate (i.e. their “outputs”).

Of these criteria, the type of description generated by the diagnostic system has been considered the most fundamental, and therefore he classified student models into four types: performance measures, overlays, bug library, and simulations. Later he proposed a new approach, constraint based modeling, which can produce a more general and flexible type of description [Ohlsson, 1992].

3.1.1 Performance Measures. Performance measures are a simple way to describe a student. It is computationally simple to measure the student’s answers and carry out some statistical aggregation procedure. It is a global representation of the student; it can only support global actions on the part of the tutor. For example, upgrading or downgrading the difficulty of practice items. But it does not provide the level of detail necessary to decide what this student needs right now in order to learn a particular concept, procedure, fact or principle.

3.1.2 Overlay Model. Overlay models are designed to represent a student by the set of subject-matter units (parts of the expert knowledge) he has mastered. This model assumes that a student is a subset of an expert. So the basic function of this model is to find the missing chunks of the expert knowledge and the instructional planner uses this information to decide what topic to teach so the student can fill up the missing chunks.

3.1.3 Bug Library. In the overlay model, the student is seen as knowing the same things as an expert, only fewer of them. This perspective ignores the possibility of misunderstanding or other false knowledge. For example, students may have erroneous

procedures, false principles, and incorrect facts. The bug library approach is try to collect all that false knowledge together and try to match the student's misbehavior to that false knowledge.

The pedagogical promise of a bug library is that the tutor can evoke remedial instruction, while in the overlay model approach the tutor can only provide gap filling of the missing chunks. But how to construct the remedial instruction is not trivial and there are no general solutions so far.

The difficulty in constructing a sufficient bug library is that it is a labor-intensive empirical undertaking. It is also not very flexible and cannot identify new errors from the student. Also the inference from an erroneous answer or solution to what the student knows is more complicated than the inference from a correct answer to what he or she knows correctly.

3.1.4 Simulations. Simulations can be used to model the student by performing the likely behavior in the relevant knowledge domain. A simulation is runnable and can predict what the simulated person would do if he solves the same task. So the system can know every step that the student has taken to solve the problem. This approach assumes that we can specify the student's mental representation of both declarative and procedural knowledge.

Such a model has a quite complete description of the student, which should provide dramatic advantages in instruction. But again, even if you know the student's solutions it is not trivial to design a plan to help the student solve the problem efficiently.

Model-tracing can be considered as a simple simulation model. It can only model one step of the student's solution. As soon as the student goes wrong, the tutor will

intervene and bring the student back to the correct path. So it actually simulates one step at a time. If it includes mal-rules along with correct rules, it can match misbehavior too.

3.1.5 Constraint-Based Modeling. Later Ohlsson proposed a new approach to the student modeling problem by representing subject matter knowledge as sets of constraints [Ohlsson, 1992]. Constraint violation on the part of the student indicates incomplete or incorrect knowledge and can therefore be used to guide the instructions of an ITS. It is neutral with respect to pedagogy. It only provides a description of the student in terms of the constraints which he or she has violated, but it leaves open the question of what instruction is implied by that description. But no real tutoring system has actually implemented this model.

3.2 Practical Design Issues

Although new theories and new models continually appear in this field, there are doubts about the value of actually constructing them. For example, Sandberg [1987] summarized a general opinion that it is debatable whether the cost of constructing very detailed, complex user models is worthwhile in terms of the gain in teaching efficiency.

Sleeman and the others in the PIXIE group also studied whether a sophisticated student model is necessary for tutoring [Sleeman et al., 1991]. They conducted experiments to compare model based tutoring based on an accurate model of the student's performance in the domain and simply reteaching. Their results showed that a detailed discussion of a pure guess might be counterproductive, as it might help "cement" the incorrect form. So both questions, whether an accurate student model produces more effective tutoring and how to use this kind of model, still need more study.

Many researchers have argued that we should not totally abandon a model of the student but we should make the student modeling problem more practical and useful, i.e., a truly effective component of an ITS that can be used to support effective tutoring.

Self argued that there are two sources that may be used to drive different opinions about the student model [Self, 1990a]:

1. Preconceptions about the potential roles of student models.
2. Theoretical and practical difficulties in building and using student models.

He then offered more productive views of the potential roles of student models and described some more realistic, practically achievable, and useful goals for student modeling -- four slogans to bypass the intractable problem of student modeling:

1. Avoid guessing – get the student to tell you what you need to know.
2. Don’t diagnose what you can’t treat.
3. Empathize with the student’s beliefs, don’t label them as bugs.
4. Don’t feign omniscience – adopt a “fallible collaborator” role.

It is very important to study the practical roles of student models and the practical issues in building and using student models in the ITS field, since the student modeling problem is also studied in cognitive science (in the ITS field the student modeling problem is also called cognitive diagnosis sometimes), and psychology. But the goal may be very different from the view of ITS design although we can learn a lot from studies in other fields. For example, in cognitive science and psychology, it is very important to

know the student's knowledge status exactly, since they are directly concerned with the student's learning behavior. But in the ITS field (at least in the Circsim-Tutor project), what we directly study is our tutor's behavior (the student model is a model from the tutor's point of view), whether the student's information is important or not, or how detailed it should be, depends on whether it makes a difference to the tutoring model or not. So, what kind of student model we should build for our ITS is totally dependent on the tutoring model. Of course, it is possible that after some study of the student model, we may find a new tutoring method to use this model and further update the tutoring model. So it may be helpful to study both models together.

Ohlsson also argued that the main purpose of student modeling in the context of intelligent tutoring is to guide pedagogical decision making and there is little need for a description of the student's knowledge unless there is some way for a tutoring system to make use of, or react to, that description [Ohlsson, 1986].

Woolf, from her own system design experience, also provided practical suggestions [Woolf and Murray, 1994]. She argued that ITS designers should carefully consider both the intended uses of a student model and the limitations and tradeoffs inherent in the technology at their proposal. A designer should not attempt to model too much or too deeply, or to build complex mechanisms that infer information about the student's cognitive state that (page 133)

1. will not actually be used by the end system, or
2. will be too uncertain or abstract to be effectively used.

She then suggested that in some ITSs, it is not necessary to build a cognitively accurate model of the student's mental states of thoughts. The student model should be only complex enough to meet the needs foreseen.

3.3 Some Interesting Examples

3.3.1 Student Modeling in LISP-Tutor. The LISP tutor was developed at Carnegie-Mellon University [Corbett et al., 1990] [Corbett & Anderson, 1992]. This tutor helps students as they write short computer programs. It presents exercises requiring students to write short programs and monitors the students' performance symbol by symbol. So it is basically a learning by doing program. It informs the student of errors and provides advice upon request. The tutor is constructed around a set of several hundred programming rules that allows the program to solve exercises step-by-step along with the student.

The student model is implemented in the form of a production system. The complete set of correct rules for writing code is referred to as the *ideal student model* and represents the instructional objectives of the text and the tutor. The student model also includes a *bug catalog* – a set of incorrect rules that reflect known misconceptions. Each rule is related to a probability parameter. An example of the rules:

IF the goal is to define a function called *name*
that accepts *n* parameters and performs the task *process*
    Then code a call to *defun*
    and set subgoals to code
        the function name *name*
        a list of *n* parameters
        the process *process*

Each time the student has the opportunity to apply a rule in the ideal student model, the tutor updates a probability estimate that the student has learned the rule, contingent on the accuracy of the student's response, they call this student modeling process *knowledge tracing*.

By *model tracing*, that is by comparing the students' response to the set of possible legal actions and the set of known erroneous actions, the tutor is able to recognize whether the student is on a correct solution path, appears to be suffering from a known misconception, or has typed something unrecognizable. It uses that information to provide feedback accordingly.

The advantage of this approach is that it breaks down the difficult plan recognition problem to only tracing one problem solving step. So, this model is computationally efficient and cognitively accurate. Since in Circsim-Tutor we will use tutoring schemata to lead the students to get the correct answer by themselves, it is possible that we can model the student at each small step without doing too much inference under uncertainty.

3.3.2 Student Modeling in Sherlock II. Sherlock II is an intelligent coached practice environment developed to train avionics technicians to diagnose faults in a complex electronic testing system {Lesgold et al., 1992] [Katz et al., 1992, 1993]. The tutor presents trainees with a series of exercises of increasing difficulty. There are two main episodes in each exercise: *problem-solving* and *review.* During problem solving the student runs the troubleshooting procedure and he can ask for advice at any point while troubleshooting.

The group argues that student models in computer-based learning systems do not need to be precise and accurate to be useful. Their approach, which is based on fuzzy set theory, aims at building imprecise, or "fuzzy" diagnostic student models.

The student modeling knowledge base is a network of “fuzzy variables” -- the student variable lattice. Each node in the lattice is an indicator of some characteristic of the student capability.

The lower level variables represent distinct components of expertise:

Conceptual knowledge (e.g., understanding of how a relay card works)

Skills (e.g., ability to use an oscilloscope)

Behavioral dispositions (e.g., tendency to replace components rather than test them)

Affective factors (e.g., self-confidence vs. coaching dependency).

The higher level variables in the lattice represent abstractions over groups of these lower level competencies (e.g., ability to use test equipment, which includes the ability to use the digital multimeter, the ability to use the handheld meter, and the ability to use the oscilloscope). These variables come from cognitive task analysis and expert judgments. The fuzzy variables have five knowledge states: no knowledge, limited knowledge, unautomated knowledge, partially automated knowledge, and fully developed knowledge.

They have a set of updating rules for local variables like:

When the student's action meets certain constraints. These are a set of rules associated to update the variables. For example:

Is the test component on the active circuit path?

If not, then update the fuzzy variables understanding of the checkout procedures and system understanding, respectively.

Updating rules for global variables make use of weighted linear equations. For example:

Circuit testing ability = 0.85 * circuit testing strategy + 0.15 * circuit testing tactical ability.

Sherlock II provides advice at both the circuit path and individual component levels of investigation. At each level, it offers functional advice — how it works, strategic advice – how to test, and procedural advice – how to trace. For example:

If the model suggests that the student should be able to make the next measurement with only a little help, Sherlock II will give minimal support, thereby encouraging the student to interpret the hint himself, and to try to come up with the next measurement to make on his own before requesting help again.

If the student model suggests that the student will have difficulty (e.g., he has had trouble testing a particular type of component before), Sherlock II will give more directive help right away.

Sherlock II is a good example for the Circsim-Tutor group to study because:

1. it uses different variables to represent distinct components of expertise;
2. it computes its high level student performance from low level expertise;
3. it uses a good mathematical tool, fuzzy logic, to represent and update imprecise student models.
4. it leaves some of the instruction to a post problem reflection or 'review' phase, because they believe that learning from task situations requires significant cognitive effort. We could build a review (or summary) at the end of a stage or state some general rules when it is necessary.

3.3.3 Student Modeling in ANDES, Andes is an ITS for Newtonian physics, developed by VanLehn and others at LRDC (Learning Research and Development Center), at the University of Pittsburgh [Gertner et al., 1998] [Conati et al, 1997]. It is used by students in an unsupervised setting.

Andes includes four components: the homework assignment editor, the tutor, and the author's tool box. Inside the tutor there are three components: the workbench, the helper, and the assessor.

This module uses Bayesian reasoning to maintain a long-term model of the student's level of mastery of individual physics concepts and the student's preferred methods of problem solving and learning.

It is automatically generated each time the student selects a new problem. The structure of the Bayesian Network is taken directly from the structure of the solution graph. The network contains five kinds of nodes: Context-rule nodes, Fact nodes, Goal nodes, Rule-Application nodes, and Strategy nodes.

It also has a model-tracing component to see which solution the student appears to be working on, and which parts of it she/he has accomplished.

The helper tries to understand what plan or goals the student is pursuing as the student does an activity. It offers help when asked, and may sometimes offer unsolicited help. It can explain the feedback given by the workbench. If it detects an important physics misconception or a bad learning habit, it may engage the student in extensive multimedia instructional activities.

This helper is the one that will use the output from the student model. It includes three different helpers: procedural help, conceptual help, example study help, and feedback to equation entries.

The procedural help generates hints by using a Bayesian Network student model. First, Andes uses the BN to infer which part of the solution the student is working on and where she got stuck – a form of probabilistic plan recognition. Second, Andes uses templates to generate hints from nodes in the solution graph. For each goal and fact node in the solution graph, Andes has an associated sequence of hint templates, ranging from quite general to very specific. Andes will give hints from the most general hint first. Third, if the student asks for further explanation then give more specific hints until the most specific one – the answer. Or it may decide to give mini-lessons.

Andes is one of the most recent intelligent tutoring systems that has been successfully used by students. There are many things we can learn from Andes: to divide the student model into different modules, to give hints according to the student's learning states, and to use a Bayesian Network to model the student.

3.3.4 Student Modeling in MENO-Tutor. MENO-Tutor [Woolf, 1984] does not have a very complicated student model. But it is a very good system to study because it uses very simple but practical student model to support sophisticated tutorial planning.

MENO-Tutor is one of the tutoring systems in the MENO project, which started in the late seventies at the University of Massachusetts at Amherst. The whole MENO project attempted to build an intelligent tutor for novice Pascal programmers, including diagnosing non-syntactic errors in simple programs, to connect these bugs to underlying misconceptions, and to tutor the student with respect to these misconceptions, while the MENO-Tutor mostly focused on the remedial aspects.

MENO-Tutor attempts to capture the discourse strategies observed in human tutors who strive to be adaptive to their listeners. They argued that a successful machine tutor requires sophisticated knowledge that is necessary to carry on an acceptable tutoring discourse and a deep understanding of the student's knowledge [Woolf, 1984]. They studied human tutoring protocols and identified some of the rules and structures that govern this kind of behavior [Woolf, 1984]. This is particularly interesting to Circsim-Tutor since,

1. we also study human tutors' tutoring behavior,
2. we want to simulate our human tutors' behavior,
3. we want our machine tutor's behavior to adapt to the student's needs,

4. we want to find the rules and structures that can produce our human tutors' behavior.

The tutoring component in MENO-Tutor is a set of decision-units organized into three planning levels – a Discourse Management Network. It includes pedagogic states, strategic states, and tactical states on all three levels. Each lower level will refine the action of the higher level. The network is traversed by an iterative routine that stays within a predetermined space of paths from node to node, but the default path can be preempted at any time by meta-rules that move MENO-Tutor to a new path. The action of the meta-rule corresponds functionally to the high level transitions observed in human tutoring.

According to this network the tutor chooses among alternative discourses based on what the tutor knows about the student's knowledge and the discourse history.

The preconditions of the meta-rules determine when it is time to move off the default path. To do so, in the prediction part several things will be examined: the student model, the discourse model, and the domain model.

In the student model they will model whether the student seems confused or not from the tutor's view. Several facts are used to evaluate the student's knowledge state. For example, the number of questions asked, the number of incorrect responses given, and the extent to which the student's frontier of knowledge has been explored.

This kind of meta-rules are important to us, too, because we need to study rules about how our tutors change their methods (may correspond to the tutoring strategy here) or topics and what kind of student information they use. The question about what kind of student information should be used to fill in the meta-rules is important to us too.

3.3.5 User Modeling in TAILOR. Paris [1993] argued that there are (at least) two very different types of descriptions in naturally occurring texts that are characterized by means of distinct discourse strategies. One is the process trace, which allows for the generation of process-oriented descriptions and is related to the procedural discourse strategy; another one is the constituency schema, which allows for the generation of parts-oriented descriptions and is related to the declarative discourse strategy.

