Development of Techniques to Perform Simulation-Adaptation in Perform a Simulation Training Environment Using Expert Systems Methods

Cheryl E. Bagshaw
University of Central Florida

DEVELOPMENT OF TECHNIQUES TO PERFORM SIMULATION-ADAPTATION IN A SIMULATION TRAINING ENVIRONMENT USING EXPERT SYSTEM METHODS

BY

CHERYL E. BAGSHAW
B.S.E., University of Central Florida, 1986

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program
of the College of Engineering
University of Central Florida
Orlando, Florida

Summer Term
1988

ABSTRACT

The use of computers for instructional purposes is steadily increasing, along with an emphasis on developing systems which create environments tailored to human beings. Artificial Intelligence techniques have been incorporated into these systems with an aim at developing better methods of modeling or simulating knowledge and intelligent behavior. One type of these systems, Intelligent Simulation Training Systems (ISTS), utilize a simulation in the training process. This is an ideal environment for the instruction of skills which focus on the ability to understand the time and space relationships of objects.

An intelligent tutor module of an ISTS must configure scenarios for the simulation which meet the objectives of the student's current lesson. This document describes research efforts aimed at designing and implementing methods in which a tutor module intelligently configures scenarios off-line and then dynamically adapts these scenarios on-line as required, within the simulation.

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to my committee chairman, Dr. Avelino J. Gonzalez, for his support and guidance throughout the preparation of this document. His advice and encouragement throughout my graduate studies was invaluable and greatly appreciated.

I wish to thank Dr. John E. Biegel for giving me the opportunity to perform research for the ISTS project. A significant amount of experience was gained and the support received from the team members is also appreciated.

I especially wish to thank Taha Sidani, for his intellectual and emotional support over these past years. He worked long and hard hours with me as we both progressed through our degrees and encouraged me every step of the way.

Special thanks is expressed to Jeannette and Gregory Bagshaw, for always offering their loving support.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This chapter describes the area of research, the objectives, the task and the focus of this thesis. Definitions of related terminology are also provided in the section which discusses the area of research.

The Area of Research

The use of computers for instructional purposes is steadily increasing, along with an emphasis on developing systems which create environments tailored to human beings. The application of computers to provide course content instruction in the form of drills, tutorials, and simulations is referred to as computer-aided instruction (CAI). One possible advantage of CAI is that a less costly form of instruction within a specific subject area may be provided. This is because any number of students may be accommodated, and the presentation of the material may be offered at any time. Another benefit is students may learn at their own rate, independent of other students' abilities. Unfortunately, CAI systems are generally inflexible and provide no individualized instruction. The evaluation and planning process tends to be static, in that no modification of the lesson occurs until after the lesson is completed.

The introduction of artificial intelligence techniques were later incorporated into computer-aided instruction with an aim at developing better methods of modeling or simulating knowledge and intelligent behavior. Systems incorporating artificial intelligence are referred to as intelligent computer-aided instruction (ICAI) or Intelligent Tutoring Systems (ITS) (Sleeman and Brown, 1982).

Individualized tutoring from human instructors has demonstrated great effectiveness in fostering learning, because the student's abilities and needs are individually evaluated and used to determine the tutor's next instructional action. By incorporating the concept of individualized tutoring within ITS, a training environment with greater instructive capabilities may be achieved. The introduction of student modeling within ICAI provided a means for creating a model of the student's progress within the subject matter. This knowledge about the student may then be used in the tutoring process to provide individualized training.

Individualized training involves creating a task for a student which is appropriate, and providing assistance in a fit and timely manner. Three main areas of knowledge are required for individualized training and these may be divided and represented by three separate components. The expert module, the student model, and the tutor module are these components (Woolf 1984, 25-27).

The expert module contains knowledge of the specific domain in which the student will receive instruction. This knowledge base holds the correct data and rules from the subject area. This information can be used to evaluate the student's actions or be referenced for presentation of topics to be discussed (Woolf 1984, 27).

Knowledge concerning the student's understanding and possible misconceptions of the domain is incorporated within the student model. This information can be referenced and utilized to make tutorial decisions about the student's progress with the subject matter. The knowledge within the student model allows the capability to conduct individualized tutoring (Woolf 1984, 39).

The knowledge which embodies teaching strategies resides within the tutor module and is necessary to achieve an effective teaching system. These strategies, rules, and processes govern the system's interactions with the student (Woolf 1984, 46). The pedagogical knowledge is the basis in which tutorial decisions are made by a teaching system. Tutorial decisions determine what topics to present, the form of presentation, when intervention is necessary, and what information should be presented at the time of intervention. The tutor module is the component responsible for utilizing the pedagogical knowledge to make these tutorial decisions.

An intelligent tutor module utilizes the knowledge within the student model to conduct instruction in a manner personalized to the needs of an individual student. The actual text that is provided to the student at times of intervention is handled by another component of an ITS, the discourse module. The discourse module is responsible for handling communication between the system and its users (Woolf 1984, 51).

The Task

Skills to be instructed may be divided into two types: cognitive skills and skills which focus on the ability to understand the time and space relationships of objects. Most prior work within ITS has dealt with the training of cognitive skills. Simulation-based training may be utilized for the training of skills required for the manipulation of objects within a time and space domain. This involves use of a simulation to dynamically display the status and location of objects. Depending on the domain of instruction, different procedures, rules, and criteria must be exercised by the student for correct manipulation of the objects. The objective of simulation-based training is to teach these procedures, rules, and criteria, and to provide a dynamic environment in which these skills may be trained.

Current research does not reflect much development of ITS within simulation-based training. The acronym "ISTS," for Intelligent Simulation Training Systems, will be used to distinguish such systems (Biegel 1988). One goal of an intelligent tutor module is the ability to conduct a lesson according to the individual needs of a student. Each lesson defines an objective which states the skills and topics to be covered, and the level of mastery to be achieved. The tutor module is required to configure scenarios for the simulation, in order to meet the objectives of a current lesson.

To personalize the lesson for a student, the tutor module will make use of the student model as a consultant. Prior to the initialization of a training session, the tutor module must configure a scenario to meet the objectives of the student's next lesson. This is based on student model information regarding where the student's progress is within the subject matter. The term "off-line" will be used for future reference to this initialization period prior to a training session.

Once a scenario has started within the simulation, it may need to be updated dynamically based on the performance of the student, which is monitored by the tutor module. At times, the level of difficulty of the currently running scenario may need to be decreased or increased. The tutor module is responsible for determining if there is a need to adapt a scenario and if so, how the scenario should be modified. This monitoring and adapting process should be performed continuously and dynamically by the tutor module. The term "on-line" will be used when referring to actions taken during execution of a running scenario.

If tutorial decisions led to the conclusion that there is a need for a change in the difficulty level, then more decisions are required regarding how to generate this effect. The methods in which the difficulty level of a currently running scenario is decreased differ from the methods used for increasing the level. One method for decreasing difficulty is to increase the amount of intervention from the tutor module.

The task of dynamically increasing the level of difficulty of a currently running scenario is evidently a difficult process. Tutorial decisions must be made concerning what skills should be challenged, what features should be added to the scenario to challenge these skills, and when should the scenario be adapted. This requires the ability of the tutor module to directly affect the simulation, depending on the outcome of the tutorial decisions.

