Building Intelligent Tutorial Systems for Teaching Simulation in Engineering Education

Brian A. A. Antao, *Student Member, IEEE,* Arthur J. Brodersen, *Senior Member, IEEE,* John R. Bourne, *Senior Member, IEEE,* and Jeffrey R. Cantwell, *Member, IEEE*

Abstract— **This paper describes a framework for building intelligent tutoring systems (ITS's) to teach students the use of various simulation systems used in engineering education. Case studies of two widely used simulators in electronics education [LASAR, a digital logic simulator, and SPICE, an integrated circuit simulator] provide the basis of pedagogical methodology for teaching the use of simulators. This methodology is used to develop a tutorial environment which includes 1) an authoring system that enables an instructor to develop and tailor the course contents; and 2) a course presentation system that communicates this information via a direct manipulation interface to the user. The student perceives the tutorial as a hypertext network which can be freely explored; however, the tutoring system monitors and dynamically reconfigures the accessible information according to the level and attainment of expertise by the student. The environment includes components to monitor and evaluate the performance of the student. This tutorial framework is used to create intelligent tutorial systems for SPICE and LASAR.**

I. Introduction

Simulation and simulators play a vital role in engineering education by providing students the ability to easily demonstrate physical manifestations of theoretical concepts. Simulators can also assist students in developing insights into basic principles. In electronics education, simulators such as SPICE [18] are useful in teaching and explicating the functioning of circuits without actual fabrication. Simulators also provide a reliable means for observing physical effects, such as the circuit response due to parametric changes. Upon graduation, students can expect to use simulators extensively in the design process to verify newly designed circuits and systems, and assist in making design refinements. Thus, teaching the use of simulators should be an essential component of any engineering curriculum. Since simulators are typically computer programs that can model system behavior based on user-defined specifications, computer-aided instructional aids can expedite the teaching process and provide individualized instruction while the student uses simulation systems. This paper shows how intelligent tutoring methodologies can be used in the teaching of simulation.

The impetus for development of a tutorial framework for teaching simulation stems from the development of an advisory system for the digital logic simulator LASAR [15]. LASAR, which stands for Logic Automated Stimulus And Response, is a digital logic simulator and test pattern generator developed and marketed by Teradyne, Inc. A primary goal for LASAR tutorial was to integrate the knowledge communication about the use of LASAR with actual operation of the simulator. The domain knowledge necessary to develop a course structure for LASAR was secured from instructors teaching the training course and encoded into the system. In the course of this development, it was observed that the tutoring technology employed could be generalized and its applicability extended to other similar simulation tools. This observation has led to the creation of the framework described in this paper for capturing a general simulation teaching methodology in an *intelligent tutoring system.*

Intelligent tutoring systems (ITS's) or intelligent computer-aided instruction systems are the new generation of computer-aided training systems that integrate course material, student models, and diagnostic capabilities within a unified framework [14], [20].ITS's incorporate techniques from artificial intelligence to generate more robust tutoring environments that allow the student to follow natural learning stages, such as exploratory learning, and learning by doing. These systems attempt to provide an environment that facilitates a student's acquisition of both factual and experiential knowledge.[1]

II. A Methodology for Simulations

With the increasing use of simulators both in academic research and in industry, the use of simulation has become a vital component of the electrical engineering curriculum. Currently, introductory and advanced level courses in electronics extensively employ SPICE simulation. Simulators serve as a kind of virtual laboratory, providing a useful analysis tool and an experimentation medium without using physical components. Expertise in being able to use a simulator and understanding the simulation methodology is a valuable asset to the student in the maturation process towards becoming a designer. In this section, we present two case studies to show how two widely used simulators typically operate, and develop a set of methodological guidelines for providing effective instruction to students.

A. Digital Logic Simulation with LASAR

LASAR is widely used in the digital electronics industry for design verification and test pattern generation in a wide

Manuscript received January 1990; revised January 1991. This work was supported in part by Northern Telecom Inc. and NSF grant USE-8953497.

The authors are with the Center for Intelligent Systems, Department of Electrical Engineering, Vanderbilt University, Nashville, TN 37235.

IEEE Log Number 9104605.

[1] Factual knowledge corresponds to the course contents, and experiential knowledge is obtained by working multiple examples.

range of applications ranging from communications to microprocessors. LASAR is also used in many universities [17]. This simulator consists of an integrated set of compilers, simulators, and device and model libraries. A system can be simulated and tested for faults by putting together a simulation model from the available device libraries, or the user can develop custom-specific models. The process of design verification and testing with LASAR is decomposed into a series of functionally independent subprocesses. For example, to create a semicustom device library, the user works with the model compiler. The complete simulation and test cycle contains seven stages.

1) Develop a simulation model of the digital circuit. This task is performed as the model/netlist compilation.
2) Define the set of test vectors, or inputs to the simulation. Verification is done by stimulus compilation.
3) Model the environment in which the target circuit will be used, by employing environment and rules compilation.
4) Run the actual simulation, which combines the results of the above three steps.
5) Model faults, using the fault simulation process.
6) Collect fault diagnostic data if using hardware testers.
7) Convert the test fault data generated in the previous step to a format used by a specific tester.

In the basic simulation process, the user first creates a correct structural simulation model of a digital circuit, which is then verified for correct syntax and connection consistency by the netlist compiler. The next stage is the generation of a set of inputs for the simulation, which are written as the stimulus files, and is verified and compiled by the stimulus compiler. In addition, the user can also model environmental effects, such as stray capacitances, noise, etc., that ensure an accurate simulation of the final circuit's target environment. These effects are compiled using the environment and rules compiler. The simulator operates on these compiled modules to carry out the actual simulation, generating the desired output results. LASAR also has advanced features, such as 1) fault modeling that permits the consequences of logic faults and hazards to be studied; and 2) generating inputs for hardware testers from simulation models.