After she carefully analyzed some naturally occurring texts, she argued that as a result of the difference in organizing structures, the type of information that is included in the descriptions is also different: in the texts from the adult encyclopedias and the manual for experts, the information included is mainly structural, while in the texts from the junior encyclopedias and the manual for novices, the information included is mainly functional (or process oriented). One of the main differences between the two audiences at which the two groups of texts were aimed is their assumed level of domain knowledge. So the reader's level of knowledge about the domain affects the kind of information provided as opposed to just the amount of information. This is significant as it adds a dimension (the different kinds of information available) along which a system can tailor its answers to users having different levels of domain knowledge.

She suggested that the user's domain knowledge should affect the content of a description with respect to the kind of information to include in a text, and not just the level of detail. She proposes using the process trace when the user is relatively naive about the domain of discourse, and the constituency schema when the user has expertise about the domain.

Since users are not necessarily either naive or expert in a domain, she argues that they may have local expertise, knowing about some objects in the domain and not others. To describe objects to users with intermediate levels of expertise, a combination of the

two strategies presented for naive and expert users is appropriate. They have identified some heuristics that determine when to mix the strategies.

The first idea of studying TAILOR is can we use this user expertise to explain our tutor's behavior when they switch between the deeper level method and the top level method?

Another idea is that there may be two kinds of knowledge that our human tutor will try to tutor in Circsim-Tutor, too: the casual relationship knowledge (the concept map), which may correspond to Paris's process trace, and the object knowledge, some structure and functional knowledge about different objects in our domain (from that we could explain why x will affect y), which may correspond to structure knowledge in Paris's book, (which is better tutored as a constituency schema in TAILOR).

3.3.6 Summary of Interesting Examples. By studying the above systems and some other systems which include a student model component, I try to discover:

1. Why they need the student model component?
2. What kind of information does that component include?
3. How do they use that component?
4. Can we implement that component in our system?

There are a couple of questions, that directly come from this study, that we want to answer, including,

1. Could we model our student as a constraint model after he or she finishes the prediction table?
2. Is the Bayesian Network or the Fuzzy model useful for our project?
3. How should we evaluate the student's performance?

CHAPTER IV

RELATING TUTORING STRATEGIES TO STUDENT MODELS

4.1 Why Analyze Human Tutoring Transcripts

One of the goals of the Circsim-Tutor project is to produce natural tutorial dialogues like those human tutors have produced. Freedman, Kim and the others have marked up human tutoring transcripts and modeled their behavior as a hierarchical structure of tutoring goals [Freedman et al., 1998a, 1998b] [Kim et al. 1998]. Freedman also produced a planning framework to simulate this behavior [Freedman, 1996a]. It is a natural thought that we should study what kind of student model is used to support this kind of behavior, so we can build a computational model to simulate it too. Since the tutoring goal structure comes from human tutoring transcripts, it is possible that we could look at the same transcripts and uncover the supporting student model.

Instead of analyzing transcripts there is another method that has been used by the Circsim-Tutor project before – interviews with human tutors. This was a very good method in the beginning when we did not have much information about the student model, because:

1. Our expert tutors accumulate lots of common misconceptions from their long time tutoring experiences.
2. They can tell us what kind of information they prefer to use and what they most want the student to internalize.

The result of this method produced very interesting ideas when we begin to analyze the transcripts. But this method alone cannot produce a very complete and accurate student model, because:

1. Generally, all that information is not exactly rules and is very hard to simulate in a machine tutor.
2. The criteria that our human tutors use to evaluate the student are very rough; they say “the student is doing poorly” or “the student got stuck.”
3. Their decisions are mostly based on obvious evidence only.

On the other hand, transcript analysis is a very good method to get accurate and detailed results, and there are plenty of statistical and Machine Learning techniques that can be used to find useful patterns and rules from large amounts of data that cannot be handled by human beings.

In the following sections I will elaborate on how to find the possible student model from human tutoring transcripts and how to use this model to support tutoring decisions. First I will discuss some practical problems and general analysis. Then I will discuss in detail how to analyze the student model and its function based on a specific tutoring model.

4.2 Analyzing Specific Student Difficulties and Misconceptions

4.2.1 Varieties of False Knowledge. One difficult issue in building a student modeling component is that the student’s incorrect knowledge is far behind the expert knowledge. You can easily infer a correct solution from a set of correct rules, but

inferring an incorrect solution is much harder and more uncertain. Lots of student modeling researchers have studied the untraceable problem of detecting the false knowledge of a student [Self, 1990a]. Some of them suggested that we should not attempt to model too much or too deeply, or to build complex mechanisms that develop information about the student's cognitive state that will not actually be used by the end system, or will be too uncertain or abstract to be used effectively [Woolf and Murray, 1994].

In Circsim-Tutor we do not perform much inferencing on the student's problem solution path (but we may need to make some inferences if we want to seriously simulate the tutor responses to student initiatives). By carefully planning tutoring questions our tutorial dialogue also avoids lots of inference. But we still have some difficulty in detecting the student's false knowledge because the student's answers are full of uncertainty. Even if we can parse all the student's incorrect answers, relating them to the knowledge base and representing them are still very difficult tasks. We have not even mentioned the problem of extracting useful patterns and information for the instructional planner.

The variety of false knowledge that our students possess makes the student modeling problem even harder. Some of the false knowledge is confusion about concepts, some is inappropriate analogies, and some is even nonsense. Now we can detect them only by small clues from the student's answers or from the prediction table entries. So, how to analyze and model them systematically is still a difficult issue.

4.2.2 The Features That Human Tutors Use to Describe the Student's Current Knowledge State. Kim and others studied human tutor responses to student replies to a tutoring question [Kim, et al., 1998]. In their study they mostly used the

category of the student's answers. While this is one of the major factors, it is not the only one since after the same kind of student answers, our human tutors may have different responses. So, we need to study other kinds of information our human tutors have used and how this information has been used.

For example, Hume argued that our tutors' decisions are determined by the student performance [Hume, 1995]. Our expert tutors think that the prediction table is a good tool to find the student's error patterns [Michael et al., 1992].

After analyzing some of the transcripts, I found that the order in which high level tutoring is performed depends on the major misconceptions inferred from the prediction order.

Generally, our human tutors develop their tutoring based on the concept map: first tutor the incorrect variables in the shortest path to MAP, then tutor those in the other paths. This will be the default strategy for Version 3 in the DR stage too.

But in some cases, if the tutors can infer some major misconceptions from the student's prediction order, they may change the tutoring order to: first address the major misconceptions, then tutor the others along the causal relation paths.

For example, in a procedure in which TPR is the first variable affected by a perturbation, students often misunderstand the circulatory system and so incorrectly predict the change of RAP from MAP; then based on this value of RAP they predict SV and CO wrong. When our human tutors recognize this misconception their behavior is: first tutor the possible misconception from MAP to RAP, then lead the student to think along the correct causal path.

So we can see from the example that the determination of the order of high level tutoring is influenced by major misconceptions while in the general case it will be the default rule will be followed.

There are some other features that may affect our tutors' decisions. But to systematically analyze the necessary features for the student model, we have to base our decisions on a solid tutoring model. In the following sections I will describe the special tutoring model that I have used as a basis for my own work and the detailed analysis of the possible student model and its function.

4.3 Analysis Based on a Specific Tutorial Dialogue Model

4.3.1 Freedman and Kim's Tutorial Model. After several years of study, Freedman developed a sophisticated form of schema to simulate our human tutor's behavior [Freedman, 1996a]. She, Kim, and others then marked up some transcripts according to her schema and developed a hierarchy of tutoring goal and strategies [Freedman, 1996b] [Kim et al., 1998]. Their hierarchy includes the following levels:

- Tutoring strategy -- method
- Topics within the strategy
- Text generation primitive (inform or elicit) for each topic

This model served as the tutoring model for Circsim-Tutor Version 3 and the following analysis is mostly based on their structure and on their markups of tutoring goals and strategies.

4.3.2 How to Choose Schemata Based on the Student Model. In Freedman's thesis [Freedman, 1996a] she identified several schemata that our human tutors actually used in their one-on-one keyboard-to-keyboard tutoring. She also suggested that the

decision to choose a schema may depend on several sources, such as tutoring history, domain knowledge, student modeling, and so on. But we still need to know exactly how human tutors make decisions based on those sources, so that we make these decisions in the same way in our tutoring system as much as we can. This approach affects the kind of knowledge involved in the student model that I want to build.

It has been argued in CST project meetings that some schemata may demand more student activity than the others, so if the student assessment is high, the tutor may choose the schemata that require higher student activity. If the student assessment is low, the tutor may choose the schemata that require lower student activity. We need to test whether this argument is true or not. If it is true, we need to discover which assessment will help determine the choice best.

Some initial analysis of the transcripts shows that there are some rules that use the student model to make that kind of decision, such as:

If the student does not know that the mechanism for some variables is neural,
then use t-does-neural-DLR schema;
else choose t-show-contradiction schema.

Here *the t-does-neural-DLR* schema is:

<***T-does-neural-DLR***>
<**T-tutors-mechanism**>
<**T-tutors-DR-info**>
<**T-tutors-value**>

The *t-show-contradiction* schema is:

<***T-show-contradiction***>
<**T-present-contradiction**>
<**T-tutors-value**>

4.3.3 Decide on Topics. In Freedman and Kim's markup [Kim et al., 1998] each schema will include several topics, so, we need to decide which topics to include in a specific schema and which ones to leave out.

Our initial analysis shows that this decision is related to the student's detailed knowledge and misconceptions, i.e., what the student already knows and what the student's difficulty really is.

For example, the decision whether or not to include the *DR-info* topic in the t-does-neural-DLR schema may depend on the difficulties that the student has.

In Example 1 the student correctly answered what controls TPR, so from our tutor's view s/he didn't have difficulty about the mechanism, but since s/he incorrectly predicts TPR's value, then it is possible that the student has misunderstood what DR means. So in this example, the tutor mentions the DR's meaning in detail.

Example 1.

K10-tu-29-4: Can you tell me how TPR is controlled?
K10-st-30-1: Autonomic nervous system
K10-tu-31-1: Yes.
K10-tu-31-2: **And the predictions that you are making are for the period before any neural changes take place.**
K10-tu-31-3: So what about TPR?

In Example 2 the student has some real difficulty in understanding the mechanism that controls Inotropic State, but the only error is in this neural variable. So after the tutoring of the mechanism our human tutor infers that the student got IS wrong basically because s/he does not understand the mechanism, not because of a misconception of the meaning of DR (since the student predicted other neural variables correctly). So in the following example what our tutor actually does is only mention 'in the DR' instead of explaining it in detail as in Example 1.

Example 2.

(After long tutoring of the control mechanism of IS)

K47-tu-60-1: Then if IS is only affected by changes in sympathetic stimulation of the heart muscle, what must be the value of IS **in the DR** period?
K47-st-61-1: 0
K47-tu-62-1: Correct.

4.3.4 Decide Primitives. There are two kinds of primitives in Freedman and Kim's annotation: *t-elicit* and *t-inform*. The distinction between them is: t-elicit produces a question to the student about a piece of information. It may need more action from the student -- the student is active. On the other hand, *t-inform* produces an explanation of a piece of information to the students. It requires less action from the student -- the student is passive.

Again since the two primitives affect the student's activity level, we need to test whether the choice of the primitives is related to the student's performance or not.

For example, if we look at the *DR-info* topic inside the t-does-neural-DLR, our tutor's behavior seems to really depend on the student's assessment in Examples 1 and 3.

In Example 1, the student predicts two neural variables incorrectly and two other non-neural variables incorrectly. So our tutor's evaluation of the student is quite low. And he actually chose the primitive that needs less activity from the student. He informed the student about the *DR-info* directly.

In Example 3, the student made only one error, in one neural variable, so the assessment is quite high, and our human tutor chose the primitive that requires more student activity. He asked the student for the information.

Example 3.

K48-tu-46-2: What mechanism does this?
K48-st-47-1: autonomic nervous system?
K48-tu-48-1: right.
K48-tu-48-2: **and during DR what changes in ANS activity occur?**

But we need to analyze more transcripts to find out if these choices also depend on other information about the student and the conversational context, especially the detailed knowledge of the student. For example, what the student learns from the DR stage may often happen to be a *t-inform* primitive of the same topic in the RR or SS stage.

4.3.5 Decide on a Response Strategy. The determination of our tutor's response strategy and how it is related to the student model may deserve the most attention since it may involve all the different parts of the knowledge in the student model and it may be more directly affected by the student model. One of the big problems in Version 2 is that it has only a few response patterns, and this is one of the aspects that we want to improve in Version 3.

Our human tutors try to maintain a global tutoring plan but at the same time reply to issues raised by students. As a result, our tutor's responses tend to have the following structure [Freedman & Evens, 1996]:

Response to the student's previous statement
    Acknowledgment of student's response
    Content-based reply to the student
Next step of tutorial plan

Each segment is optional, although at least one must be present. Those three parts may be instantiated by a variety of response strategies to the former turn, for example:

1) Only new materials;
2) Acknowledgment and new materials;
3) Content oriented reply and new materials;
4) Acknowledgment, content oriented reply, and new materials; and so on.

Each part has its own categories that can be chosen. For example:

The acknowledgment may be positive, negative, or a mixture if the student's answer was partially correct.

The content oriented reply may be a correction of some misconceptions, some restatement, or simply a short summary.

The new material may be the next primitive -- a hint, a restated question -- inside the same topic, or the next topic inside the same schema, or a totally new schema.

Each sub-category may be instantiated as different patterns too. For example: correction of misconceptions may be instantiated as:

- Only point out the errors;
- Simply give the answer;
- Point out the erroneous answer and compare it to the correct concept;
- Simply lead the student to look at the concept from another point of view, etc.

So, to decide which pattern to choose, along with how to instantiate the sub-category and the sub-sub-category we have to analyze a lot of transcripts to find what really influences our human tutor's decisions, so that we can implement as much as we can in Version 3.

Since the tutoring setup is one-on-one, we believe that our human tutors make the student's state a major consideration in choosing a response. Certainly the immediate answer of the student to the last question is an important factor in determining the response strategy [Kim et al., 1998]. After different students give different answers (in different categories) to the same question, our human tutor's responses are quite different:

Example 4: Answer category is incorrect.

K13-tu-37-3: First, what parameter determines the value of rap?
K13-st-38-1: Venous return and peripheral resistance influences return.
K13-tu-39-1: Not in the way that you seem to think.
K13-tu-39-2: If CO is made to vary what effect will that have on the central venous compartment or rap?

Example 5: Answer category is don't know.

K26-tu-64-1: Do you know what determines the value of rap?
K26-st-65-1: No
K26-tu-66-1: Rap is essentially the same as central venous pressure.

Example 6: Answer category is near-miss, which is defined as 'answers that are pedagogically useful but not the desired answers,' by Glass [1997] .

K22-tu-40-1: What parameter DOES produce a change in rap?
K22-st-41-1: End diastolic volume
K22-tu-42-1: When you talk about EDV what structure in the heart are you referring to?

Example 7: Answer category is misconception.

K11-tu-57-2: If CC is under neural control and we are talking about the period before any change in neural activity then CC???
K11-st-58-1: But, it is ALSO under intrinsic control
K11-tu-59-1: You are confusing Starling's Law with a change in contractility.
K11-tu-59-2: The length/tension relation of the heart is not a change in contractility.
K11-tu-59-3: A change in contractility moves the length/tension curve from one location to another.
K11-tu-59-4: Increased contractility means that at a given EDV you get more contractile performance out of the ventricle.

Our tutor will give different responses not only for different answer types, but also for different types of incorrect answers (different types of misconceptions and the reasons causing them). Since one of CST's goals is to remedy the student's misconceptions, it is very important for us to collect and categorize different misconceptions and the corresponding remedy strategies.