To have an intelligent, simulation-based training environment, there is clearly a need for the tutor module to dynamically adapt a scenario on-line within the simulation for the requirements of a student. This must be achieved in a timely manner and the modification must challenge the skills in need.

Objectives

This document describes research efforts aimed at designing and developing methodologies in which a tutor module of an Intelligent Simulation Training System can adapt a simulation according to the individual needs of a student. The important abilities of a tutor module within this type of training environment are also investigated and discussed in this report.

The goal is to make contributions to several areas of research. These areas are Intelligent Tutoring Systems, Simulation, and Expert Systems. Very limited research in Intelligent Simulation-based Training has been performed. This research describes the tutoring strategies required for ITS which are simulation-based and discusses those strategies which were designed and implemented using expert system techniques.

The focus of this thesis is towards designing and implementing methods in which the tutor module intelligently configures scenarios off-line and then dynamically adapts these scenarios on-line as required, within the simulation. Artificial Intelligence knowledge representation methods were used to encode the various teaching strategies required to perform simulation-adaptation. Expert system techniques for problem solving were applied to perform those tutorial decisions required to determine which skills need to be challenged and how to implement this challenge within the simulation.

The teaching strategies were applied to the domain of air traffic control. The data required from components of a complete system were simulated to test the effects of these teaching strategies. Development of these components is currently in process by the ISTS project at the University of Central Florida, in Orlando, Florida.

The remainder of the material presented is organized in the following manner. Chapter 2 is a background chapter, providing an overview on existing tutoring systems. Comments concerning the tutorial strategies used within each system are highlighted.

A design for a complete tutoring component for intelligent simulation-based training systems is discussed in Chapter 3.

Chapter 4 is a description of the methods researched and implemented involving off-line configuration of scenarios and on-line simulation-adaptation.

Chapter 5 discusses conclusions from the study, and suggestions for further research are provided.

CHAPTER 2

BACKGROUND

This chapter provides an overview of related research in the areas of intelligent tutoring systems. A section discussing microworlds is provided to illustrate their differences as compared with ITS. The last section discusses the major differences between ITS and intelligent simulation training systems.

<u>An Overview of Existing Intelligent Tutoring Systems</u>

Tutoring systems developed in the past have illustrated progress towards intelligent and adaptive tutoring achieved through computers. An overview of some of these systems is provided, with the strong points and weakness of the implemented tutoring methods highlighted.

BIP

The BASIC Instructional Program (BIP) developed at Stanford University was designed to teach introductory programming concepts and skills (Barr, Beard, & Atkinson 1976). Their attempt to personalize tutoring was to select problems for a student based on the state of the student's knowledge of the subject matter. A Curriculum Information Network (CIN) was used to represent the skills and concepts of the subject matter and their interrelationships. The

CIN was used to determine the student's progress within the domain and to select the appropriate problems for the next lesson (Wescourt, Beard, & Gould 1977).

The CIN was authored by persons knowledgeable in the programming subject matter but was an attempt to move away from the strictly structured curriculum which was followed by most computer assisted instructional systems. The semantic network represented a human's interpretation of how each of the skills and concepts related to each other in terms of difficulty, and sets of tasks using these skills and concepts were defined. The semantic network was in fact defined by a human author. However, the succeeding task chosen by the system for presentation depended on the student's state of knowledge.

The student model is updated by the evaluation of the student's performance and by a student's self-evaluation. An opportunity to indicate skills which the student feels are weak is available upon successful completion of a problem.

There were aspects of BIP's tutoring and evaluation methods which were major drawbacks of the system. The solutions used to compare the student's solution against were limited, and there were several instances in which no match was made. This allowed for solutions generated by a student, which were correct, to sometimes be interpreted as incorrect, if that particular solution was not listed. Help and solutions were available at the student's request. The system did not monitor the student's current session to make decisions on whether or not to intervene, and what to present if intervention is deemed necessary. The student decided at which level the requested help would be provided, and the student was allowed to quit in the middle of a problem. The student was allowed too much control over the tutoring process and the system did not construct individualized tutoring interactions. These drawbacks and others promoted inaccurate modeling of the student's knowledge and prevented appropriate tutoring based on the student's needs (Cochran 1985, 102-121).

SOPHIE

John Seely Brown and Richard Burton at the University of California, Irvine, began development of A Sophisticated Instructional Environment (SOPHIE) (Brown, Burton, and Bell 1974, 1975). This system provided an environment in which students were allowed to create their own hypothesis and explore their own ideas. This reactive learning environment critiques the student's ideas and provides advice. The domain of electronic troubleshooting was used because the student can perform experiments and measurements to test proposed hypothesis as to where the problem in the circuit lies. An electronic simulator was used to model the circuit and the system inserted a fault to be isolated by the student.

Once the student had performed a series of tests, he/she then made a hypothesis regarding the fault. SOPHIE's job was to determine if the hypothesis was consistent with the student's measurements. A list of possible hypothesis would be provided for a help request made by the student, in the case that no hypothesis could be formulated by the student.

Three versions of SOPHIE were developed. SOPHIE I did not have a student model and did not make tutorial initiatives. Questions asked by the student were answered and proposed hypotheses were evaluated. The system did not interpret when the student was having problems, and therefore did not provide assistance until asked for by the student. In addition to providing the features inherent in the first version, SOPHIE II furnished a means for allowing the student to watch the system demonstrate troubleshooting strategies on a given faulted circuit. The system would proceed through a series of tests and measurements, providing textual feedback of the troubleshooting strategy used. The discourse generated during this execution appears impressive, but it is merely achieved through prestored explanations (Wenger 1987, 51-78).

One goal of SOPHIE III was to provide a coaching environment in which the system determined if intervention is necessary to provide the student with advice. This would be dependent on the student model. Neither of these features, however, was fully implemented. The inferencing capabilities of the system were enhanced so the system could better explain the reasoning behind the student's troubleshooting behavior. Knowledge engineering techniques were applied to provide SOPHIE with more reasoning capability.

One aspect of SOPHIE with major importance is the natural language processing abilities of the system. In a reactive learning environment, the student needs to be allowed to ask questions, preferably in a format as natural as possible. SOPHIE illustrated significant power in the interpretation of student's inputs.

STEAMER

A simulation-based training system, called STEAMER (Hollan, Hutchins, Weitzman 1984) was developed to investigate models people use to think and reason, graphical interfaces for interactive inspectable simulations, conceptual fidelity, and implementation philosophy. The goal of this system was to provide instruction on propulsion engineering. A color graphics interface to a simulation of a propulsion plant was provided. This interface allowed the student to monitor the plant at different levels and manipulate the plant's controls.

Much effort by Hollan, Hutchins, and Weitzman was put into developing and implementing methods to maintain an accurate representation of the statuses of both the plant and student. The representation of the information required to manage different levels of the plant was another major representation issue.

The instructional strategy of STEAMER allowed students to manipulate and control different components of a plant and visually inspect the effects of the changes. The students also had the ability to view different aspects of the system which one could not normally witness in a real plant. This instructional strategy is limited by not having adequate questioning provided by the system itself. The student had the freedom of exploring ideas, but was not guided enough to provide instruction which covered all necessary concepts. The evaluation process of the student's behavior is limited and future expansion by Hollan, Hutchins, and Weitzman is proposed.