Here are some typical categories of student's misconceptions and the tutor's remedy responses that I already collected.

Type 1: caused by similarity between concepts. The typical example is the confusion about CC and Starling Law. The corresponding tutoring method is to explain each concept and the relationship between them.

Type 2: caused by misuse of the domain principles. For example, consider the reflex in the DR stage, or misuse the following two rules: RAP directly affects CO through SV if RAP changes first, CO inversely affects Rap if CO changes first. The typical tutoring method is to mention the problem solving context and state the general rules that declare which situation should use which rule.

Type 3: caused by inappropriate analogy. For example: lots of students think the increase in MAP will cause RAP to increase because they think that an circulatory system behaves like a circular pipe. The typical tutoring method is to lead the student to look at the problem in the correct way.

Type 4: caused by insufficient knowledge of the facts. For example: lots of students do not know which part has the agonist receptor. The typical tutoring strategy is to give some hint and lead the student to consider the related objects.

Type 5: caused by adding some other knowledge to the problem solving procedure. For example: the heart will be stronger if it contains more blood.

From these examples I believe that to respond to the student's misconceptions we need to categorize them and map them to specific correction patterns.

Also for the same 'do not know' answer category our human tutor may give different hints, some are more obvious and some are a little vague, so we need to decide how the choice of different hints is related to the student assessment.

And we are looking for other factors from the student model such as the general knowledge that the student already knows. This is important, particularly if we ask ourselves a question: How can we automatically decide the content for the content oriented reply part and also how can we automatically generate hints for the new material parts since they are not in the schemata structure. And this is a real problem, when we tried to add explanations and hints for the current Version 2, we have to add a whole new hinting model.

4.3.6 Summary of Transcript Analysis. From the above analysis we can see that the student model has been used to support different tutoring decisions, from the high

level tutoring order decisions, to the lowest level of detailed response strategy decisions. But for decisions of different levels made by different components, the expected information about the student may be different. So we certainly need to include several kinds of student information in the student model to support different needs.

CHAPTER V

USING MACHINE LEARNING TO FIND TUTORING RULES AND USEFUL STUDENT MODELING FEATURES

5.1 Why Machine Learning

Chapter IV showed that transcript analysis is a powerful method to get necessary information about tutoring and student modeling. But there is one problem with analyzing transcripts manually – we can handle only a small portion of the transcripts. When the number of transcripts become very large it is very hard for human beings to handle them. So we need to find an automatic method to do this kind of analysis. In this chapter I will discuss applying Machine Learning techniques to automatically learn useful tutoring rules from large numbers of transcripts.

There are several methods that people use to analyze transcripts automatically: statistical methods, Machine Learning methods, and so on. I chose Machine Learning methods because they can generate new rules that we can use in our system directly, while statistical methods usually only give concurrence.

Also, Machine Learning is a very good method to automatically find useful rules for large data sets, too big for human beings to deal with. We have accumulated over 5000 turns of transcripts of human tutoring sessions. It is hard to analyze them manually.

Another advantage of Machine Learning is that it is an objective method while manual analysis may be influenced by human bias. But we will also use other methods to help us construct better rules, since Machine Learning has its own limitations too.

5.2 Using Machine Learning to Discover What Kind of Information the Tutor Needs to Know in One-on-One Tutoring

Basically this is the feature selection problem in the Machine Learning field. In the real world there may be lots of information that can affect the decision. But there are some features that are particularly powerful, and we need to find them.

In our experiment we used more than ten conditional features that we think may be related to the decision in the database. So it is necessary to find an effective way to choose which conditions are really powerful.

In the student model there are three types of information that may useful for the decision: student assessment, general knowledge (or domain knowledge), misconceptions (which are different from the domain knowledge).

Within CST the most debated question that we need to decide is the student assessments: whether they are useful or not; if useful, how to calculate them. So, we need to develop some experiments to test them.

Another problem is about the student misconceptions. How can we get the data automatically? We certainly cannot extract it automatically from the current SGML markups. But since one of our goals is to remedy the student's misconceptions, our tutor's response will definitely be affected by the student misconceptions. Also in the real tutoring system we built a diagnostic model to analyze the student's misconceptions, so this information is directly related to the future system. One possible solution to this problem is that we can manually add this information to the transcripts and so we may include it as a conditional feature for the Machine Learning experiment.

5.3 Applying Machine Learning to Discover How Human Tutors Make Decisions Based on the Student Model, Experiments and Results

We have designed several experiments to find out how our human tutors make decisions on different levels by using Quinlan's [1993] C4.5 learning algorithm. The results show that this Machine Learning method can help us find some real tutoring rules [Freedman et al., 1998b, 1998c].

Here are some experimental results showing that our human tutor's decisions are truly dependent on the student's state along with domain knowledge and some other facts.

5.3.1 Experiment 1: Choosing a Response Strategy. In the first experiment, we were interested in categorizing the tutor's response strategy directly after a student replies to a tutoring question. In the markup by Kim and others [Kim et al., 1998], student replies are categorized as one of the following:

Correct,
Correct but hedged,
Partially correct,
Near miss,
Don't know,
Incorrect.

We tried a number of features, of which the most explanatory turned out to be:

The category of student response
Whether the variable is neural or non-neural

Sequence of the variable within its category (neural or non neural)

How many previous attempts had been made to tutor this variable

Possible tutor behaviors were:

Proceed.

Give info and proceed.

Give info and re-elicit.

Give answer and proceed.

Nested method.

New method.

For this experiment we had 57 cases of tutor behavior following a student response. C4.5 produced the following decision tree, which required no simplification:

If student answer was correct,

proceed

If student answer was correct but hedged

give info and proceed

If student answer was partially correct

give info and proceed

If student answer was a near miss

introduce a ***nested method***

If student answer was incorrect

if variable is neural then

if this is the first attempt for this variable

try a new method

else {subsequent attempts}

give info and re-elicit

else {non-neural variable}
if first non-neural variable in dialogue
give answer and proceed
else {subsequent variables}
give info and re-elicit

This tree provides an algorithm for updating the tutoring plan based on the student response. It correctly classifies 50 of our 57 cases, yielding an error rate of 12%.

From the decision tree we can see that the most important factor in this case is the student's previous answer category.

5.3.2 Experiment 2: Choosing an Explicit Acknowledgment. In the second experiment, we explored the style of acknowledgment issued by the tutor in response to a student answer. Here issuing an acknowledgment is part of the tutor's response in Freedman's tutoring goal hierarchy [Freedman, 1996a], while Brandle [1998] studied acknowledgments from the view point of Clark's [1996] joint action theory. In 62 cases of tutor acknowledgments, we categorized the observed acknowledgments as follows:

Positive, e.g. "Correct."
Partial, e.g. "Well that's partly correct,"
Negative, e.g. "No."
Nil, no explicit acknowledgment.

The possible features were the same as in the previous experiment. In this experiment C4.5 found significance in the number of times the tutor retried the ***t-elicit*** to which the student was responding. The resulting decision tree is quite messy, with 28 nodes before simplification and 15 nodes after, and the simplified tree misclassifies more

than 20% of the cases, so we do not have a comprehensive set of rules to cover all cases. However we did obtain a few fairly solid rules:

If answer was partially correct
    then issue ***partial*** ack
If answer was correct
    then issue ***positive*** or ***nil*** ack
If the answer was incorrect
    if we are working on the first incorrect vbl in its catg
        if we are on the first attempt to tutor that variable
            then ***nil*** ack
            else {a subsequent attempt}
                negative ack

There are two reasons that may cause this inaccuracy and the huge tree:

1. Too few examples. Our learning data set is relatively small, especially for some of the categories.
2. The condition features are not complete. For example, in Brandle's experiment, he used local assessment as one of the features and it turns out to be very powerful. But using the current markup structure, it is not easy to calculate the local assessment.

But we do see that the choice of the type of acknowledgment mostly depends on the student's answer category from this experiment.

5.3.3 Experiment 3: Choosing a Realization within a Topic. In this experiment, we tried to decide which realization form to choose after a topic has been chosen. For this experiment we had C4.5 build a decision tree to classify 18 cases where ***t-tutors-dr-info*** is realized as a ***t-elicit*** or as a ***t-inform.***

The ***t-does-neural-DLR*** is one of the methods that our tutors use frequently to teach neural variables. It has the following structure:

<***T-does-neural-DLR***>

<**T-tutors-mechanism**>

<**T-tutors-DR-info**>

<**T-tutors-value**>

The following decision tree was obtained from C4.5:

If all three neural variables were incorrectly predicted

then use ***t-elicit***

else use ***t-inform***

This rule, which uses only the student's prediction about the three neural variables, correctly classifies 15 of the 18 cases. We proved that the decision about the choice of realization is related to the student performance.

5.3.4 Experiment 4: Choosing a Tutorial Strategy. The fourth experiment explored how tutoring operators are chosen at the method level. Each case represented one attempt to tutor one variable. The features we chose were:

The total number of variables predicted incorrectly

The number of neural variables predicted incorrectly

Whether the variable is neural or non-neural

Sequence of the variable within its category (neural or non-neural). (The tutors usually tutor all the neural variables first, followed by the others.)

How many previous attempts had been made to tutor this variable

In our sample, five different tutoring methods were attested:

- t-does-neural-dlr, tutoring using a guided series of questions exploring the behavior of a neural variable
- t-shows-contradiction, tutoring using an inconsistency in the predicted values of the variables
- t-tutors-via-determinants, tutoring using the set of variables that have an effect on the incorrect variable
- t-moves-forward, tutoring using causal reasoning from a known-correct variable to the incorrect one
- t-tutors-via-deeper-concepts, tutoring using other concepts and more specific physiological parameters

In this experiment we had 23 cases, each with five features. There were five possible outcomes. The original tree produced by C4.5 is too specific to be useful. C4.5 produced the following simplified tree:

If var is neural

if first var in category, use ***t-does-neural-dlr***

if second, use ***t-shows-contradiction***

else {variable is not neural}

if first vbl in catg, use ***t-tutors-via-determinants***

if second, use ***t-moves-forward***

This tree misclassifies three of the 23 cases, for an error rate of 13%. Of the three that are misclassified, two involve second and subsequent attempts to tutor the same variable. These attempts tend to employ the less common tutoring methods. Although we have coded about half of the transcripts involving these tutoring methods, we have coded only about 10% of the total corpus. We hope that completing the annotation of the corpus will produce decision trees that cover other phenomena of interest to us, including criteria for invoking the lesser-used tutoring methods.

What this tree has uncovered is a high-level plan for the entire tutoring session. Previous research has found an algorithm for sequencing the variables to be tutored, and this rule tells us which tutoring strategy to use for each variable.

As Freedman and Glass pointed out [Freedman et al., 1998b], within each group of variables, neural and non-neural, C4.5 noticed an interesting rhetorical pattern. The second variable to be tutored in each group uses a method that builds on knowledge taught in the first variable. For example, a common rhetorical pattern for tutoring two non-neural variables v_1 and v_2 is to tutor v_1 via determinants, then move forward to v_2 as follows:

<***t-corrects-variable variable=v$_1$***>

<***t-tutors-via-determinants***>

What variables in the prediction table determine v_1?

(v_1 is tutored, based on its determinants)

</***t-corrects-variable***>

<***t-corrects-variable variable=v$_2$***>

<***t-moves forward***>

And what effect would v_1 have on v_2?

(v_2 is tutored by moving forward from v_1)

</***t-corrects-variable***>

The pattern for neural variables is similar.

In this experiment no student information has been used in the simplified tree, but by carefully analyzing the data I found that this is mostly caused by the coverage of the data. Since the mark up process is a very time consuming work, our markups only include the DR stage for one procedure. So most of the data involves only the major tutoring methods and can be classified without much information about the student.

5.4 Further Discussion

In the last two years several researchers have applied Machine Learning techniques to transcript analysis, such as dialogue act prediction, cue word usage, planning rules, and discourse segmentation. For example, Van der Linden and Di Eugenio used C4.5 to learn micro-planning rules for preventative expressions [Van der Linden and Di Eugenio, 1996a, 1996b].

In the AAAI Spring Symposium 1998 '*Applying Machine Learning to Discourse Processing,*' around 60 researchers who are interested in exploring the potential

contribution of machine leaning to discourse interpretation and generation gathered together to discuss how can Machine Learning techniques help discourse analysis. They addressed issues from the discourse processing and Machine Learning points of view.

From the discourse processing point of view:

What discourse understanding/generation tasks are or are not suitable for processing using ML-acquired models?

How should traditional approaches be integrated with ML approaches?

What types of pragmatic knowledge (e.g., discourse recipes, cue phrase classification) can or cannot be acquired by Machine Learning?

What kinds of categories and features can be tagged automatically and/or reliably?

How can useful features be identified?

From the Machine Learning point of view:

What Machine Learning techniques may be suitable for acquiring knowledge for discourse processing? How do they compare in terms of accuracy, efficiency, speed, and amount of training data needed?

What discourse corpora are currently available for Machine Learning? What other corpora are needed?

What characteristics of discourse processing cause problems for existing Machine Learning techniques?

We also need to consider some of these questions to further our study of applying Machine Learning techniques to analyze tutoring transcripts. For example, how to integrate the Machine Learning results with other research results such as manual analysis and interviews; how to make the process more automatic; how to apply recent data mining technology to this process and so on.

CHAPTER VI

ADAPTIVE HINTING

In Chapter IV I described the transcript analysis used for student modeling and the role of the student model. In Chapter V, I applied Machine Learning techniques to automatically find rules and useful patterns from the transcripts. Also there are lots of research results about tutoring analysis in this group [Freedman, 1996a] [Kim et al., 1998]. In this chapter I will try to put most of those results together to build a computational hinting model for Circsim-Tutor Version 2 and try to convert this model to plan operators using the Atlas Plan Engine [Freedman 1999] for CIRCSIM-Tutor Version 3.

6.1 Introduction to Hinting

6.1.1 Importance of Hinting. Hinting is a general and effective tutoring tactic in one-on-one tutoring when the student has trouble solving a problem or answering a question. In many student-oriented tutoring systems, the machine tutor will give hints when the student asks for help, e.g. Andes [Gertner et al., 1998]. In this tutoring setup, the central issue of hinting is to help the student recall the related domain rules or facts that the student may have trouble with. In a system where the tutor has control over the conversation and asks the questions, hinting is also a good strategy to help the student find the expected answer when the student gives an unexpected one. But in this tutoring setup, not only the student's possible weakness but also the tutor's plan and the tutoring context are important for issuing hints. Since there may be more than one pedagogical plan for tutoring a domain concept, the hinting strategy is closely related to the tutoring method or tutoring plan, although the detailed content of each hint is closely related to the

domain concept. So how to issue a follow-up hint which is helpful to the student, coordinated with the tutoring plan, and coherent in the dialogue context is an important issue in this tutoring setup.

6.1.2 Earlier Work on Hints in Circsim-Tutor. Hume et al. [1996] studied the use of hints by experienced tutors in the hope of formulating a strategy for using hints in an ITS. They observed that human tutors frequently use hints as a pedagogical tactic. However, the theory of hints in their framework is too broad, as they identify most tutoring moves as hints in some contexts. Furthermore, these hints were defined without reference to the structure required by the CIRCSIM-Tutor planner.

The original CIRCSIM-Tutor v. 2 produced very general hints to ask about physiological variables missed by the student. But it failed to tailor the content to the student's previous answer, as it issued only fixed hints such as "Think about the value of <the desired physiological variable>." To improve the hinting capability in CIRCSIM-Tutor, we started by adding hints, tailored to the student answer, which were given in response to a partially correct answer (this is the main situation in which the tutor gives hints in the original CIRCSIM-Tutor). This improved version was used by 24 students from Rush Medical College in April 1998. After reading the log files from this experiment, we found that the new hints were effective but there were other kinds of student answers that the system failed to respond to with follow-up hints.

For this reason we broadened the hinting ability in CIRCSIM-Tutor to include responses to other categories of student answers. This new version was used by 50 students from Rush Medical College in November 1998. We will discuss the experimental results later in this paper. This improved version, which also has a new input understander [Glass, 1999], is robust and useful enough that our expert tutors from

Rush Medical College believe that it can be used without supervision by medical students as a regular part of the curriculum.

6.2 Hinting Strategies in Human Tutoring

In an interview with our expert tutors they identified two rules for how they give hints:

First give evoking terms or synonyms

Otherwise try to give an intermediate step.

These two rules indicate that human tutors are trying to help the students think actively (by giving more evocative language) and also trying to help them think along the right chain of causal relationships by giving a small step along the causal path. Hume [1995] classified hints into two general categories -- point-to hints and convey-information hints – that are similar to these two general rules.

These two rules cover many of the cases of the human tutors' usage of hints, but they are too general and abstract to actually implement hinting in CIRCSIM-Tutor. So we analyzed transcripts of human-to-human tutoring sessions conducted by the expert tutors in order to identify the types of hints used in different situations and identify more specific strategies that could actually be used to build an ITS.

Hints occur in many different surface forms. To implement them in a principled and useful way in a real ITS, we need to identify the underlying principles in order to avoid just giving canned hints in each situation. So we want to isolate hinting strategies that are not dependent on specific domain facts or rules.

By using machine learning methods and reading transcripts, we found the following examples of hinting strategies frequently used in our human tutoring transcripts.

6.2.1 Strategy: Give an Intermediate Causal Link. This is one of the rules indicated by our human tutors. It actually has three sub-rules, each related to a different tutoring plan. Suppose there are several causally related physiological variables A affects X affects B, where the tutor usually teaches the relationship between A and B, ignoring intermediate steps like X.

- If the tutor asked which variable is affected by a change in A, then mentioning the link from A to X can be an effective hint toward the answer B.
- If the tutor asked which variables cause a change in B, then mentioning the link from X to B can be an effective hint backward toward the answer A.
- If the tutor asked how A and B are related, then mentioning either of the relationships from A to X or from X to B can be a hint.

By giving hints like this, the tutor offers a small piece of information relating the variable in question to the desired answer. The pedagogical expectation is that the student will think along these lines and find the desired answer.

6.2.2 Strategy: Refer to an Anatomy Object. Although our tutors prefer to focus on the physiology, they occasionally point to an anatomy object to help the student concentrate on the right part of the cardiovascular system if the student can not answer a question. This kind of hint is especially useful when the student has trouble finding the first variable affected by a physiological change. For example:

T: What is the first variable affected by the alpha agonist?
S: I don't know.
T: Where in the CV [cardiovascular] system are alpha receptors found?

6.2.3 Strategy: Point Out the Laws of Physics Involved. Although our domain is physiology, it is occasionally useful for the tutor to point to some physics rules to help the student visualize why the causal relation should be the way it is. For example:

T: When MAP [mean arterial pressure] increases, what will that affect?
S: (incorrect answer)
T: When MAP increases, it's harder for the ventricle to pump blood.

6.2.4 Strategy: Give Evoking Terms or Synonyms. While most of the time our tutors use a specific set of physiology terms in order to encourage students to use the same terms, they sometimes choose more evocative phrases. For example, in certain contexts they often use "afterload" as a synonym for "mean arterial pressure," evoking images of the pressure the heart is pumping against. This strategy is used mostly when the tutor is tutoring the causal relationship from mean arterial pressure to stroke volume after an incorrect student answer.

6.2.5 Strategy: Linguistic Hint. Since our human tutors use natural language (just as our tutoring system does), they sometimes give subtle linguistic hints which

include very little domain information. These hints are intended to help the student to think more actively.

A typical example occurs when the tutor is expecting several parameters from the student and the student gives only some of them. The tutor may simply reply with "And?" to indicate that more is expected.

6.2.6 Other Strategies. The above strategies are the most frequently used. There are also some other strategies that are used infrequently or are used only in special tutoring situations. These include pointing out the function of a drug, using capital letters to indicate a core variable, giving a definition, pointing out the problem-solving context, and referring to an equation.

6.3 Implementing Hinting in CIRCSIM-Tutor

Now that we have analyzed the hinting strategies of human tutors, we will discuss our implementation of these strategies in a running intelligent tutoring system—CIRCSIM-Tutor. Although there may be some deeper cognitive reasoning behind the human tutors' hinting strategy, we do not model such reasoning in the tutoring system. First, we lack a sufficiently comprehensive cognitive theory of hinting. Second, from a practical point of view, our analysis of human tutors' hinting strategies demonstrates that we can generate precise hints without invoking such a theory. The following algorithms describe our simulation of human tutors' hinting behavior.

6.3.1 Factors Determining the Hinting Strategies. There are several factors that may affect the choice of a specific hint: the tutoring topic, tutoring context, tutoring

history, student's answer, and so on. From our interviews with human tutors and the study of their tutoring transcripts we find several factors to be particularly relevant.

First, to be pedagogically useful, a hint has to be related to the tutoring topic and be useful in helping the student find the expected answer. So the tutoring topic is important.

Second, the student's answer is important since hints are intended to help the student figure out the expected answer from what he or she has already said.

Third, the specific question used by the tutor, which is a reflection of the high level tutorial plan, is important because there may be several questions available for tutoring the same concept. Different kinds of tutor questions may indicate a different conversational context or focus.

Finally, the tutoring history is also important, especially for the second or third hint in a row. The tutor needs to base further hints on earlier ones for two reasons. From a discourse point of view, it increases the coherence of the conversation. From a pedagogical point of view, it makes the tutoring logic stand out more clearly.

6.3.2 A Classification Model for Student Answers. We added a classification module to categorize student answers. Below are the categories that we use to classify students' answers:

1. Correct
2. Partially correct answer, i.e., some part of the answer is correct and the rest is incorrect

3. Near miss answer, which is pedagogically useful but not the desired answer [Glass, 1997]
4. "I don't know" answer
5. "Grain of truth" answer, where the student gives an incorrect answer, but also indicates a partially correct understanding of the problem [Woolf, 1984]
6. Misconception, a common confusion or piece of false knowledge about the concept being tutored
7. Other incorrect answers
8. Mixed answers, i.e., a combination of answers from other categories

These categories, which were abstracted from our analysis of human tutoring transcripts, are one of the features used to decide which hint to give. The more information the tutor can find in the student's answer, the more specific the hint can be. Although the categorization is based on the domain model, it is important to recognize that it is largely a pragmatic categorization, i.e. a correct answer is one which our human tutors do not feel the need to correct or augment.

6.3.3 Use of a Quantitative Student Model. Hume et al. [1996a,b] observed that human tutors maintain a rough assessment of the student's performance. They argued that when and how to give hints is based on that measurement of student performance. Although our definition of hint is narrower than theirs, we still feel that student performance is a good criterion for deciding when to deliver hints rather than giving the answer. Thus we added a student performance module to CIRCSIM-Tutor's original student

model. It includes four levels of measurement: global assessment (total measurement of the student so far), procedure-level assessment (measurement for each problem the student is asked to solve), stage assessment (measurement for each of the three physiological stages in a problem), and the local assessment attached to each variable that has been tutored. The local assessment is updated after each tutoring interaction and other assessments are calculated from the local assessment. If the student's performance is too low, the tutor gives the answer instead of issuing a hint. This model will be explained further in Chapter VII.

From experiments with medical students, we have found that we need other history data along with the assessment of the student for deciding between giving a hint and giving the answer, especially when the student gives the same kind of wrong answer twice in a row.

6.3.4 Identifying the Possible Hinting Strategies. From our analysis of human tutoring transcripts, we abstracted a set of hinting strategies, detailed below. We then built a hinting algorithm for each category of student answer. Each answer category is associated with a predefined list of strategies. Some of the algorithms are quite simple. For example, if the student gives a near miss answer, the tutor responds with a leading question that points to an intermediate link from the near miss to the correct answer. Some of the algorithms are more complex. For example, if the student's answer is incorrect, there are several strategies available. If the tutor is tutoring a causal link in the forward direction, most hinting strategies focus on evoking terms related to the variable already mentioned or on giving an intermediate link in the forward direction.

6.3.5 Using Heuristic Rules to Rank the Strategies. If the tutor still has several strategies to choose among, the tutor ranks the possible hints using heuristic rules that attempt to generate more specific hints first. We consider hints in the following order:

1. Hints that are specifically related to the student's answer
2. Hints involving equations
3. Hints involving evocative synonyms
4. Hints involving a related anatomical object
5. Hints that give an intermediate logical step
6. Others

6.3.6 Locating Appropriate Content. The result of this procedure is a list of hint types only. To decide the details of the content in a hint, the tutor searches the domain knowledge base for each of the available hinting strategies. For the first hinting strategy, it looks to see if the knowledge base has an entry for the concept currently being tutored. If the domain knowledge has an entry, then the search terminates; if not, the algorithm tries the remaining hinting strategies in sequence. If no entry is available for any of the possible strategies, the tutor gives a default hint.

To support the hinting strategies that we identified from the human tutoring sessions, we are in the process of extending our domain knowledge base with additional evocative terms, related anatomical objects, and related physics rules.

6.3.7 Using Templates to Deliver Hints. We use hint templates determined by the content selection algorithms discussed above to deliver the hints. For example:

Like <the related variable in the student's answer>, the other determinant is also related to <the anatomy object>.

But what I am asking is <definition of the tutored variable>.

Do you remember an equation: <the variable been tutored> =…?

6.4 The Model in Practice

6.4.1 Derivation of a Hint. The following example illustrates how the machine tutor determines the follow-up hint step by step after the student gives an unexpected answer. Suppose the recent dialogue is:

T: What determines CVP [central venous pressure]?
S: I don't know.

Here the category of the student's answer is "I don't know." If the tutor decides to give hints, the hint algorithm related to the "I don't know" answer will be evoked. Since there are several strategies available, the algorithm will first check the tutoring plan. Here the system is trying to tutor a causal relation backward. So the possible hinting strategies reduce to:

- Find equations related to the dependent variable.
- Point to a feature of an anatomy object related to the dependent variable.
- Give an intermediate step backward from the dependent variable.

Using the heuristic rules above, this list is the final list after ranking the preferences. Then the tutor checks the domain knowledge base and finds that the second

strategy has suitable content available. Finally the tutor will find the related hint template and deliver the hint as:

T: Remember the CVC [central venous compartment] is very compliant. So what determines CVP?

If the student still can not get the correct answer, the tutor could issue a further hint giving an intermediate step between CVC and CVP. But if the student gives a near miss answer, e.g. CBV (central blood volume) instead, the tutor could use the near miss as a basis for issuing a follow-up hint instead. That hint might be expressed as a question pointing to an intermediate link between the near miss and the desired answer:

S: How about CBV?
T: What determines CBV?

6.4.2 Influence of the Tutoring Question. The wording or intent of the tutor's questions is not an issue in systems where the student asks the questions. But the opposite is true in CIRCSIM-Tutor, particularly when the question indicates the direction the tutor is following along a causal link. When the tutor is teaching the relationship between A and B, different questions can address the same causal link in different directions. In response to an incorrect answer, each question might benefit from a different hint. In Dialogue 1 below, the tutor tries to teach about the link from cardiac output to central venous pressure, working backward from CVP. In Dialogue 2, the tutor works forward from CO. Although the student gives an incorrect response in both cases, the resulting hint is different.

Dialogue 1:

T: What determines CVP?
S: (Incorrect answer.)
T: Remember the CVC is very compliant. (Looks backwards from CVP to the volume of blood in the CVC.)

Dialogue 2:

T: When Cardiac Output decreased, how would that affects the value of Central Venous Pressure?
S: Decrease. (Incorrect answer.)
T: No. When CO decreases, think about what will happen to the volume of blood in the central venous compartment. (Gives a forward hint from CO to the volume of blood in the CVC.)

6.4.3 Influence of Tutoring History. The tutor keeps a tutoring history to avoid repeating the same hint and to make sure that the hints do not return to a causal relationship that was already tutored if there are several hints in a row.

T: What are the determinants of Stroke Volume?
S: It is RAP [right atrial pressure]. (Near miss.)
T: And what determines RAP? (Hint based on the near miss.)
S: EDP [end diastolic pressure]. (A near miss, but further from the correct answer than RAP.)
T: EDP comes from RAP. Think again, what parameter in the prediction table determines RAP? (Point out the relation between RAP and EDP, then give a further hint based on RAP.)

If we did not keep a dialogue history, the tutor could ask “And what determines EDP?” In that case the tutor would be following the student’s lead away from the right answer.

6.4.4 Sample Output. Although our model of hinting is based on a number of simple algorithms, it can generate a variety of hints that allow the students to answer for themselves rather than being told. Below is an example produced by CIRCSIM-Tutor using our model.

T: What are the determinants of Stroke Volume?
S: Heart Rate and Cardiac Output. ("Grain of truth" answer, i.e. wrong but pedagogically useful.)
T: Well, it is true that CO = SV * HR. But what I was asking is what determines how much blood is ejected from the heart each time it beats (the SV)? (Hinting strategy: give definition.)
S: Preload. (Near miss.)
T: Right. And what represents preload in the prediction table? (Hinting strategy: point to the intermediate link from the near miss. Note that the tutor gave a positive acknowledgment ("right") to the near miss, showing that there is not always a one-to-one relationship between student evaluation and the correct discourse move.)
S: Central Venous Pressure. (Correct, but still missing one determinant for the original question, so we categorize it as a partial answer.)
T: Right. Central Venous Pressure determines SV. The other determinant of SV is neurally controlled. What is the other determinant of SV? (Hinting strategy: point to the domain type of the missing variable.)
S: It is Inotropic State. (Correct.)
T: Right. Inotropic State is the other determinant of SV. SV is determined by Central Venous Pressure and Inotropic State. (The goal of the original question has been satisfied without telling the student the correct answer. The tutor can move to the next item on the agenda now.)

6.5 Planning Hints Using ATLAS

Freedman [1996a] modeled the behavior of our expert human tutors as a hierarchical structure of tutoring goals represented as schemata. She produced a reactive planner ATLAS to imitate this planning behavior [Freedman, 1999]. In addition to the capabilities in CST v. 2, her planner can back up to a higher goal and retry from any

level, not just at the goal where the student made an error. Furthermore, it can add interactive subplans to an existing plan, not just declarative material. The basic knowledge representation for the discourse planner is a sophisticated form of schema that allows static and dynamic preconditions, recursion and full unification. The most common form of schema encompasses three levels:

Tutoring strategy

Topics within the strategy

Text generation primitives (inform or elicit) for each topic

If the student answers a question incorrectly, ATLAS allows the system to backtrack at each of these levels.

CIRCSIM-Tutor v. 3, which is currently under development, is based on Freedman's ATLAS planner. I wrote a set of operators using ATLAS to simulate the behaviors that I have added to CIRCSIM-Tutor v. 2. In the following paragraphs, I give detailed examples. I first explain some of the high lever operators and then explain the operators related to hinting.

First the planner needs to decompose the goal of correcting-a-variable down to the primitive or turn goals. The following operator is a typical example that decides which method to choose to correct a variable according to the domain knowledge and tutoring history. It can be described as:

If the goal is to correct a variable,
and the variable is a neural variable,
and we did not try the neural-DLR method yet,
then we will choose the neural-DLR method.