WEST

One of the first "computer coaches," WEST, was developed by Richard Burton and John Seely Brown (1979). The term "coach" describes a computer-based teaching environment in which the student performs or solves problems while the system "looks over the shoulder" and provides guidance and help. WEST was a coaching system built around the game "How the West Was Won". The coach recognizes weaknesses within the student's performance and provides explanations for these weaknesses. The coach intervenes when the student is in need of an idea and provides suggestions at this point.

"How the West Was Won" was a board-game, originally designed by Bonnie Anderson of the Elementary Mathematics Project at the University of Illinois. In WEST, the game board was computer-simulated and was 70 spaces long. The object of the game was to be the first player to land exactly on space 70, while following rules of the allowable moves which could be made by a player (Cochran 1985, 362-364).

To have a successful coaching strategy, decisions on when to interrupt the student and what to provide at the time of intervention have to be made carefully. These decisions were based on the information of the student's knowledge represented in the student model. The tutoring paradigm used by the WEST system was called "Issues and Examples." The skills and concepts the student was expected to master were defined as the issues and the problems or tasks representing the issues were called examples. Four levels of help were available to the student in which the detail of the hint was dependent on the degree of weakness shown.

Limitations within the evaluation and modeling methods used in WEST were present. The system could not accurately evaluate which issue kept the student from making a correct move that involved more than one issue. Also, student's are not always consistent and forget to use a skill that they do in fact know. This skill may then be labeled as unknown within the student model (Cochran 1985, 372-373).

WEST's instructional strategy of coaching provided an environment which assisted the student through times of difficulty and suggested better moves which otherwise may never have been discovered by the student. However, no curriculum or instructional sequences were used by the system to exercise specific skills.

GUIDON

GUIDON developed by William Clancey and his colleagues at Stanford University (Clancey 1984) was an intelligent tutoring system for teaching medical diagnosis. MYCIN (Shortliffe 1976), an expert system for selecting antibiotic therapy for infectious diseases, was the basis for the GUIDON project. Clancey felt tracing MYCIN's reasoning during a consultation by asking "why" or "how" did not provide an efficient method for teaching the knowledge within MYCIN. GUIDON was developed by utilizing the knowledge base of MYCIN and explicitly representing teaching methods independently.

A case is selected and described by GUIDON and the student asks questions and formulates hypothesis to diagnose the problem. Differential modeling is used to evaluate the student. This technique compares the student's behavior against the expert's behavior. The teaching methodology used was called "case method tutoring" by Clancey.

The important feature of GUIDON is the complete separation of domain knowledge from pedagogical knowledge. This concept of modularity allows the tutorial portion to be easily adapted for use in other domains (Wenger 1987, 265-268). The tutorial strategies involved provide intervention when the student's performance is observed to be non-optimal or when the student requests intervention.

CMU-LISP Tutor

The LISP tutor was developed at Carnegie-Mellon University by John Anderson, Brian Reisor, Robert Farrel, and colleagues. The LISP tutor (Anderson and Reiser 1985) presented short instructional sequences to the student, and then guided the student through a series of programming problems. Two major modules, the "problem-solver" and the "advisor," are utilized by the tutor. The problem-solver monitors the student's performance and models the student. The advisor provides tutorial interaction for the student. A successful aspect of the LISP tutor was that immediate feedback was provided. The program monitors the students as they write their code, and alerts them to errors immediately.

The LISP tutor can function in four distinct problem spaces to cover issues of design and coding. The problem space holds production rules which are ordered by classes of difficulty. The tutor can change problem space according to the needs of the student. Therefore, if the student needs exercise within one aspect of programming, the tutor can reference problems from the appropriate problem space. Each lesson makes use of a different rule set, especially tailored to the needs of its specific level. These rule sets are ordered by complexity and each are accessed by the tutor when the student has reached the appropriate level.

The system contains an "ideal model" which represents the correct rules which the tutor is trying to teach. A "buggy model" is also contained within the tutor's knowledge base. The buggy model contains rules which are a variant of the ideal model's rules. Both of these models are used to evaluate the student's course of action. After each response made by the student, the tutor makes inferences upon which rules or goals could have produced the student's responses. Hence, the LISP tutor performs student modeling interactively.

MHO

MHO (Lesgold, Bonar, Ivill, and Bowen 1987) is a tutoring system which supports both free exploration and guided problem solving. The domain of instruction was electronics troubleshooting. The concept of "steering testing" is used by the tutoring component when the system is in control of the interaction. Tasks are generated dynamically based on the student's observed performance and the goals of the current lesson.

A layered curriculum representation proposed by Lesgold (1987) organizes the curriculum for tutoring systems into three layers. At the lowest level is the knowledge layer, in which the subject matter is represented as separate issues which are linked together. Above the knowledge layer, resides the curriculum layer. The curriculum layer represents the goals and subgoals, defining how the subject matter should be organized into successive lessons. At the top of the curriculum representation scheme, is the aptitude layer. This layer represents skills such as learning abilities or reasoning skills.

The student model of the system contains a separate evaluation of mastery for each skill or issues to be addressed. The tutoring component uses the information within the student model and the knowledge about the curriculum's structure to generate problems for the student. MHO concentrates on task generation for guiding the student's learning process rather than focusing on complete explanations of the behavior of the circuit. This corresponds to the bite-sized tutoring architecture presented by Bonar, Cunningham, and Schultz (1986). Tutoring systems following this architecture are organized around pedagogical issues, called bites. This differs from the organization of systems around functional components, such as a diagnostic or an expert module. A bite focuses on a specific piece of subject matter and contains information about its conceptual and curricular relations to other bites. Conceptual relations correspond to information regarding how bites are classified into classes and subclasses with respect to related bites. Curricular relations define which bites are prerequisite to a related bite. Each bite contains student model information stating the student's mastery of the particular subject matter within the bite. Tutorial strategies also reside within each bite which allow for problem generation or instructional interventions relating to the knowledge of the bite (Wenger 1987, 146-149).

Microworlds

Microworlds are software which provide a training environment in which students may explore ideas. They usually involve the use of graphics. A microworld simulates the domain and the student is responsible for managing the learning process. The student serves as his own tutor. No specific learning agenda is embedded within the software. Therefore, the scope of the subject matter learned by the student will only be that in which the student decided to investigate. There is no assurance that all important concepts will be covered. Also microworlds do not judge a student's performance and utilize this information for future sessions.

An example of a microworld is the LOGO project applied to turtle geometry (Papert 1980). This was developed to help children learn problem-solving strategies. Children were provided with commands which allowed the drawing and combining of geometric shapes. The system also furnished commands which permitted the student to manipulate and change the components comprising the shapes. The student observes the effects of these changes, therefore building an understanding of the mathematical relationships of regular shapes.

Like microworlds, STEAMER and SOPHIE provide a simulation and allow the student to explore the domain. However, these systems differ from microworlds in that they simulate knowledge about a domain, while a microworld simulates a domain under study. The use of knowledge-based systems within STEAMER and SOPHIE is a major difference from an AI standpoint. Thus microworlds are not considered to be ITS (Wenger 1987, 423-425).