This can be translated into the ATLAS operator

```
(def-operator correct-vbl-by-neural-DLR
    :goal (did-correct-vbl-1 ?vbl)
    :filter  ()
    :precond     ((is-neural ?vbl)
                  (not (have-tried-neural-DLR ?vbl)))
    :recipe      ((assert (have-tried-neural-DLR ?vbl))
                  (goal (did-neural-DLR ?vbl)))
    :temp        ((assert (w-method-is neural-DLR))
                  (assert (w-level-is method))))
```

The next operator defines the actual neural-DLR method.

If the goal is to try neural-DLR method
Then put the three step recipe into the tutoring agenda
1) tutor the mechanism that controls the variable
2) tutor the DR-information
3) tutor the value of that variable

And the operator is:

```
(def-operator do-neural-DLR
:goal (did-neural-DLR ?vbl)
:filter  ()
:precond ()
:recipe        ((goal (did-tutor mechanism ?vbl))
                (goal (did-tutor DR-info ?vbl))
                (goal (did-tutor vblvalue ?vbl)))
:temp          ())
```

The planner will continue to refine the goals in the above operator until it reaches the leaf level where the tutor asks a question (using the primitive T-elicit) or gives some information (using the primitive T-inform). If the tutor asks a question, the tutor will wait for the student's reply. If the student gives the correct answer, the tutor will continue on to the next item in the tutoring agenda. If the student gives an unexpected answer, the tutor will replan. The next two operators show how to replan after the student gives an unexpected answer.

The following example operator describes how to replan if the student shows some understanding of the topic. In this case, the student gives a near miss answer and the tutor tries to give a follow up hint.

```
If the student gives a near miss answer
        then invoke the operators that handle near miss answers
        (which will give a hint eventually).
```

The operator is:

```
(def-operator student-near-miss
:goal (did-reevaluate-agenda)
:filter  ()
:precond        ((i-input-type-is text)
                 (i-input-catg-is near-miss))
:recipe         ((goal (did-handle-near-miss)))
:temp           ())
```

The next operator describes how to change a plan at the tutoring method level if the student gives a totally incorrect answer.

If the student's answer is totally incorrect
then invoke the operators that handle incorrect answers
(which will try a new method eventually).

The operator is described as:

```
(def-operator student-incorrect-dont-know
:goal           (did-reevaluate-agenda)
:filter         ()
:precond        ((i-input-type-is text)
                 (i-input-catg-is ?x)
                 (p-summary-catg-is ?x bad))
:recipe         ((goal (did-handle-bad)))
:temp           ())
```

The next operator describes how to handle “partially correct” answers. The tutor will give hints accordingly.

```
(def-operator handle-partial-answer
  :goal (did-handle-partial-answer)
  :filter  ()
  :precond      ((w-variable-is ?vbl)
                 (not (is-neural ?vbl))
                 (w-topic-is determinants)
                 (w-method-is via-determinants)
                 (w-level-is topic)
                 (i-input-concept-is ?partial)
                 (e-get-missing-determinant ?vbl ?partial ?det)) ; Student Model
  :recipe       ((retract (no-error (w-partial-answer-is ?x)))
                 (assert (w-partial-answer-is ?partial))
                 (goal (did-acknowledge-right-part ?vbl ?partial)); acknowledgment
                 (goal (did-give-hint-of-missing-det ?vbl ?det)) ;;hints
                 (goal (did-ask-the-other-determinant ?vbl ?det)))
  :temp         ())
```

We will add more student model information and tutoring histories into the precondition as we build a complete set of operators.

One advantage of using the ATLAS planner is that it can back up at any level and at any time.

6.6 Comparison to Related Work

There are several tutoring systems that use hints as a tutoring tactic. Andes [Gertner et al., 1998] generates individual hints. It uses a Bayesian-network based student model to tailor its follow-up hints to the student's knowledge, and delivers them by using an associated sequence of hint templates for each goal and fact in its knowledge base. The Lisp tutor [Anderson et al., 1995] also generates hints from a sequence of hint templates. It uses model tracing techniques to detect that the student is not following the correct solution path. Sherlock II [Lesgold et al., 1992] generates a paragraph after the conclusion of the tutoring session that sometimes contains a hint.

In CIRCSIM-Tutor, we use heuristic rules to choose a hinting strategy based on the category of the student's answer, the tutorial plan, and the tutoring history. We then decide the content by searching the domain knowledge base to instantiate the strategy. So our hints are focused on both the student's needs and the current tutorial plan. By considering the current tutorial plan, the tutor can make sure the hints are coordinated with the tutorial plan and ensure conversational coherence while at the same time tailoring the content of the hint to the student's needs.

Merrill et al. [1992] compared the effectiveness of human tutors and intelligent tutoring systems. Their study indicated that a major reason that human tutors are more effective is that they let the students do most of the work in overcoming impasses, while at the same time providing as much assistance as necessary. Although in CIRCSIM-Tutor the tutor mainly leads the students in correcting the errors they have made in the prediction table, it is also important to let the student do as much as possible. By giving follow-up hints tailored to the student's answer rather than giving the correct answer, the

tutor provides necessary guidance to the student while promoting a more active style of learning.

In CIRCSIM-Tutor sometimes the student model can recognize the specific confusion of the student through its categorization of the student's answer. In that case, the hint is specifically related to the student's knowledge state. But even if the student model can not infer a deep understanding of the student's mental model, hinting is still more useful than just giving the answer for two reasons. In addition to giving the student a second chance to correct the error, the content of the hint may offer the student useful information for understanding the material.

6.7 Evaluation

An earlier version of CIRCSIM-Tutor which implemented a portion of the hinting model described above was used by 50 first-year students from Rush Medical College in November 1998. All of the students had already completed the regular lectures. They used the program for one hour. Twenty-four students worked in pairs at a computer and 26 students worked alone. We obtained a log file from each student or pair of students, giving a total of 38 log files.

The tables below describe our initial formative evaluation of this portion of the hinting model. In this experiment, CIRCSIM-Tutor asked approximately 1700 questions. In the course of tutoring 565 student errors, it generated 97 hints. Table 1 shows the effectiveness of hints for different student answer categories and Table 2 shows the effectiveness of each hinting strategy.

Table 1: Hints Used by Answer Category

Category of answer	No. of hints	No. of correct answers	% of correct ans.
Partially correct	55	41	75%
Near miss	12	9	75%
Incorrect	14	14	100%
Mixed	16	14	88%

Table 2: Effectiveness of Hints by Strategy

Category of hinting strategy	No. of hints	No. of correct ans.	% of correct ans.
Involving equations	2	2	100%
Evocative language	17	10	59%
Point to anatomical object	4	3	75%
Intermediate step	19	16	84%
Point to variable type	31	25	81%
Others	24	22	92%

In evaluating this performance it must be noted that in this experiment CIRCSIM-Tutor did not have a hint to give in all situations. In particular, hints for the incorrect answer category tended to occur on questions which had only a few possible answers, such as yes/no questions. Additionally, we believe that these questions were among the easier ones. As a result these hints tended to produce good results. Hints for the near miss

and partially correct answers were more likely to come from questions with a larger range of possible responses.

We have now implemented most of the possible hinting strategies for each answer category, and we hope to evaluate these hints in a later experiment. We are also looking forward to comparing the learning results between students who use the system with hints and without. Additionally, we are in the process of analyzing experimental data that will allow us to carry out a detailed analysis of student learning by comparing pretest and posttest results.

Another possible method for evaluating hints would be to let our human tutors compare the hints generated by CIRCSIM-Tutor to what they would like to say in the same situation. This method was used during the initial development of the system. We believe that this method of evaluation is important since the goal of this project is to simulate human tutoring behavior as closely as possible. Currently one of our expert tutors is working with the latest version of CIRCSIM-Tutor with this goal in mind.

The students were also positive about the quality of the hints and explanations (1.90 on a scale from 1= definitely YES to 5 = definitely NO, computed from the experiment survey form).

6.8 Summary of This Chapter

In this chapter I addressed how to systematically deliver pedagogically sound and conversationally coherent hints in a dialogue-based ITS, CIRCSIM-Tutor. The strategy involved categorizing student answers, and considering both tutoring plan and dialogue history. First, human tutoring transcripts are studied to identify human tutors' hinting

strategies and factors that might affect their choice of a hinting strategy. We then implemented these strategies in a real tutoring system as much as possible.

It is worthwhile to note that we are also planning to replace CIRCSIM-Tutor v. 2 by a completely rewritten v. 3 based on the work of Freedman and Evens [1996]. That project, currently in progress, will allow us to add more complex kinds of remediation since we will be able to use nested plans and delete agenda items that have become irrelevant. As I have shown in section 6.5, the operators are more flexible. We are looking forward to identifying uses for these new features. However, since the hinting algorithms described here are based on an actual corpus of tutoring transcripts, they will remain pedagogically valid and we intend to re-implement them in the new system.

Since most of the strategies isolated from the tutoring transcripts are not related to specific domain knowledge, we also expect to generalize them to other causal domains.

CHAPTER VII

BUILDING A STUDENT MODEL TO SUPPORT ADAPTIVE TUTORING

7.1 Applying the Knowledge from Transcript Analysis to Build a Student Model

In Chapter IV I described the transcript analysis used for student modeling and discussed the role of the student model. In Chapter V, I applied Machine Learning techniques to automatically find rules and useful patterns from the transcripts. In Chapter VI I discussed possible aspects of the student model in CST Version 2 that we need to improve. In this chapter I put most of those results together to build a student model for Version 3.

I address what to model and how to divide the student model into components in the context of the dialogue-based intelligent tutoring system, CIRCSIM-Tutor. I analyze the variety of decisions that the system needs to make and extract the information necessary to support these decisions. I then describe four distinct models that provide different aspects of this information, taking into consideration the nature of the domain and the constraints imposed by the tutoring system. At the end of this chapter, I discuss some general issues related to two questions and illustrated with examples from two CIRCSIM-Tutor experiments.

7.2 What to Build?

The content of student models in Intelligent Tutoring Systems (ITS) varies widely. Some student models are designed to recognize student plans or solution paths [Conati et al., 1997], some are designed to evaluate student performance or problem

solving skills [Katz et al., 1993], and some are designed to describe constraints that the student has violated [Mitrovic, 1998] [Ohlsson, 1992]. But there is one question that must be answered whenever we begin to build a new student model: what aspects of the student should we model in a specific intelligent tutoring system?

Many researchers argue that the main purpose of a student model in the context of intelligent tutoring is to guide pedagogical decision-making. There is little need for a description of the student's knowledge unless there is some way for the system to make use of, or react to, that description [Ohlsson, 1986] [Woolf and Murray, 1994]. This functional view of the student model prompts the following question: if the tutoring system has several modules or layers for making different decisions, should the student model be similarly structured into different models?

These two questions are critical to Circsim-Tutor, too. Especially after we analyze the tutoring setup and underlying tutoring principles, we will see their impacts on what we need to model and how they should be modeled in the student model.

The version we are currently building, CIRCSIM-Tutor version 3, is an ITS that helps medical students understand the negative feedback system that controls blood pressure. CIRCSIM-Tutor tutors by having students solve problems. The system presents the student with a description of a physiological change in a window located on the upper-right corner. For example, the patient might hemorrhage and lose a liter of blood. The tutor then asks the student to predict the effect of that change on seven important physiological parameters. The tutor conducts a dialogue with the student to correct the errors in the student's predictions. Depending on the tutoring protocol, this tutorial dialogue can occur after each erroneous prediction or after a larger hierarchical grouping such as a column or the entire table.

The tutoring setup and the underlying tutoring principles have a strong influence on the student model that we are building. For example:

- The goal of the tutoring system is to help students learn qualitative reasoning in a causal domain. In this domain, the causal relationships between parameters can be described at several levels. So how the student understands the causal links is a very important aspect of the modeling process. It is not enough to just model isolated domain facts.
- The students use a prediction table to solve the problem [Michael et al., 1992]. This table offers us the opportunity to get multiple inputs at the same time. So the student modeler can analyze all the predictions together to extract the domain principles that have been violated.
- The tutoring dialogue is plan based and the tutor takes control most of the time. This tutor-led tutoring setup makes it very difficult to trace the student's real solution path in contrast to other student-led systems that may easily build a trace model. Also the dialogue plan will rely more on the history data to maintain conversational coherence.
- The tutoring dialogue is handled as free text input and output. The ability of understanding free text will significantly limit the information we can get from the student.

Generally speaking, these features influence the information needed for tutoring decisions and constrain the possible information that the tutor can collect to build a student model.

7.3 Dividing the Student Model into Components

7.3.1 Decisions in CIRCSIM-Tutor. First I need to analyze what kinds of decisions have to be made in CIRCSIM-Tutor and what kind of information is needed to support these decisions. At the highest level of tutorial decision-making, CIRCSIM-Tutor has to make decisions such as choosing an appropriate problem for the student. At the same time, at the lowest level, as a dialogue-based ITS with free-text input and output, CIRCSIM-Tutor has to make decisions as detailed as how to choose an acknowledgment or a discourse marker. In a fully developed version of CIRCSIM-Tutor several modules will be needed to make different types of decisions: the curriculum planner, the tutorial planner, the turn planner, the discourse planner, and the surface generator. Each of these modules needs to communicate with the student model in order to make appropriate decisions. For some decisions, information about the student's overall performance is needed; while for other decisions, such as in discourse planning and surface generation, the most relevant information is the most recent statements of both student and tutor.

In the following sections I provide detailed analyses of some of the information needed by different modules in CIRCSIM-Tutor.

Adjusting the Curriculum. To decide on the right difficulty level for the next problem, we may need to know the overall performance of the student [Cho et al., 1999]. But, to determine the description of the problem presented to the student, we may also need to use detailed knowledge of how much the student knows about specific issues in physiology.

Switching Tutoring Protocols. We have three different protocols; each of them has different constraints about how the tutor assists the student in solving the problem [Khuwaja et al., 1995]. The protocol determines the number of related variables predicted by the student before the system starts to tutor the mistakes. Thus the decision to switch protocols may be governed by the student's performance on the causal relationships between variables.

Planning the Tutoring Dialogue. The detailed tutoring dialogue in CIRCSIM-Tutor is plan-based and controlled by the tutor most of the time, but new goals can emerge in an opportunistic fashion in response to unexpected student input. In CIRCSIM-Tutor, the student learns how to solve a problem in a causal domain. In this domain the causal relations between two parameters can be described at various levels of detail.

The tutor's decision about which way to teach is mostly determined by how the student has replied and in which order. If the student invokes an intermediate variable, we need to record where along the causal link that variable stands, so the tutor can continue from that point. Sometimes the tutor will invoke an intermediate variable by way of a hint. These intermediate causal relationships are not usually taught, but as soon as they are mentioned, it becomes important to teach them correctly. Thus the intermediate variables the student mentions, as well as their order, must be recorded in the student model.

7.3.2 Four Components of the Student Model. From the above analysis, we can see that different types and levels of student information are needed for different kinds of decisions. Some of the information is stored numerically and some symbolically, while

some must be kept as detailed history records. After analyzing the information needed by the system, based on the nature of the domain and the structure of tutorial decision-making, we designed four student models that offer different information. I describe the four components of the student model for CIRCSIM-Tutor in the next four sections.

7.4 Performance Model

The performance model computes an assessment of student competence, based on performance. To determine the levels of this evaluative model, we considered mainly the structure of tutorial planning of CIRCSIM-Tutor. In this system, the overall organization of a tutoring session is determined by the following structure:

- The student or the curriculum planner chooses problems for the student. Using medical terminology, each of the problems is called a *procedure*.
- The student then solves each procedure in three stages.
- In each stage, the student makes predictions about seven important physiological parameters and enters them in the prediction table.
- The tutor corrects incorrect predictions one by one and tutors the relationships between the concepts.

From this overall tutoring structure, it is natural to divide the evaluation of the student into four levels:

- Global assessment, overall measurement of the student's ability
- Procedure-level assessment, measurement for each problem the student is asked to solve
- Stage assessment, measurement for each of the three physiological stages in a problem
- Local assessment, measurement for each variable that has been tutored

The global assessment includes the following variables: prediction-table-performance (or problem-solving-skills), performance-to-tutoring, performance-to-hinting, performance-to-get-first-variables.

The procedure assessment includes the following variables: procedure-performance, performance-in-hinting-for-this-procedure and so on.

The stage assessment includes: prediction-table-performance-in-this-stage, performance-in-tutoring-in-this-stage, performance-in-hinting-in-this-stage.

The local assessment is the performance within each tutored concept. It includes performance-in-tutoring-in-this-concept and performance-in-hinting-in-this-concept. The local assessment is updated after each tutoring interaction, and the other assessments are calculated from the local assessment and the related performance in problem solving. The assessment model is currently based on a set of heuristics. The local assessment depends on fifteen different patterns found in the student's answers. A statistical model or an inference model could be used to propagate the local assessment to each of the upper level assessments. The local assessments range between 0 and 1. For each answer pattern, an updating weight is assigned and the local performance is calculated based on

Equations 1 and 2, which are used to update the probabilities that the student knows something based on the student answer pattern.

If the category of the student's answer has any correct part, we will calculate the local assessment using equation 1.

Equation 1:

$$pKnown_t = pKnown_{t-1}(1 + wCategory)(1 - pGuess) \quad (1)$$

If the category of the student's answer does not have any correct part, we will calculate the local assessment using equation 2.

Equation 2:

$$pKnown_t = pKnwon_{t-1}(1 - wCategory)(1 + pSlip) \quad (2)$$

pSlip estimates the probability that the student will make an error even though the concept has been mastered (i.e., the student "slips"); *pGuess* is the probability that the student will give the correct answer even though the skill has not been mastered (i.e. the student guesses correctly). *WCategory* is the weight assigned to the student's answer. $pKnown_{t-1}$ is the probability that the student knows the tutoring concept before the calculation. $pKnown_t$ is the updated probability of how much the student understands the tutored concept.

7.5 Student Reply History

A student reply history is attached to each concept that CIRCSIM-Tutor teaches. To decide what to record in this model, we mainly consider the nature of the domain. In our domain, a causal relation between two parameters can be described at various levels of detail. The history model records which part of the causal chain has been covered as well as a history of student answers about that chain. For each student answer we also record its classification. Student answers are classified into eight categories to help the tutor decide on a response strategy [Zhou et al., 1999a]. For example, a near miss answer is one that is pedagogically useful but not the desired answer. Below are the categories that we use to classify students' answers:

1. Correct
2. Partially correct answer, i.e., some part of the answer is correct and the rest is incorrect
3. Near miss answer, which is pedagogically useful but not the desired answer [Glass, 1997]
4. "I don't know" answer
5. "Grain of truth" answer, where the student gives an incorrect answer, but also indicates a partially correct understanding of the problem [Woolf, 1984]
6. Misconception, a common confusion or piece of false knowledge about the concept being tutored

7. Other incorrect answers

8. Mixed answers, i.e., a combination of answers from other categories

Here are some example of the categories extracted from the experiments.

Partial answer:

T> What are the determinants of Stroke Volume?
S> Inotropic State. (Expected two parameters, but get one instead.)
T> Right, Inotropic State is one determinant of Stroke Volume. The other determinant represents preload. What is the other determinant of Stroke Volume?

Near miss answer:

T> What are the determinants of Stroke Volume?
S> preload (This is true, but preload is not one of the variables on the student's worksheet.)
T> Right, preload determines Stroke Volume. What parameter in the prediction table represents preload?

Misconception:

T> By what mechanism is Inotropic State controlled?
S> It is filling. (The tutor's response explained why this is a misconception.)
T> Well, you are confusing the Frank-Starling effect (increased filling gives greater output) and contractility. Inotropic State (contractility) is the force of contraction at any given filling or at any end diastolic volume. An increase in sympathetic stimulation of the myocardium increases the Inotropic State of the muscle. So, it is neurally controlled

Grain of truth answer:

T> What are the determinants of Stroke Volume?
S> Heart Rate and Cardiac Output. (i.e., SV = HR * CO. But this is not what the tutor asked for.)
T> Well, it is true that CO = SV * HR. But what I was asking is what determines how much blood is ejected from the heart each time it beats (the SV)?

It is important to point out that this category is the result of transcript analysis, it is not generated randomly.

So, after each student answer, the student modeler will classify the student's answer into a specific category. The eight categories are ideal situations while in practice, one student answer may actually have several parts and each of them can be classified in one of the eight categories. So the actual student answer record is a list of all the possible categories and the related part in the answer. The student modeler will go through the student answer, pick up each category appeared in the answer and assemble them in the record.

Here is a sample frame that records the student's reply history, including a detailed history of answers in a different tutoring plan context, such as the intermediate causal links.

```
(sm-model *sv-overlay*
      (sm-model-type        variable
       eval-value           1       ;;how much the student know about this var
       determinants         nil     ;;answer history for determinants question
       value                nil     ;;answer history for value question
       actual-determinant   nil     ;;answer history for actual-det question
       relation             nil     ;;answer history for relation question
       other-task           nil     ;;answer history for other tasks
       tutoring-history     nil     ;;tutoring history of this variable
       refer-to             *sv*))
```

7.6 Student Solution Record

This model records how many errors the student made while solving the problem, along with a detailed record of each error. In CIRCSIM-Tutor, we try to help the student to solve a problem in a logical order. So, it is important to compare the order of the student's predictions to any appropriate logical orders (which are not unique) and to analyze any inappropriate sequences on the part of the student. This model is similar to the idea of the constraint-based student model [Ohlsson, 1992]. But we also group all similar errors and analyze all possible misconceptions. For example, we will group all neural variables together, so the planner may plan to tutor them together too.

Hume [1995] proposed an error pattern model to analyze the student's solution. In Hume's definition, an error pattern represents a physiological concept about which the student has appeared to reason incorrectly.

I prefer to use a constraint violation approach because it is not possible to infer which logical path the student used to make predictions in CST's tutoring setup. So, basically we can not say that the student does not understand one specific causal link, he/she may simply not use it but instead use an unrelated link. But we can safely say that the link in the current prediction has been violated and we need to tutor it. So, we emphasize what we can model.

The following list contains some constraints that have to be satisfied when a student solves a problem. A violation of these constraints can be detected by simply comparing the student's solution and the correct solution.

1. Direct causal relationship constraints:

- CVP to SV when CVP is the main determinant of SV.
- IS to SV when IS is the main determinant of SV.
- SV to CO when SV is the main determinant of CO.
- HR to CO when HR is the main determinant of CO.
- CO to MAP when CO is the main determinant of MAP.
- TPR to MAP when TPR is the main determinant of MAP.

2. Inverse causal relationship constraints:

- CO to CVP when CO changes before CVP.
- MAP to SV when MAP is the main determinant of SV.
- MAP in DR phase to IS in RR phase.
- MAP in DR phase to HR in RR phase.
- MAP in DR phase to TPR in RR phase.

3. Equation constraints:

- CO = HR * SV.
- MAP = CO * TPR.

4. Others:

- Neural variables remain unchanged in the DR stage if they are not the primary variable.
- MAP in SS makes the same qualitative change as MAP in DR.
- And so on.

By comparing the student's solution and the possible problem solving logic (the causal links), we can compute the violated constraints and form a detailed solution record. The inference rules are:

1. Find all incorrect parameters.
2. Start from the source of the causal link, check each link, record the violated links (which will be tutored by the tutoring dialogue).
3. Group similar violations.

Table 3 shows an example of student's prediction extracted from an experiment.

Table 3: An Example of Student Predictions for the DR Stage.

CV Variable	Student's Prediction	Correct Value	Student Error
IS	+	0	X
CVP	-	-	
SV	+	-	X
HR	0	0	
CO	+	-	X
TPR	+	0	X
MAP	+	-	X

The correct reasoning logic is:

Neural variables IS, HR, and TPR do not change in the DR stage.

CVP is the primary variable.

CVP $\xrightarrow{+}$ SV $\xrightarrow{+}$ CO $\xrightarrow{+}$ MAP

So, the student violated 2 constraints although 5 incorrect predictions occurred. The violated constraints are:

Neural variables do not change in the DR stage if they are not the primary variable.

The direct causal relationship constraint: CVP to SV when CVP is the main determinant of SV.

So in the student solution record we will record these two violations and the tutor may try to tutor them in detail. Even though the student made incorrect predictions of CO and MAP, the underlying relationship of SV to CO and CO to MAP are correct and the tutorial planner may chose to tutor them by simply moving forward after correcting SV. For example, the following dialogue is possible:

T:> (Detailed tutoring of direct causal relationship from CVP to SV.) So, what's value of SV now?
S:> It's going down.
T:> Right. And if SV goes down, what will happen to CO?
S:> It goes down too.
T:> And MAP?
S:> Down.
T:> Great!

So, we can see that by analyzing the violated constraints, the student model can offer useful information to the dialogue planner. This solution record is a preliminary analysis of the student's prediction. If from the dialogue the tutor finds new evidence that the student does have some difficulty in understanding some causal relationships that were originally correct, the tutor can replan its dialogue to specifically teach that relationship.

It is important to separate the incorrect parameters and the incorrect causal links, because the student may predict an incorrect value of a variable that is in the early path of the causal link. In this case the incorrect value may propagate to the following causal links if the student understands the rest of the causal link correctly. So the student may get five variables wrong, but actually only one causal link is incorrect.

It is necessary to keep this model simple, because the student's solution path cannot be inferred from his or her predictions in the prediction table. The prediction table only roughly reflects the student's ability to solve a problem logically. If we provide a way to force students to draw their solutions, then we may need to adopt another indicator of the students' problem solving skill.

7.7 Tutoring History Model

This model includes both the plan history and the discourse history. It does not contain direct knowledge of the student; rather, it is a record of what the system has done. We consider it as an extended part of the student model because it is important for tutorial decision-making [Freedman, 1996a] and it also implicitly reflects how much the student has done and how well.

The tutoring history is mostly constrained by the planning mechanism. Freedman [Freedman, 1996a] modeled CIRCSIM-Tutor's tutorial plan as a hierarchical structure of tutoring goals implemented as schemata. The tutoring hierarchy includes three levels: tutoring strategies, topics within the strategy, and text generation primitives (*inform* or *elicit*) for each topic. So for each procedure we record the tutoring history according to this hierarchy.

The history data can be further divided into dialogue history and planning history. Dialogue history records the detailed hints, questions, and other information in logic forms [Kim, 1997]. The planning history records what kind of plan we have tried, their success and so on. XML-based or SGML-based notation will be good for this purpose.

The four models are stored separately for decision making, evaluation and possible comparison. The performance model is stored as a set of numbers, while the other models are stored as lists and frames.

7.8 Further Decoupling Information in Student Model

As we have discussed, there are many data included in the student model. Some are domain dependent, some are tutoring method dependent, and some are more generic. For example, the understanding of a specific concept is domain dependent, but may not be tutoring method dependent; the plan history data is planner dependent but may not be domain dependent; while the general preference for hinting may not depend on either domain or tutoring method. When we try to build cooperative tutoring systems, decoupling from this point of view may become critical.

7.9 Tutoring Rules That Use Student Models

We use the category of the student answer and the tutor's assessment of the student to roughly decide the response strategy in Circsim-Tutor Version 2. In Circsim-Tutor Version 3, we also use the tutoring history, and other discourse context to decide the response strategy. Here are the rules that we used to select the response strategy in Version 2.

If the category of student answer is correct,
    acknowledge it and go to the next step in the original plan;
If the category of student answer is partially correct,
    acknowledge the correct part, give a hint related to the missing part;
If the category of student answer is near-miss,
    Ask a leading question from the near-miss answer;
If the category of student answer is a "grain of truth answer",
    Acknowledge the "grain of truth", ask the question in different words;
If the category of student answer is misconception,
    Explain the difference and give the correct answer;
If the category of student answer is other incorrect answers or "I don't know answer",
    If the assessment of the student is larger than 0.5,
        Give hint;
    Else {less or equal to 0.5}
        Give correct answer;
If the category of student answer is a mixed answer,
    If there is a near miss parameter,
Acknowledge any correct part in the answer, and then ask follow up questions from the near-miss part;
    Else
Acknowledge any correct part and then give the correct answer

So, we use the student's answer category to decide the response strategy, but also consider the student's performance when the category is incorrect or "I don't know."

Here are some examples that show how we use the student model in the plan operators of CST Version 3.

Using the answer category:

```
(def-operator student-partial-answer
  :goal (did-reevaluate-agenda)
  :filter  ()
  :precond       ((i-input-type-is text)
                  (i-input-catg-is partial-answer)) ;; From student model
  :recipe        ((goal (did-handle-partial-answer)))
  :temp    ())
```

Using the performance assessment:

```
(def-operator nonneural-partial-determinants
  :goal (did-handle-near-miss)
  :filter  ()
  :precond ((w-variable-is ?nonneural-to)
              (w-student-performance-is ?performance)  ;;;;from student model
              (is-good-performance ?performance)          ;;check student model
.:recipe    ((goal    (did-near-miss-specific-nonneural    ?vbl-from    ?nonneural-to
          ?partial)))
  :temp    ())
```

7.10 Making Consistent Decisions

When there are several student models available, each model may suggest different decisions. So, how to make consistent decisions is very important. For example, in CIRCSIM-Tutor, the overall performance of the student may indicate that the student is struggling and the tutor should give more direct information to prevent the student from becoming too lost; on the other hand, the recent student answer history may indicate that the student has some understanding of the current topic and so the tutor should encourage the student to do as much as possible by him or herself. Given this conflict, it is important to make consistent decisions. In CIRCSIM-Tutor, we have simple rules to decide which model is more important. For example, if the recent history indicates positive evidence that the student is doing well on a topic, then the tutor will always try to encourage the student to do more, even when the overall performance of the student is poor. This rule expresses our human tutors' belief that the tutor should encourage the student to do as much as possible. So, the student will always get a second chance after an error if he or she evinces any understanding of the topic at all. The study in [Merrill et al., 1992] also indicated that human tutors let the students do most of the work in overcoming impasses, while at the same time providing as much assistance as necessary.

7.11 Using Multiple Student Models to Support Tutoring Decisions

In this section I describe how I use different student models to support decisions in CIRCSIM-Tutor. Specifically, I will discuss how to use different student models in the planning of tutorial dialogue designed to remediate erroneous student predictions.

From the student's errors and the tutoring history the planner will choose the high level tutoring methods. The next step is to plan detailed tutoring dialogue by using the answer category, the student's performance, and the student's reply history.

From the decision-making analysis, we have seen that the tutor has a variety of ways to tutor each causal relationship and the student's reply is a key to decide which way to teach. In Dialogue 1, the machine tutor is trying to tutor the causal link between Stroke Volume (SV) and Central Venous Pressure (CVP).

Dialogue 1:

T: What is the determinant of Stroke Volume?
S: It is EDP [end diastolic pressure]. *(near miss)*
T: And what parameter in the prediction table determines EDP?
S: EDV [end diastolic volume]. *(near miss in the wrong direction)*
T: EDV comes from EDP. Again, what determines EDP?