Differences Between Intelligent Simulation-Based Training Systems and ITS Developed to Date

The tutoring systems discussed in the previous section concentrate on providing instruction for cognitive skills. Examples of cognitive skills include programming, ability to formulate hypotheses for specific problem areas, understanding of a subject matter, and the ability to solve problems requiring mental models or reasoning. Tutoring geometry, LISP, electronic troubleshooting, and arithmetic are examples of instruction on cognitive skills. The problems presented to test cognitive skills are generally completely defined and posed in entirety at the time of inquisition. The student answers the question or solves the problem after some type of mental reasoning has been completed.

Another and different area of skills available for instruction are those that are required to understand the time and space relationships of objects. Examples of domain areas representing these skill types are air traffic controlling, driving, and flying. These environments may be simulated graphically by a computer, allowing students to control and manipulate the objects simulated. Tutoring may then be provided to teach the students rules, concepts, and procedures which are appropriate for the simulated environment. This is the basis for Intelligent Simulation Training Systems (Biegel et al. 1988).

The problems posed by the tutoring component of an ISTS are dynamic, because only the starting conditions for the scenario are provided. The simulation is dynamically updated to reflect the current status of the scenario. The outcome of a scenario depends on the student's input. Therefore, initial problems generated by the system are not complete since as the simulation runs, the status of the problems within the scenario changes. This is a major difference from the static problems generated by ITS tutoring cognitive skills.

Some ITS developed use a simulation to generate problems for a student, but the simulated objects are not functioning dependently amongst each other in a time and space domain. SOPHIE and STEAMER are examples of systems which utilize a simulation to achieve tutoring. SOPHIE displays an electronic circuit and STEAMER exhibits gages, valves, and various pipes within a propulsion plant. The student is allowed to interact with the simulation, but the objects simulated do not relate to one another within a time and space domain.

The tutoring systems discussed in this chapter represent only a subset of the tutoring systems researched and investigated. These systems were selected for discussion because they embodied features regarding teaching strategies which are relevant to the thesis.

CHAPTER 3

A COMPLETE TUTOR MODULE FOR ISTS

The main objective of an intelligent tutor module is to conduct a lesson in a manner that best suits a student's needs. This includes enabling the tutor module to modify its behavior depending on the abilities of the student and the current mode of the system.

Modes of Operation

The tutor module in an Intelligent Simulation Training System should behave accordingly, based on the role to be played for the student's current session with the system. These different modes in which the tutor module should operate pertain to the student's need for a demonstration, review, coach, or an evaluation at different stages in the student's learning process.

Demonstration Mode

There are two ways in which the demonstration mode can be evoked. The first is when a new student is introduced to the system for the first time. The tutor module will provide a general demonstration of the system and familiarize the student with the system commands.
The demonstration mode is also initiated when a new skill or topic is to be introduced to the student. In this case, the

tutor module will furnish a presentation demonstrating how the concept should be applied.

Reference Mode

The reference mode may be initiated upon a student's completion of a session. The history of a student's session is stored for a limited time and may be accessed by the instructor or the student. This allows the student to proceed through a session to re-enact the problems that were presented. Errors made may become more obvious and the student may further reinforce better solutions which were suggested by the expert. This access to history files also permits the student to pose questions for instructors and have available for display the situation which caused the error. Instructors have access to student's files and may occasionally review sessions to prevent "loosing touch" with student's accomplishments and weaknesses.

Coaching Mode

During the coaching mode, the tutor module will act as a coach, personalizing a session for the student's individual needs. Each time a student begins a new session, the tutor module will reference the student model to determine where the student is within the subject matter. Some domains in which instruction is to be provided may be governed by regulations and agencies. These agencies may enforce strict guidelines on how the information should be presented. This is the case of Air Traffic Controlling. For domains under this type of influence, a lesson-sequence will be defined by appropriate individuals. The lesson-sequence specifies the order in which the domain should be taught. This should begin with lessons covering basic concepts and progress logically to lessons which are more advanced and challenging.

Each lesson in the sequence specifies an objective. This objective defines which skills or concepts are to be covered and the degree of difficulty. The level of mastery which must be achieved by the student before progressing to the next lesson is also specified. Lessons do not explicitly define the simulation situations that are necessary to exercise the skills and topics to be covered. These scenarios must be configured and maintained by the tutor module.

For domains which are not regulated by agencies or laws, a differential modeling approach may be taken to determine what will be covered for each session. The student's knowledge of the domain may be modeled in a manner similar to that of the expert. Each skill or concept that the student "learns" will be added to the knowledge within the student model. At the start of each session the tutor module will differentiate between the expert's knowledge and the student's knowledge. This difference in knowledge is then used to construct a lesson for the student.

Under the coaching mode of operation, the tutor module must also administer help and remediation when appropriate. Positive reinforcement should also be provided to encourage the student. All of these functions of the tutor module provide an environment for the student which is as personalized and human-like as possible. These functions will be discussed in more detail in the latter part of this chapter.

Evaluative Mode

Under this mode of operation, the student is provided a test and the tutor module will not intervene during the session. The student's performance is evaluated throughout the session and the final evaluation is presented by the tutor module. The strong and weak points will be highlighted and suggestions for improvement will be furnished. The actual evaluation of the student is the responsibility of a different component within an ISTS. This component provides an evaluation of a student during the coaching mode as well. This information is used to update the student model.

Functions Provided to Personalize the Coaching Mode

Many functions must be performed by the tutor module in order to serve as a personable coach during a student's session. These functions are addressed in this section.

Help

Help is a function of the tutor module which supplies the student with hints or advice for specific situations in a scenario which is currently running within the simulation. Two sets of tutorial decisions have to be made regarding this issue. The first set determines when to intervene and the second decides what to present.

There are two possible ways to evoke help. One is when the student requests help by pressing a pre-assigned help-key or typing a predefined command. A series of menus will then be provided by the system to pinpoint the objects and situations in which help is desired. Help can also be evoked by the tutor module itself. There are many considerations concerning when to allow the tutor module to intervene. Too much intervention inhibits a student's "learning by discovery" process. On the other hand, too little intervention may frustrate and discourage a student.

The tutorial decisions regarding when to intervene can be based on the events occurring in the simulation. Potential conflicts or problems may be predicted and the time of occurrence of these events may be calculated. The tutor module may intervene at certain time intervals leading up to the time in which the violation will occur. The amount of information provided will depend on how close to the actual time of violation the intervention takes place. If there is plenty of time prior to an occurrence of a violation, the tutor module may hint towards the pending situation. If little time is left, the tutor module may provide a complete and optimal solution to correct the problem. The expert module provides the necessary solutions for the tutor module to reinforce the student. As the tutor module monitors the progress of the pending event, intervention will take place regularly, with the amount of information provided increasing with time. This process continues until the student has remedied the situation or until the violation occurs.

Another instance in which the tutor module may intervene is when a student consistently displays a weakness in a skill. The tutor module may provide guidance during the student's next performance of activities which requires the use of this skill. The amount of information presented at the time of intervention will depend on the level of help which has been provided for the skill previously.

Remediation

Remediation may be provided as a supplement to a lesson if it is determined necessary by the tutor module. The decision to initiate remediation may be determined before, during, or upon completion of a session. Before a session is initiated, special instructions may have been left by a human instructor to remediate the student on certain skills or topics. The remediation will be conducted before the student is allowed to begin the session.