The tutor begins by asking the student a question. The student may give different answers. According to our classification scheme, the student model will classify the answer into one of the eight categories. The tutorial planner can then decide what to do next. If the student's answer is totally incorrect, then the planner may use the student's performance to help decide further between giving the correct answer or giving a hint. In this example the student model classifies the student's answer as a near miss and puts it into the student reply history along with the detailed content of the answer. Based on this category information, the planner then decides to ask a question based on this near miss in order to help the student find the expected answer. Now the student can continue. Again, the student model will classify the student's answer and put it into the record and the planner will take action based on the classification and the reply history. The

student's next answer returns to a link that has already been tutored, so the tutor acknowledges the relationship and asks a follow up question based on the last correct link instead of the student's latest response.

7.12 Experiments with CIRCSIM-Tutor

Two experiments with CIRCSIM-Tutor have been conducted with first-year students from Rush Medical College. In April 1998, 22 students used an earlier version of CIRCSIM-Tutor, developed in Lisp by Woo and others [Woo et al., 1991]. In that version the student model is a simple overlay model of the prediction table. In November 1998, 50 students used a new version of CIRCSIM-Tutor, which includes a new input understander, an updated tutorial planner, and a new student model. The new student model includes prototypes of the four models described earlier: the student performance evaluation model, the tutoring history, the student reply history, and the student solution record. A log file was produced for each student. Comparing the log files from these two experiments, we can see that the new student model improves tutoring quality even though the four implemented models are very simple. From studying the log files, we see that the new student models were important in the following ways:

- The evaluation of the student's performance helped decide when to give hints and when to give answers when the student cannot answer a question.
- The student reply history allows the tutor to end a dialogue and return to the original plan when the student could not continue along a causal link.

- The category of student answer, another aspect of the student reply history, is effective in helping to decide on different retry strategies. It recognizes near misses and other categories of answers that were once treated as totally incorrect answers.
- The tutoring history prevented the tutor from giving the same hint repeatedly.

For example, in Dialogue 2, which is extracted from the April experiment, the tutor responds to an unexpected answer by telling the student he or she is wrong, even though strictly speaking he or she is correct, since the student model does not have enough categories of student answers.

Dialogue 2:

T: What is the determinant of Stroke Volume?
S: It is EDP.
T: No. Stroke Volume is not determined by EDP. Stroke Volume is determined by Central Venous Pressure and Isotropic State.

On the other hand, in Dialogue 3, which is extracted from the November experiment, the tutor can generate an appropriate hint according to the detailed categories of student answers.

Dialogue 3:

T: What is the determinant of Stroke Volume?
S: It is EDP. (near miss answer)
T: And what parameter in the prediction table determines EDP?

Since Version 3 is currently under development, I used simulations to evaluate part of my student model, mostly the performance model. By extracting the student's response data from the November, 1998 experiments and applying the new algorithms to the data, the simulations showed changes in the student's performance.

7.13. Discussion

Determining what information to include in the student model is a system-dependent task. There are several constraints we have to consider. The first constraint is the nature of the domain: Is it a quantitative domain or a qualitative domain? Is the focus on causal reasoning as in our system or on pure facts? Another constraint is the structure of a tutorial session. Basically, we need to consider how the system interacts with the student: how the system presents the problem to the student, how much the system can observe about the student's solution, and whether the system uses natural language or not. A third constraint is the tutorial decisions that the system needs to make. Does the system need to plan the curriculum, to switch between tutoring protocols, to plan tutoring dialogue, or just give simple feedback without multi-step plans.

Another important issue is how to evaluate the student model. Since we have different student models, we may adopt different evaluation methods for them. We may even adopt different methods to evaluate the history models and the performance models.

7.14 Summary of This Chapter

In this chapter, I discussed how to build a student model composed of different sub-models by considering the actual demands of different decision-making components of the system. I discussed how to determine the knowledge required in each model by considering three types of constraints imposed by the system: the nature of the domain, the structure of the tutoring session, and the tutorial decisions that the system needs to make. I compared the quality of tutorial dialogues based on two versions of the student model in CIRCSIM-Tutor: one with only a simple overlay and one with prototypes of the four models.

There is another advantage of component based student model — separate the domain specific data from domain independent data. For example, the student's ability to follow up hints maybe domain independent while the history data are mostly domain dependent. So, in the future, when we start building tutoring agents that can communicate with each other, this separation will be critical, since domain dependent information will be useful only to agents that work in the same domain.

CHAPTER VIII

DISTRIBUTED ITS — THE FUTURE

In this chapter I discuss the possibility of building Web-based distributed ITSs. I explain the motivations and the possible approaches. Specifically, I discuss the opportunity of reengineering existing Computer Aided Instruction (CAI) systems and ITSs in the Circsim-Tutor project by taking advantage of the current studies on student modeling and Web-related techniques and component-based architectures. The ultimate goal is to build reusable tutoring components, from which we can build distributed intelligent systems that can cooperate with each other easily.

8.1 Motivation for This New Research Direction

8.1.1 How to Make CAI Systems More Intelligent. CAI systems have been developed for a long time and there are numerous CAI systems out there. There may exist several CAI systems for the same domain, but each of them may use different tutoring methods or strategies. For example, between our group and the Rush Medical College, we have text-based, diagram-based, and simulation-based CAI systems for the same domain or similar domains. Most of them have been used for a long time, but there are some problems with these systems:

a. They are often quite old (some are DOS programs) and not very user friendly. So, most of them need reengineering.

b. Most CAI systems are not aware of each other and cannot cooperate with each other even though they operate in the same machine.

c. CAI systems do not usually maintain any student model.

So one motivation is to reengineer the old CAI systems, to make them more intelligent, friendlier, and more able to cooperate with each other.

8.1.2 Building Web-Enabled ITS. The Internet is the most efficient medium to reach more and more students. The advances in Web-related technology in the commercial world offers lots of opportunities for tutoring systems to take the advantage of the large distributed computing world. Web-based tutoring applications may offer the following advantages:

a. The Web offers the opportunity to reach more varied groups of students.

b. The drop and click user interface is friendlier.

c. It is easy to distribute and collect information.

d. Web application offers the opportunity to keep track of student's moves. Since Web-based applications are naturally distributed systems, any moves of the student need to be sent to the server to proceed.

e. The n-tier component-based Web application architecture offers a great opportunity to encapsulate and reuse tutoring logic.

f. The n-tier architecture also offers the possibility of centralizing or communicating student information and the opportunity to cooperate.

Several researchers in the ITS area already explored some of the opportunities to use the Web as the main medium to deliver tutoring systems [Brusilovsky, 1998, 1999].

8.1.3. Studies in Student Modeling. Student models are considered to be the key part to make CAI systems become Intelligent Tutoring Systems (ITS). The content of student models in Intelligent Tutoring Systems (ITS) varies widely. Some student models are designed to recognize student plans or solution paths [Conati et al., 1997], some are designed to evaluate student performance or problem solving skills [Katz et al., 1993], and some are created for describing constraints that the student has violated [Mitrovic, 1998] [Ohlsson, 1992]. My research in student modeling has shown the possibility of dividing the student models into different components and the potential to use different components to support different decisions in a dialogue-based ITS [Zhou and Evens, 1999]. One natural extension of this research is to study how to divide the student model into different modules to facilitate the possibility of sharing student information among different tutoring systems and therefore achieve further customized tutoring.

One key issue of student modeling in distributed ITS is to separate domain dependent from domain independent information and to separate tutoring method dependent from tutoring method independent information in the student model. The component-based student model can be extended naturally to consider these types of information separately.

8.2 Challenging Issues

In order to integrate the current tutoring logic, the studies of student models, and the possible advantages of current Web technologies, there are several issues that need to be considered:

1) How to make the CAI system more intelligent?

One possible solution is to build a simplified template student model and use this model to select a tutoring template. Since Web-based systems will be able to catch student moves naturally, it is possible to build simple student models when CAI systems are transferred to Web-based systems.

2) How to reuse them and make them web-enabled?

There are several component architectures that focus on how to reuse computer software. Among them the Common Object Request Broker Architecture (CORBA) and the Component Object Model (COM) are the two most used architectures. Both of them use Interface Definition Language (IDL) to wrap existing code if possible. CORBA's IDL for language binding makes it quite easier to reuse legacy systems. So, by using component based architecture, we should be able to reuse traditional CAI systems. Also, both CORBA objects and COM components can be used in web-based applications, so component-based architecture should be a good answer.

3) How to make them aware of each other?

To solve this issue we may borrow some network development experience. We can either use a centralized management system or have a broadcasting service.

4) How to make them cooperate with each other?

This may be the hardest problem in building truly cooperative distributed ITS. For tutoring systems that working on a same domain, they can cooperate by sharing the understanding of the domain. Some researchers have already initiated studies on semantics and ontology interoperability [Koedinger et al., 1998][Macrelle and

Desmoulins, 1998]. For totally unrelated domains, ITSs have to find common information such as whether the student likes hints, likes help, and so on.

5) What type of architecture will most suitable for cooperative tutoring systems?

We can consider both centralized and peer-to-peer architecture. But we may have to wait till we understand more about different ITSs.

One solution that I think can be used to improve existing CAI systems is to build web applications for existing CAI systems and collect simple student model information in the server, in databases or in simple text files. Thus the CAI system can customize its contents from this simple student model. At the same time, by retrieving from the same database, other CAI systems can get information about users who used this CAI system and use this information to customize their tutoring contents. With the promise of component-based techniques, current ITS systems can also be easily wrapped to form components that can be reused in Web applications.

Next, another CAI system can be added and we can make the two systems share student information, which is stored in a central data store. We can also build tools to enable human tutors to view student states and create authoring tools.

A further step can be studies on distributed student models by formatting information in XML or a special student model mark up language.

In particular, to study distributed ITSs in the Circsim-Tutor project, we can build a three-tier web-based application for an existing CAI system – Circsim [Michael and Rovick, 1986]. Circsim is a CAI program written in Basic that has been used by hundreds of students at Rush Medical College. The domain of this CAI system is Cardiac Physiology. To build the Web enabled tutoring system, we can reuse most of the tutoring

logic in the original CAI systems, and at the same time record student information in a database. Then by updating the tutoring logic, we can upgrade the CAI system to use the student information and customize its tutoring. And we can continue this process by adding a diagram-based CAI and other tools too.

CHAPTER IX

CONCLUSION

9.1 Summary

This thesis outlines a new and radically different student model for the Circsim-Tutor project. By consulting this model, our machine tutor can adapt its tutorial dialogue to the student's needs in a more flexible fashion. This thesis also presents a study of human tutoring transcripts using Machine Learning techniques and describes a new model of hinting based on transcript analysis.

This research has evolved in several steps. Below I summarize what I have done.

1. Analyzed the advantages and disadvantages of the model in Circsim-Tutor Version 2.

In this step I defined the problems I needed to solve by analyzing the limitations in Circsim-Tutor Version 2 carefully, especially the student model.

2. Analyzed human tutoring transcripts.

In this step I tried to find the possible solutions for the problems I discovered in the first step, i.e., what kind of student model is used by our human tutors to support their decisions, especially what kind of information in the student's answer do they focus on in making each type of decision.

3. Applied Machine Learning techniques to find tutoring rules.

In this step I applied Machine Learning techniques to find detailed tutoring rules and gained more knowledge about what is useful in the student model. This study verified some of the hypotheses in Step 2.

4. Updated Circsim-Tutor Version 2.

In this step, I combined these research and experimental results and updated the original Version 2. As a result our new version can handle a number of different kinds of student answers and choose different response strategies when a student cannot answer a question. In this process I also discovered some new problems that we need to handle in the future; especially we need to recognize more categories of student answers.

This updated version has been used by students from Rush Medical College in April 30, 1998, November 16, 1998, and November 1999.

5. Categorized student answer and analyzed the transcripts further.

In this step I analyzed more transcripts in order to solve the new problems I found in Step 4. I collected more specific low level tutoring rules, categorized student answers in more detail, and identified more misconceptions. Now we have classified student answers into eight categories.

6. Designed and developed a model of hinting.

I built a new model of hinting for Circsim-Tutor. The new hinting model chooses hinting strategies by considering domain knowledge, the type of error made by the student, the focus of the tutor's question, and the conversational history.

7. Built a student model for Circsim-Tutor Version 3.

I ported the student model from Circsim-Tutor Version 2 to Version 3 and updated it by integrating all the results from the above steps. It includes different components to support different levels of decisions. The performance model also includes four levels of measurements.

8. Evaluated the new student model.

I evaluated part of the student model using simulation.

9. Studied the problems of extending CST to a distributed tutoring system.

I extended my study to multi-tier distributed intelligent tutoring systems.

9.2 Significance

A detailed student model in an Intelligent Tutoring System is essential if the system wants to tailor its tutoring to student needs. In this research I built a sophisticated

student model for Circsim-Tutor. By carrying out this goal step by step, my research has shown its importance for this project.

First my research has improved the ability of Circsim-Tutor Version 2. I have added a new student model component and updated some other modules, the new version can simulate some sophisticated human tutoring behaviors, such as hinting, under a simple planning framework. It can handle different types of answers: near misses, partially correct answers, misconceptions, “grain of truth answers,” and so on.

The hinting model that I have developed for Circsim-Tutor Version 2 can systematically deliver pedagogically sound and conversationally coherent hints in the tutoring dialogue by considering the domain knowledge, the type of error made by the student, the focus of the tutor’s question, and the conversational history.

My research is significant for the new version of Circsim-Tutor too. In Circsim-Tutor Version 3, we will have a more sophisticated planner, which will plan on different levels and produce different aspects of the tutorial dialogue. This planner will need much more information from the student model than Version 2 uses. But many of the ideas that I have implemented in Version 2 can be adapted to Version 3 and I have written some prototype operators for the new version. The transcript analysis and the Machine Learning study have also given us much insight into the necessary student model that can support human like dialogue. The student model I have built includes four different components that can support decisions at different planning levels.

My initial study of distributed ITS proposed a new research direction in CST and the student model I have built can be extended to a distributed system easily.

In addition, since this approach is designed to build a student model for supporting natural language dialogue, it will influence other projects that are going to build a user model for adaptive natural language interfaces.

BIBLIOGRAPHY

Anderson, J. R., and Reiser, B., 1985. The Lisp Tutor. Byte, Vol. 10, No. 4, pp. 159-175.

Anderson, J. R., Corbett, A., Koedinger, K., and Pelletier, R., 1995. Cognitive Tutors: Lessons Learned. Journal of the Learning Sciences 4(2), pp. 167-207.

Brandle, S., 1998. Using Joint Actions to Explain Acknowledgments in Tutorial Discourse: Application to Intelligent Tutoring Systems. Ph.D., Thesis. Illinois Institute of Technology, Chicago, IL.

Brandle, S. and Evens M. W., 1997. Acknowledgments in Tutorial Dialogue. In Eugene Santos, ed., Eighth Midwest AI and Cognitive Science Conference, Dayton, OH, 1997, Menlo Park, CA: AAAI Press, AAAI Technical Report CF-97-01, pp. 13-17.

Brusilovsky, P., 1998. Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies. In: Proceedings of Workshop "WWW-based Tutoring" at 4th International Conference on Intelligent Tutoring Systems (ITS'98), San Antonio, TX.

Brusilovsky, P., 1999. Adaptive and Intelligent Technologies for Web-based Education. In C. Rollinger and C. Peylo (eds.), Special Issue on Intelligent Systems and Teleteaching, Künstliche Intelligenz, Vol. 4, pp. 19-25.

Cho, B., Michael, J. A., Rovick, A. A., and Evens, M. W., 1999. A Curriculum Planning Model for an Intelligent Tutoring System. Proceedings of the 12th Florida Artificial Intelligence Symposium (FLAIRS–99), Orlando, Florida, pp. 197–201.

Clancey, W., 1986. Qualitative Student Models. Annual Review of Computer Science, Vol. 1, pp. 381-450.

Clark, H., 1996. Using Language. Cambridge University Press, Cambridge, England.

Conati, C., Gertner, A., VanLehn, K., and Druzdzel, M., 1997. On-line Student Modeling for Coached Problem Solving Using Bayesian Networks. Proceedings of UM-97, Sixth International Conference on User Modeling, pp. 231-242

Corbett, A. T., Anderson, J. R., and Patterson E.G., 1990. Student Modeling and Tutoring Flexibility in the Lisp Intelligent Tutoring System. Frasson, C. and Gauthier G., Eds., Intelligent Tutoring Systems: at the Crossroads of Artificial Intelligence and Education, pp. 83-106, Norwood, N.J.: Ablex.

Corbett, A. T. and Anderson, J. R., 1992. The LISP Intelligent Tutoring System: Research in Skill Acquisition. Larkin, J., Chabay, R., Scheftic, C., Eds., Computer Assisted Instruction and Intelligent Tutoring Systems: Establishing Communication and Collaboration, pp. 73-109. Hillsdale, NJ: Erlbaum.

Freedman, R., 1996a. Interaction of Discourse Planning, Instructional Planning, and Dialogue Management in an Interactive Tutoring System. Ph.D., Thesis. Northwestern University, Evanston, IL.

Freedman, R., 1996b Using a Text Planner to Model the Behavior of Human Tutors in an ITS. Seventh Midwest AI and Cognitive Science Conference, Bloomington, IN. URL: http://www.cs.indiana.edu/event/maics96/Proceedings/Fredman/ freedman.html or freedman.ps.

Freedman, R., 1996c. Using Tutoring Patterns to Generate More Cohesive Text. Proceedings of the 2nd International Conference on the Learning Sciences, Evanston, IL, 1996. pp. 75-82.

Freedman, R., 1999. Atlas: A Plan Manager for Mixed-Initiative, Multimodal Dialogue. AAAI-1999 Workshop on Mixed-Initiative Intelligence, Orlando, FL.

Freedman, R. and Evens, M. W., 1996. Generating and Revising Hierarchical Multi-Turn Text Plans in an ITS. Intelligent Tutoring Systems: Third International Conference ITS '96, Montreal, 1996, pp. 632-640.

Freedman, R., Brandle, S., Glass, M., Kim, J., Zhou, Y., Evens, M.W., 1998a. Content Planning as the Basis for an Intelligent Tutoring System (System Demonstration). Proceedings of the Ninth International Workshop on Natural Language Generation (INLG-9), Niagara-on-the-Lake, Ontario, 1998, pp. 280-283.

Freedman, R., Zhou, Y., Glass, M., Kim, J., Evens, M.W., 1998b. Using Rule Induction to Assist in Rule Construction for a Natural-Language Based Intelligent Tutoring System. Twentieth Annual Conference of the Cognitive Science Society, Madison, 1998, pp. 362-367.

Freedman,. R., Zhou, Y., Glass, M., Kim, J., Evens, M.W., 1998c. SGML-Based Markup as a Step toward Improving Knowledge Acquisition for Text Generation. AAAI 1998 Spring Symposium: Applying Machine Learning to Discourse Processing, Stanford University, pp. 114-117.

Gertner, A.S., Conati, C., and VanLehn, K., 1998. Procedural Help in Andes: Generating Hints Using a Bayesian Network Student Model. Proceedings of the 15th National Conference on Artificial Intelligence, Madison, Wisconsin, 1998, pp. 106-111.

Glass, M., 1997. Some Phenomena Handled by the CIRCSIM-Tutor Version 3 Input Understander. Proceedings of the Tenth Florida Artificial Intelligence Research Symposium, Daytona Beach, FL, 1997, pp. 21-25.

Glass, M., 1999. Broadening Input Understanding in an Intelligent Tutoring System. Ph.D. diss., Dept. of CSAM, Illinois Institute of Technology, Chicago, IL.

Hume, G. D., 1995. Using Student Modelling to Determine When and How to Hint in an Intelligent Tutoring System. Ph.D., Thesis. Illinois Institute of Technology, Chicago, IL.

Hume, G. D., Michael, J., Rovick, A., and Evens, M.W., 1996a. Student Responses and Follow Up Tutorial Tactics in an ITS. Proceedings of the Ninth Florida Artificial Intelligence Research Symposium (Flairs '96), Key West, FL., pp. 168-172.

Hume, G. D., Michael, J., Rovick, A., and Evens, M.W., 1996b. Hinting as a Tactic in One-On-One Tutoring. Journal of the Learning Sciences, Vol. 5, No. 1, pp. 23-47.

Hume, G. D., Michael, J., Rovick, A., and Evens, M.W., 1996c. The Use of Hints by Human and Computer Tutors: the Consequences of the Tutoring Protocol. Proceedings of the 2nd International Conference on the Learning Sciences, Evanston, IL.

Katz, S., Lesgold, A., Eggan, G., Gordin, M., 1992. Approaches to Student Modeling in the Sherlock Tutors. Andre, E., Cohen, R., Graf, W., Kass, B., Paris, C., Wahlster, W., Eds. Proceedings of the Third International Workshop on User Modeling, pp. 205-230. Dagstuhl Castle, Germany: International Conference and Research Center for Computer Science (IBFI).

Katz, S., Lesgold, A., Eggan, G., & Gordin, M., 1993. Modeling the Student in SHERLOCK II. Journal of Artificial Intelligence and Education (Special Issue on Student Modeling), Vol. 3, No. 4, pp. 495-518.

Khuwaja, R. A., 1994. A Model of Tutoring: Facilitating Knowledge Integration Using Multiple Models of the Domain. Ph.D. Thesis. Illinois Institute of Technology, Chicago, IL.

Khuwaja, R. A., Rovick, A. A., Michael, J. A., and Evens, M. W., 1995. A Tale of Three Protocols: The Implications for Intelligent Tutoring Systems. Yfantis, E. A., Ed. Intelligent Systems: Third Golden West International Conference, Las Vegas, Nevada, 1994. Dordrecht: Kluwer Academic, pp 109–118.

Kim, J., Freedman, R., and Evens, M.W., 1998. Responding to Unexpected Student Utterances in CIRCSIM-Tutor v.3: Analysis of Transcripts. Proceedings of the Eleventh Florida Artificial Intelligence Research Symposium, Sanibel Island, FL, 1998, Menlo Park, CA: AAAI Press, pp. 153-157.

Kim, J., 1997. The Development of Logic Form for Circsim Tutor Version 3. Ph.D. Qualifying paper, Illinois Institute of Technology, Chicago, IL.

Kim, N., 1989. CIRCSIM-Tutor: An Intelligent Tutoring System for Circulatory Physiology. Ph.D. Thesis. Illinois Institute of Technology, Chicago, IL.

Koedinger, K., Shuthers, D., Forbus, K., 1998. Component-Based Construction of a Science Learning Space. Goettl, B., Halff, H., Redfield, C., and Shute, V., Eds., Intelligent Tutoring Systems. 4th International Conference, ITS'98, San Antonio, Texas, USA. Berlin: Springer-Verlag, pp. 166-175.

Lavoie, B., Rambow, O., and Reiter, E., 1997. Customizable Descriptions of Object-Oriented Models. Proceedings of the 5th International Conference on Applied Natural-Language Processing (ANLP-1997), pp. 253-256, Washington, DC.

Lesgold, A., Katz, S., Greenberg, L., Hughes, E., Eggan, G., 1992. Extensions of Intelligent Tutoring Paradigms to Support Collaborative Learning. Dijkstra, S., Krammer, H., and van Merrienboer, J., Eds., Instructional-design Models in Computer-Based Learning Environments. Berlin: Springer-Verlag, pp. 291-311.

Li, J., Seu, J. Evens, M., Michael, J.A., and Rovick, A.A.. 1992. "Computer Dialogue System (CDS): A System for Capturing Computer-Mediated Dialogues." Behavior Research Methods, Instruments, and Computers (Journal of the Psychonomic Society). Vol. 24, No. 4. pp. 535-540.

Macrelle, M., Desmoulins, C., 1998. Macro-Definitions, a Basic Component for Interoperability between ILEs at the Knowledge Level: Application to Geometry. Goettl, B., Halff, H., Redfield, C., and Shute, V., Eds., Intelligent Tutoring Systems. 4th International Conference, ITS'98, San Antonio, Texas, USA. Berlin: Springer-Verlag, pp. 46-55.

McCoy, K., 1989. Generating Context Sensitive Responses to Object-Related Misconceptions. Artificial Intelligence, Vol. 41, pp. 157-195.

Merrill, D., Reiser, B., Ranney, M., and Trafton, J., 1992. Effective Tutoring Techniques: A Comparison of Human Tutors and Intelligent Tutoring Systems. Journal of the Learning Sciences, 2(3): pp. 277-305.

Michael, J. A., Rovick, A.A., Evens, M.W., Shim, L., Woo, C., and Kim, N., 1992. The Uses of Multiple Student Inputs in Modeling and Lesson Planning in CAI and ICAI Programs. Tomek, I. Ed., Computer-Assisted Learning: 4th International Conference, ICCAL '92, Wolfeville, Nova Scotia, pp. 441-452. Berlin: Springer-Verlag, 1992. Published as v. 602 of Lecture Notes in Computer Science.

Mitrovic, A., 1998. Experiences in Implementing Constraint-Based Modeling in SQL-Tutor. Goettl, B. P., Halff, H. M., Redfield, C. L., and Shute, V. J. Eds., Intelligent Tutoring Systems. 4th International Conference on Intelligent Tutoring System (ITS–98), San Antonio, Texas, pp. 414–423. Berlin: Springer-Verlag. (Lecture Notes in Computer Science, no. 1452.)

Ohlsson, S., 1986. Some Principles of Intelligent Tutoring. Instructional Science, Vol. 14, pp. 293-326.

Ohlsson, S., 1992. Constraint-based Student Modelling, 1992, Artificial Intelligence in Education, Vol. 3, No. 4, pp. 429-447.

Paris, C., 1988. Tailoring Descriptions to a User's Level of Expertise. Journal of Computational Linguistics, Vol. 14, No. 3, pp. 64-78, September 1988.

Paris, C., 1993. User Modelling in Text Generation, London: Frances Pinter.

Quinlan, R J., 1993. C4.5: Programs for Machine Learning. Los Altos, CA: Morgan Kaufmann.

Ragnemalm, E.L., 1996. Student Diagnosis in Practice: Bridging a Gap. User Modeling and User-Adapted Interaction, Vol. 5, pp. 93-116

Reiter, E., Cawsey, A., Osman, L., and Roff, Y., 1997. Knowledge Acquisition for Content Selection. Proceedings of the 1997 European Workshop on Natural Language Generation, pp. 117-126. Duisberg, Germany.

Rovick, A.A., and Michael, J.A. 1986. CIRCSIM: An IBM PC Computer Teaching Exercise on Blood Pressure Regulation. Proc. 30th Int. Union Phys. Sci. p. 318.

Sandberg, J.A.C., 1987. Coaching Strategies for Help Systems: EUROHELP. The Third International Conference on Artificial Intelligence and Education. AICOM, vol. 0, pp. 51-53.

Self, J. A., 1990a. Bypassing the Intractable Problem of Student Modelling. Frasson, C. and Gauthier G., Eds., Intelligent Tutoring Systems: at the Crossroads of Artificial Intelligence and Education, pp. 107-123, Norwood, N.J.: Ablex.

Self, J. A., 1990b. Theoretical Foundations for Intelligent Tutoring Systems. Journal of Artificial. Intelligence in Education, Vol. 1, No. 4, pp. 3-14.

Self, J. A., 1994. Formal Approaches to Student Modelling. Greer, J.E. and McCalla, G.I., Eds., Student Modelling: the Key to Individualized Knowledge-Based Instruction, Berlin: Springer-Verlag, pp. 295-352.

Shim, L., 1991. Student Modeling for an Intelligent Tutoring System: Based on the Analysis of Human Tutoring Sessions. Ph.D., Thesis. Illinois Institute of Technology, Chicago, IL.

Shim, L., Evens, M. W., Michael, J., and Rovick, A., 1991. Effective Cognitive Modeling in an Intelligent Tutoring System for Cardiovascular Physiology. Proceedings of the 4th Annual IEEE Computer-Based Medical Systems Symposium, Baltimore, pp. 338-345. IEEE Computer Society Press.

Sleeman, D. H. and Brown, J. S., 1982. Eds., Intelligent Tutoring Systems. New York: Academic Press.

Sleeman, D. H., Ward, R. D., Kelly, E., Martinak, R., and Moore. J., 1991. An Overview of Recent Studies with PIXIE. Goodyear, P., Eds. Teaching Knowledge and Intelligent Tutoring, Norwood, NJ: Ablex Publishing Corporation, pp. 173-185.

Stevens, A. L., Collins, A., and Goldin, S., 1982. Misconceptions in Student's Understanding. Sleeman, D. and Brown, J.S., Eds., Intelligent Tutoring Systems, pp. 13-24. New York: Academic Press.

Van der Linden, K. and Di Eugenio, B., 1996a. A Corpus Study of Negative Imperatives in Natural Language Instructions. Proceedings of the 17th International Conference on Computational Linguistics (COLING '96), Copenhagen. Cmp-lg/9607014.

Van der Linden, K. and Di Eugenio, B., 1996b. Learning Micro-Planning Rules for Preventative Expressions. Eighth International Workshop on Natural Language Generation (INLG '96), Sussex, UK. Cmp-lg/ 9607015, pp. 11-20

VanLehn, K., 1988. Student Modeling. Polson, M.C., and Jeffrey, J., Richardson, Eds., Foundations of Intelligent Tutoring Systems, pp. 55-78. Hillsdale, NJ: Lawrence Erlbaum Associates.

Wenger, E., 1987. Artificial Intelligence and Tutoring Systems. Los Altos, CA: Morgan Kaufman.

Woo, C W., 1991. Instructional Planning in an Intelligent Tutoring System: Combining Global Lesson Plans with Local Discourse Control. Ph.D., Thesis. Illinois Institute of Technology, Chicago, IL.

Woo, C.W., Evens, M.W., Michael, J.A., and Rovick, A.A., 1991. Dynamic Instructional Planning for an Intelligent Physiology Tutoring System. Proceedings of the 4th Annual IEEE Computer-Based Medical Systems Symposium, Baltimore, pp. 226-233. IEEE Computer Society Press.

Woolf, B.P., 1984. Context-Dependent Planning in a Machine Tutor. Ph.D. Thesis. University of Massachusetts at Amherst, Amherst, MA. COINS Technical Report 84-21.

Woolf, B.P., 1992. Towards a Computational Model of Tutoring. Jones, M. and Winne, P.H., Eds., Adaptive Learning Environments: Foundations and Frontiers. NATO ASI Series, 1992.

Woolf, B.P. and Murray, T., 1994. Using Machine Learning to Advise a Student Model. Greer, J.E. and McCalla, G.I., Eds., Student Modelling: the Key to Individualized Knowledge-Based Instruction, Berlin: Springer-Verlag, pp. 127-146.

Zhou, Y. and Evens, M. W., 1999. A Practical Student Model in an Intelligent Tutoring System. Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence. Chicago, IL, pp. 13-18.

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A., and Evens, M. W., 1999a. What Should the Tutor Do When the Student Cannot Answer a Question? Proceedings of the 12th Florida Artificial Intelligence Symposium (FLAIRS–99), Orlando, FL, pp. 187-191.

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A., and Evens, M. W., 1999b. Delivering Hints in a Dialogue-Based Intelligent Tutoring System. Proceedings of the 16th National Conference on Artificial Intelligence (AAAI–99), Orlando, FL, pp. 128-134.