During a session, if a student is currently displaying very poor performance on certain skills, regardless of how much assistance is given, the tutor module may decide to freeze the simulation and conduct remediation. This is done for skills which are designated as known by the student model. The assumption is the student knows the skills, but needs to be refreshed.

At the end of a session, the tutor module may decide that the student may progress to the next lesson, only after the student is remediated on certain skills. This is possible if the student has mastered skills with a score on the low end of the satisfactory range. The final scores may pass the student to the next level of lessons, however the scores are marginal and further exercise is necessary.

Remediation may be provided in many forms. The simplest is in textual form in which rules or concepts are presented to be read by the student. This may be appropriate to remind the student of specific rules or concepts, possibly forgotten, which should be applied.

If the student needs exercise on how to apply specific rules or concepts, the simulation may be utilized for remediation purposes. Special drills concentrating on certain skills may be conducted, thus providing a more interactive form of remediation.

Positive Reinforcement

The tutor module has provisions for notifying a student of mistakes or errors. Positive reinforcement should be provided when the student demonstrates correct use of a new skill or performs an excellent maneuver. This will create an encouraging environment for the student.

Explanation

A student may request the system to justify why a solution was suggested by the expert to handle a situation in the simulation. A series of menus will be provided, similar to those for help requests made by the student, to determine which situation the student is referring. The tutor module will need to reference the solution that was generated and require the expert module to explain the reasoning process which led to the solution.

Simulation-Adaptation

The tutor module is required to configure a starting scenario to be portrayed by the simulation at the beginning of a student's session. This scenario will be configured depending on where the student is within the subject matter. The topics and skills to be covered must be exercised by these starting conditions.

After the session is initialized, the status of the scenario changes as the simulation runs. The student's input affects the scenario, and as time passes, the scenario is modified depending on the status of the objects within. The tutor module will need to adapt the scenario dynamically to challenge appropriate skills at appropriate times. The modification induced by the tutor module is based on the time of adaptation, the current situations within the scenario, and the purpose of the adaptation.

The simulation-adaptation function is a new and complicated issue, but a requirement for personalized tutoring within an Intelligent Simulation Training System. This issue is addressed in more detail in the next chapter.

CHAPTER 4

METHODOLOGY FOR SIMULATION-ADAPTATION

A tutor module needs to perform simulation-adaptation with the intent to provide tasks for the simulation which fulfill the objectives of the student's current lesson and to dynamically adapt the session to meet the needs of the student. This function distinguishes tutor modules for Intelligent Simulation Training Systems from those of Intelligent Tutoring Systems, which, in general, lack this type of simulation interface. This chapter discusses the different approaches available for generating tasks for students and justifies the method chosen for exploration by this research. The implementation of this method with respect to ISTS is also described in detail.

Methods for Generating Tasks

Three main methods in which tasks are generated for a student have been demonstrated in previous ITSs. The first is the exploratory approach in which the student is free to choose the topics or concepts to investigate. The system presents a problem for the student which relates to the subject matter chosen by the student. This approach has merit because it provides an interesting environment; however, several drawbacks are apparent. One disadvantage is the subject matter is not presented in an organization which promotes the material to be learned in a logical order. Prerequisites may or may not be satisfied for topics elected for investigation. Also, there is no assurance that all necessary topics or concepts will be presented to the student.

The second method involves a differential process between the information contained within the student model and the domain knowledge. As the student learns the material in the domain, the student model is updated in the same manner in which the domain knowledge was developed. The student model essentially keeps track of the topics learned. When the system is ready to generate a task for the student, the knowledge contained in the student model is compared against the domain knowledge to determine the next concept necessary for presentation. The tasks generated to exercise the concepts are generally completely defined. In other words, once the problem is stated, it does not change. The student derives a solution and the student model is updated based on the solution provided. This process continues until all concepts have been covered. This method works well for teaching cognitive skills, but requires sophisticated student modeling.

The last method for discussion involves the tutor module following a lesson sequence which has been previously outlined by a human instructor. This lesson sequence is comprised of individual lessons concentrating on specific concepts. The lessons are logically ordered by increasing complexity and in a manner ensuring prerequisites for a particular lesson would have been satisfied by prior lessons. This methodology allows the tutor module to proceed through the subject matter in a manner defined as well organized by a human instructor. All concepts deemed necessary for coverage will be referenced by the tutor module. The effectiveness of this method relies heavily on how much information is rigidly specified in a lesson. If each lesson has a completely defined task to be used to exercise the topic specified by the lesson, the overall effect of task generation resembles a workbook. Each student proceeding through the sequence will receive the same tasks. The concepts of lessons and lesson sequences do, however, have potential for generation of tasks in ISTS. This principle is discussed below.

Off-Line Task Generation in ISTS

As mentioned previously in this document, the teaching of skills which involve understanding the time and space relationships of objects can be effectively accomplished by the use of a simulation. The tutor module must generate tasks for a student which reflect the current topic of discussion. The generated tasks will be presented by the simulation. Since the simulation is updated dynamically, the task initially presented will change as time passes. Therefore, the tutor module can only generate the starting conditions of the task. These tasks which are generated off-line will be referred to as "scenarios" because they specify the situations to be present within the simulation at the time in which the session begins running.

The method proposed for off-line scenario generation involves having the tutor module utilize a lesson sequence (see Figure 1). Lessons will be defined generally enough to allow the system to generate a different scenario for a lesson, each time the lesson is referenced by the tutor module. Each lesson specifies a description of what should be present within its corresponding scenario, but does not specify how this should be represented within the simulation. For example, a lesson may specify five objects to appear in a scenario, each with different capabilities in speeds, and heading in directions which will cause no future intersections amongst them, unless otherwise changed. The system will then have to calculate the required coordinates, headings, speeds, and other directives for the simulation which reflect the conditions specified by the lesson. The directives will be generated with as much randomness as allowed by the guidelines specified by the lesson. This promotes sessions which provide instruction specified by the current lesson, but with enough variability to prevent a rigid lesson sequence.

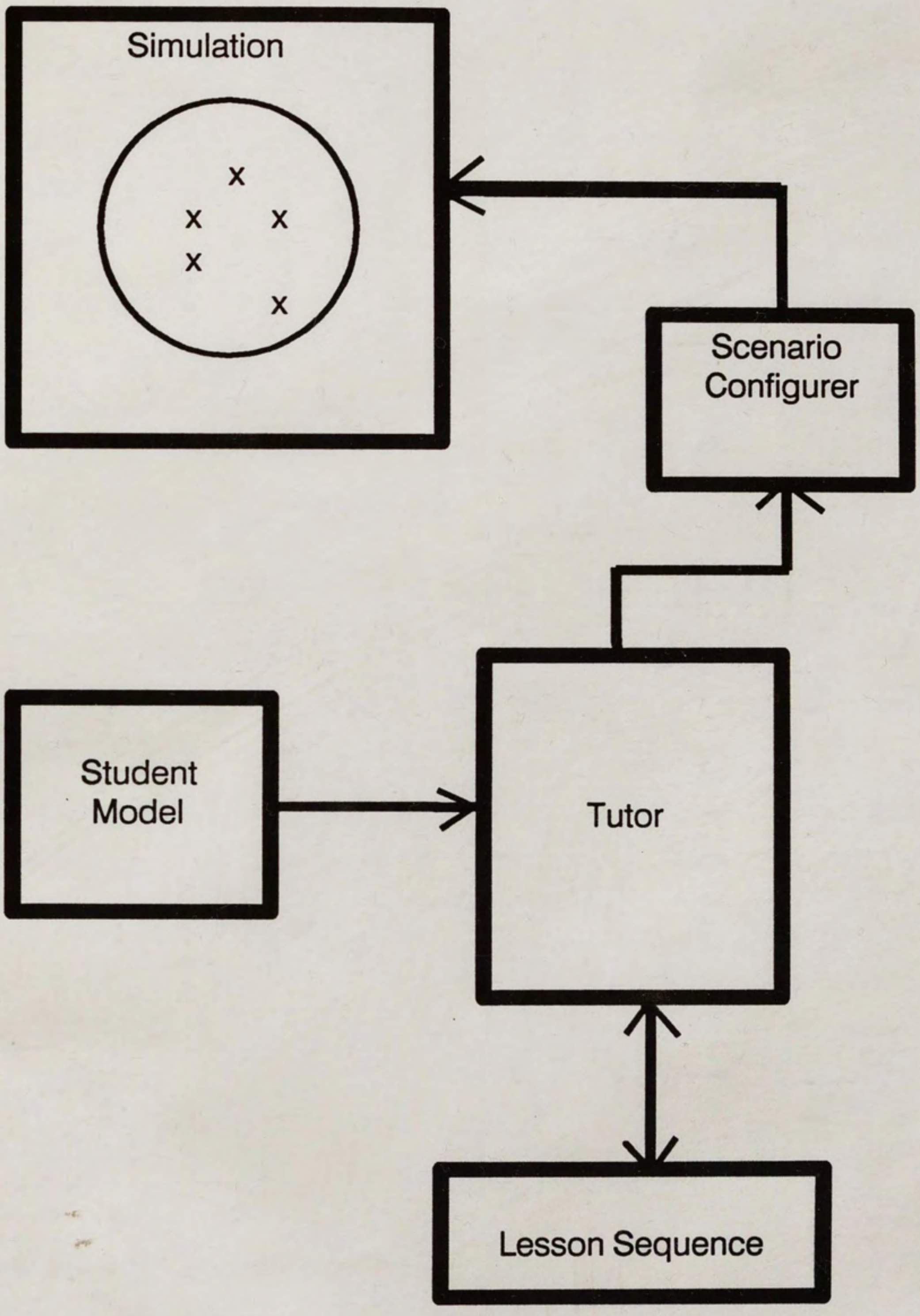

Figure 1. Interfaces Between the Components of the System.

On-Line Generation of Tasks

After the generated scenario has been loaded into the simulation, the simulation begins to run, thus initializing a student's session. The student will proceed with executing commands to control the objects as required by the lesson. The performance of the student can be dynamically monitored and made available for the tutor module. The tutor module can therefore make decisions concerning the student's demonstrated performance of the skills covered by the present lesson. At times the tutor module may decide to increase the difficulty of the current session.

The lessons and lesson sequencing concepts can again be utilized to accomplish increasing difficulty. In addition to having the lessons specify, in general, what scenario should be present in the simulation at the start, the lesson can also specify techniques which the tutor module should use to generate additional tasks that increase the difficulty of the lesson. For example, a topic for instruction within Air Traffic Control is the maintenance of separation standards between aircraft. This topic, along with the directives for the starting scenario, will be defined by a lesson. Maintenance of separation standards becomes more difficult with the increase in traffic or with the introduction of inclement weather. Methods, such as these, for increasing the difficulty of the topic of separation standards will also be specified by the lesson.

The tutor module can reference these methods at times when it is determined the student needs to be challenged. The tasks generated on-line to be added to the currently running simulation should again be created with as much variability as possible.

Implementation

The methods described above were implemented on a Symbolics LISP Machine and were developed for use in the instruction of handoffs and the maintenance of separation standards in the domain of Air Traffic Control. The Automated Reasoning Tool (ART), developed by Inference Corporation, was used to develop the rules necessary for making decisions concerning a student's past and present performance. It was also utilized to create a menu-driven authoring process to generate lessons. ART's blackboard architecture provided a convenient method for enabling the tutor to dynamically affect the simulation. A blackboard architecture is an inferencing mechanism which posts asserted facts on a blackboard which may match the premise or "if" condition of a rule. When a rule's premise is satisfied by the facts on the blackboard, the rule fires, carrying out the actions listed in the "then" part. ART continuously checks for matches between facts on the blackboard and conditions of rules. Therefore, rules which affect the simulation, are continuously monitored to determine when its conditions are satisfied. At the instant these conditions are satisfied, the actions of the rule are immediately carried out, hence, dynamically affecting the simulation.

Common LISP was used to implement the techniques which generate the directives necessary to drive the simulation. The simulation utilized for this research mimics a radar scope for an air traffic controller, and has the ability to display and update aircraft on the scope with time. The simulation was developed at the Simulation and Control Department of the General Electric Company in Daytona Beach, Florida, by Mr. Michael S. Kelsen and Mr. Blake Moselle, under the direction of Ms. Janice Eisele.

Several functions and processes were implemented to develop a system which performs simulation-adaptation as described, and permit the demonstration of these techniques. The implementation of each of these processes is discussed in the following sections.

The Authoring Process

This process was implemented to allow a human instructor to create lessons and organize these within a lesson sequence for future reference by the tutor module. Whenever the system is loaded, the tutor's main menu is provided, which allows the user to choose between the authoring process or the initialization of a session (see

Figure 2). If the authoring process is selected, the lesson menu is presented which provides options concerning lesson sequence construction.

Six functions may be selected from the lesson menu to assist instructors in the construction of lessons and sequences. The **create** function allows an instructor to create a new lesson and insert the created lesson within the lesson sequence at the position of his/her choice. Each lesson is created as an individual object, which contains a set of attributes or slots. The topic slot allows the instructor to indicate the topic(s) that are covered by the lesson. Slots are provided which specify information concerning the number of objects to appear in the starting scenario and the capabilities these objects should have. Each lesson also contains a slot which permits the instructor to specify whether or not problems should exist between the objects at the start of a student's session which covers the lesson. The system prompts the instructor for the values of these attributes and fills the slots accordingly.

The lesson menu provides three functions for use on existing lessons. The **show** option permits an instructor to view a lesson and verify the values assigned to the attributes. If an instructor feels that a lesson's specifications should be modified, the **edit** option provides the ability to change any values specified. The **delete**

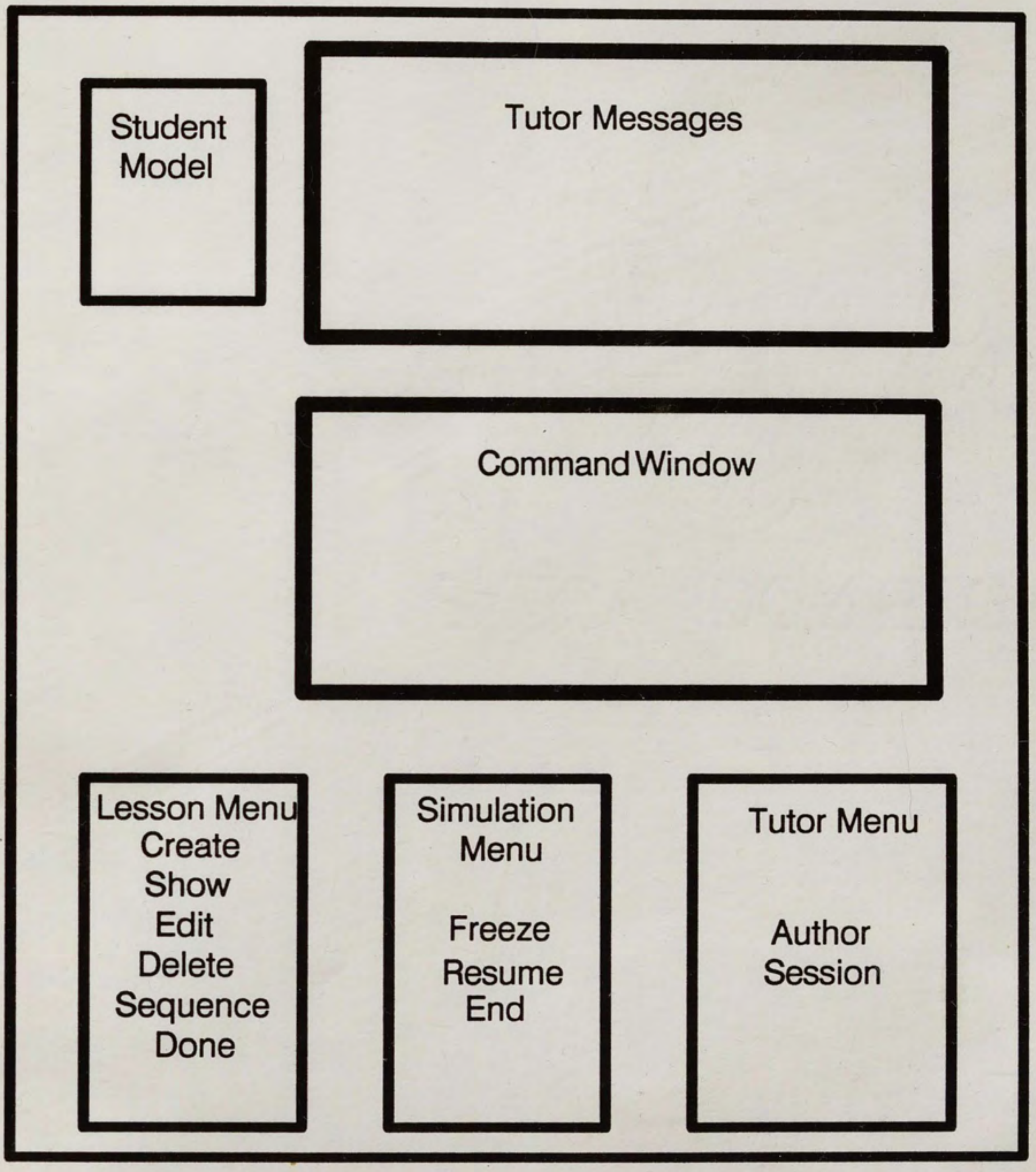

Figure 2. Windows and Menus for Tutor Communication.

utility allows the deletion of a lesson from the lesson sequence. These functions must be performed on lessons existing within the lesson sequence and the system will notify the user of attempts to perform these functions on non-existing lessons.

When the authoring process is exited, the tutor's main menu reappears, permitting a student to initiate a session with the system. The implementation of the tutor module's actions at this stage is discussed next.

The Initialization of a Session

When a session is initialized by a student, the tutor prompts for information regarding the last lesson from which the student received instruction. This knowledge is used by the tutor to determine which lesson in the lesson sequence will be referenced for the student's current session. The specifications of the current lesson are consulted and utilized by the tutor to generate the starting scenario to appear in the simulation.

The calculations for generating the directives necessary for creating a scenario were divided into two classes. One class handled calculations for creating scenarios with problems existing between the planes appearing on the radar scope. The second class determined directives which placed planes on the scope with no problems present at the start of the session. Each lesson specifies if the planes appearing will have mixed abilities in speeds and if they will appear at different altitudes initially. This information is utilized to generate the student's initial scenario. As each plane is created, a scenario file is updated which will be loaded by the simulation when the scenario has been generated completely.

For the generation of scenarios in which there are no current problems and the planes appear at different altitudes, the system will randomly generate altitudes for each plane, with enough vertical separation to meet separation standards. Logical values for the coordinates, headings, and speeds are then generated at random for each plane. The calculations become more involved, however, to create a scenario with no existing problems, when the planes are to appear at the same altitude. First, the system randomly generates a logical altitude to place all planes. The system must then place and head the planes in a manner producing no intersections amongst them. This was accomplished by dividing the scope into quadrants. Coordinates on the scope were generated at random for each plane and the corresponding quadrant was determined. The heading for the plane was then randomly generated from a range of angles, dependent on the quadrant selected, which would head the plane on a path radiating outward from the center of the scope. In the generation of coordinates and altitudes, precautions were taken to prevent placing more than one plane at the same location.

The calculations producing scenarios which contain problems between the planes initially are performed in a manner similar to the approach above. These calculations ensured that at least one problem would be initially present between two planes. For scenarios in which the planes are to appear at the same altitude, an altitude was generated at random as described before. Next, logical coordinates for two planes would be generated at random. The headings of these planes would then be calculated such that the planes will be heading towards each other when the simulation begins. In the case in which different altitudes will be used, the same type of calculations for the coordinates and headings will be performed to head two planes towards each other, but their altitudes will not differ by an amount which meets the vertical separation standards.

Once the initial scenario has been created and stored into a file, the system initiates the session. The simulation references the scenario file and displays the planes on the scope accordingly. The clock is then started and the simulation proceeds to update the status of the planes as time passes. The tutor is now ready to make decisions concerning when to challenge the participating student.

Challenging the Student

If the student is performing well during a session, the tutor needs to add to the simulation to make it more challenging. Implementation of this feature required simulating data which would be provided by a student model. This was accomplished by providing the user with a student model window which allowed the input of student model information.

Each lesson created contains a slot which holds the name of a function to be called by the system when the student needs to be challenged during that lesson's coverage. Whenever the simulated student data indicated good performance, the tutor referenced the current lesson to determine what function was listed in this slot. Any function listed had to have been defined and available for the system to call. The blackboard inferencing mechanism of ART was used for the dynamic monitoring of a student's performance and modification of the simulation. At the instant the simulated student data indicated the student needed to be challenged, this "fact" was posted on the blackboard which triggered the rules governing the on-line task generation scheme. This process was nearly instantaneous which provided a timely manner of updating the simulation.

The function defined for implementation by this system added planes to a currently running simulation. In Air

Traffic Control, the increase in traffic increases the difficulty of maintaining separation standards between aircraft. Each time the simulated student data imply the student should be challenged, an aircraft is added to the simulation. The coordinates, headings, etc., for each aircraft are generated in manners similar to those calculations used to generate startup scenarios.

CHAPTER 5

CONCLUSION

This document has described the differences between Intelligent Simulation Training Systems (ISTS) and Intelligent Tutoring Systems (ITS). The role of a tutor module within ISTS has also been described and the design and implementation of simulation-adaptation for the environment of Air Traffic Control has been recounted. This chapter discusses the results and implications of this investigation, and provides suggestions for further research.

Results

The approach taken to perform simulation-adaptation proved to be effective by the resulting system. Air Traffic Control is carefully regulated and strict guidelines govern how controllers are instructed. The subject matter is taught in a well organized sequence which lends itself well to the methodology developed by this thesis using lessons and lesson sequences.

Several lessons were constructed and initiated to test the off-line scenario generation. The scenarios which were generated and displayed by the simulation reflected the specifications of the lessons. The system was also tested for variability in which the same lesson was repeatedly initiated and the resulting scenarios generated were compared. Each scenario met the specifications of the lesson, but the arrangement of the objects were different in each. This illustrates the potential of utilizing a lesson sequence to teach a domain in a well structured manner, along with the ability to generate various and new tasks each time a lesson is initiated.

For the implementation of the on-line simulation-adaptation method, the use of ART to dynamically affect the simulation was beneficial. This is due to the blackboard architecture utilized by ART. The process of detecting student performance changes and modifying the simulation accordingly was accomplished in a timely manner.

The system developed contains portions which are domain specific and portions which are generic. Generic systems are systems which may be used in any simulation-based training environment. The authoring process implemented may be used for any domain. Each lesson contains a series of slots which are given values by an instructor. The slots specifying the topic, the number of objects, the abilities in speeds and the functions to be called for challenging a student, all can be utilized for any domain concerned with the manipulation of objects in a time and space domain.

The tutorial rules developed which make decisions based on the student's past and present performance can be applied to similar domains and are not restricted to Air Traffic Control. Also, the technique in which the tutor retrieves the specifications of the lesson for use is again not restricted to the domain. The generation of the directives which drive a domain specific simulation are, however, restricted to the domain. If this configurer of scenarios is separated from, but under the control of the tutor module, the possibility of a generic tutor is apparent.

Summary

In the introduction of this document, the need for individualized tutoring within Intelligent Tutoring Systems was stressed. This was the underlying approach taken for the development of techniques to perform simulation-adaptation for an Intelligent Simulation Training System. The utilization of expert system techniques for the implementation of these methods was somewhat limited by the lack of student information which should be provided by a student model. The system developed did, however, elucidate the methods proposed.

Suggestions for Future Research

The system developed by this investigation may be expanded and examined in various ways. Techniques were developed to create starting scenarios for the topics of handoffs and the maintenance of separation standards. The system could be expanded to manage the creation of scenarios for other topics in Air Traffic Control. Also, the current system could be investigated under various domains involving skills which focus on the ability to understand the time and space relationships of objects. This type of exploration would distinguish the generic portions of the system.

The complete design of a tutor module for an Intelligent Simulation Training System was discussed in Chapter 3. The design and implementation of the features not covered by this investigation would contribute to the effectiveness of a tutor module and is also suggested for future research.

REFERENCE LIST

Anderson, J., and B. Reiser. 1985. The LISP tutor. BYTE, 10(4): 159-175.

Biegel, John E., Leslie D. Interrante, Jenifer M. Sargeant, Cheryl E. Bagshaw, Camille M. Dixon, George H. Brooks, Jose A. Sepulveda, and Chin Lee. 1988. Input and instruction paradigms for an intelligent simulation-based training system. In Proceedings of the First Florida Artificial Intelligence Research Symposium, 250-253. St. Petersburg: The Florida AI Research Symposium.

Barr, A., M. Beard, and R. C. Atkinson. 1976. The computer as a tutorial laboratory: The Stanford BIP project. International Journal of Man-Machine Studies 8: 567-596.

Bonar, Jeffrey, Robert Cunningham, and Jamie Schultz. 1986. An object-oriented architecture for intelligent tutoring. In Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages and Applications, 269-276. New York: Association for Computing Machinery.

Brown, John Seely, Richard R. Burton, and A. G. Bell. 1974. SOPHIE: A sophisticated instructional environment for teaching electronic troubleshooting. BBN Report 2790. Cambridge: Bolt Beranek and Newman Inc.

Brown, John Seely, Richard R. Burton, and A. G. Bell. 1975. SOPHIE: A step towards a reactive learning environment. International Journal of Man-Machine Studies 7: 675-696.

Burton, Richard R., and John Seely Brown. 1979. Toward a natural-language capability for computer-assisted instruction. In Procedures for instructional systems development, edited by H. O'Neil. New York: Academic Press.

Clancey, William J. 1984. Methodology for building an intelligent tutoring system. In Artificial intelligence and instruction: Applications and methods, edited by Greg P. Kearsley. Reading and Menlo Park: Addison-Wesley Publishing Company.

Cochran, W. Joseph, Jr. 1985. Intelligent computer-assisted instruction: Artificial intelligence in instruction and training. Ph.D. diss., University of Southern California.

Hollan, James D., Edwin L. Hutchins, and Louis M. Weitzman. 1984. STEAMER: An interactive, inspectable, simulation-based training system. In Artificial intelligence and instruction: Applications and methods, edited by Greg P. Kearsley. Reading and Menlo Park: Addison-Wesley Publishing Company.

Lesgold, Alan M. 1987. Toward a theory of curriculum for use in designing intelligent instructional systems. In Learning issues for intelligent tutoring systems, edited by H. Mandl and A. M. Lesgold. New York: Springer-Verlag.

Lesgold, Alan M., Jeffrey G. Bonar, J.M. Ivill, and A. Bowen. 1987. An intelligent tutoring system for electronics troubleshooting: DC-circuit understanding. Technical Report. Learning Research and Development Center, University of Pittsburgh, Pittsburgh, Pennsylvania. (To appear in Resnick L.B. (Ed.) Knowing and learning: Issues for the cognitive psychology of instruction. Lawrence Erlbaum Associates, Hillsdale, New Jersey.)

Papert, Seymour. 1982. Tomorrow's classrooms. In Horizons in educational computing, edited by Masoud Yazdani. West Sussex: Ellis Horwood Limited.

Papert, Seymour. 1980. Mindstorms: Children, computers, and powerful ideas. New York: Basic Books.

Shortliffe, Edward. H. 1976. Computer based medical consultations: MYCIN. New York: American Elsevier Publishing Company, Inc.

Sleeman, Derek, and John Seely Brown. 1982. Intelligent tutoring systems. New York: Academic Press.

Wenger, Etienne. 1987. Artificial intelligence and tutoring systems. With a Foreword by John Seely Brown and James Greeno. Los Altos: Morgan Kaufmann Publishers, Inc.

Wescourt, Keith T., Marion Beard, and Laura Gould. 1977. Knowledge-based adaptive curriculum sequencing for CAI: Application of a network representation. In _Proceedings of the National ACM Conference held in Seattle Washington_, 234-240. New York: Association for Computing Machinery.

Woolf, Beverly Park. 1984. Context dependent planning in a machine tutor. Ph.D. diss., University of Massachusetts.