B. Circuit Simulation with SPICE

SPICE is a simulator for integrated electronic circuits that can simulate circuits with components such as resistors, capacitors, transistors, etc. SPICE can simulate all types of electronic circuits at the device level and has built-in device models for most components. In addition, the simulator facilitates detailed semiconductor device modeling [4] and the use of these models in circuit simulation. SPICE also provides a wide range of analysis, such as dc analysis, ac analysis, transient analysis, etc. [5],[8]. The stages in SPICE simulation can be summarized as follows.

1) Write a SPICE circuit description.
2) Create new, or use default device models; environmental effects can be incorporated into device models.
3) Specify the types of analysis required.
4) Specify the input signals that are to be used in the simulation.
5) Indicate the output signals to be generated by the simulation.
6) Run the simulation.
7) Post-process the simulation results.

C. Modeling the Simulation Process

From the evaluation of the simulation process presented in the previous sections, a simple physical model for a typical simulation process can be abstracted. The model is simple since the focus of the tutorial is on classroom instruction. A more detailed and complex simulation model would be necessary and appropriate in a design automation environment. The model presented below serves as a guideline for teaching the simulation process.

1) Create a description of the physical model in a format that can be interpreted by the simulator. This stage involves writing netlists and SPICE circuit descriptions.
2) Define the simulation environment. Typically, simulators also allow environmental constraints such as stray capacitance and operating temperature effects to be modeled.
3) Define a set of inputs on which the simulation needs to be run, i.e., define a set of test vectors in case of LASAR, and define the input signals in case of SPICE.
4) Specify the nature of outputs; what signals need to be plotted and/or postprocessing the output.
5) Evaluate additional features; typical simulators have advanced features that are used in more complex applications.

III. Pedogogical Strategy for Teaching Simulations

The pedagogical objectives in teaching the use of a simulator is to enable the student to develop a thorough appreciation for the tool and how the simulator can be used to enhance his basic understanding of the underlying concepts. Understanding should go beyond providing superficial instruction in using the tool, such as providing a list of all the possible commands and their functions. More in-depth understanding would enable the student to use the tool more creatively as he progresses towards designing new circuits or systems. Furthermore, the knowledge imparted should augment but not substitute for the necessary analytical skills.

The training methodology adopted for teaching simulation is based on decomposing the process of setting up and running simulations into a series of subprocesses. Each of the stages in the generic simulation model is a subprocess. This decomposition allows the simulation teaching methodology to be segmented according to the guidelines outlined in the previous subsection. The student is provided with instruction on each of the individual subprocesses, and must display sufficient proficiency by taking a test on that section. He is then asked to develop a simulation model while concentrating only on that aspect of the simulation. For example, in the training system for LASAR, the first stage in the simulation process is to create a description or model of the simulation, i.e., to create a netlist

TABLE I
SIMULATION STAGES WITH THE PEDAGOGICAL OBJECTIVES FOR EACH STAGE

STAGES IN TEACHING SIMULATION	PEDAGOGICAL OBJECTIVES
Introduction	• General overview and functioning of the simulator. • Domain theory; digital logic, electronic circuits.
Create simulation model	• Understand the syntax of the simulator. • Study and understand the model to be simulated.
Environmental characteristics	• Understand the semantics of the simulator. • Study the environmental and global effects on the simulation model.
Specify inputs and outputs	• Study how the data can be extracted and interpreted from the simulator. • How to extract information from the specific model. Use various examples to illustrate the wide range possible.
Advanced information	• Work with the advanced features of the simulator. • Use complex examples to illustrate these features.

description of the logic network that needs to be simulated. The user is given a simple logic network, and asked to develop the corresponding netlist. This stage is successfully completed if the model is compiled using the netlist compiler with no syntax errors. For the SPICE simulator, this stage involves teaching the student about how to create syntactically correct SPICE circuit models. The next stage in the simulation process is to specify a correct set of inputs or test vectors, which are then simulated using the previously compiled model.

Table I shows the various stages in teaching the use of a simulator along with the pedagogical objectives for each stage. These objectives link understanding of how the circuit functions along with the stages in the simulation. For each stage, there are two objectives: 1) a general objective pertaining to the simulator use; and 2) explicating on the specific simulation models or circuit examples.

IV. General Architecture

A vital task in the development of intelligent tutoring systems deals with addressing the problem of *knowledge communication* [20]. The emphasis on knowledge communication is based on the pedagogical goals of tutoring, i.e., the efficient communication of knowledge or information from the teacher to the student. A computer-based tutoring system should be able to effectively represent knowledge about the specific domain and knowledge about the student, while providing a means of conveniently communicating this information to users, both instructors and students. The theme of knowledge communication stresses two architectural issues that need to be addressed in an intelligent tutoring system:.

1) knowledge representation.
2) communication with the user.

The issue of *knowledge representation* deals with developing a model of the domain, and employing a suitable formalism to encode the domain model in the tutoring system, i.e., developing a model of *communicable knowledge* [20]. A representation methodology developed in the field of artificial intelligence that is appropriate for this domain is the *Semantic Network* or *Associative Network* [10]. A network-based representation is composed of nodes which are connected by links. The nodes represent real world concepts, and the links represent relationships between them. Links also convey meaning about how the two connected concepts are related. Network representations encode information by extensive use of inheritance links, i.e., specific descriptions are created by specializing a more general description. These levels of description are related via inheritance links.

The second issue to be considered is the development of an easy- to- use and convenient mechanism for communication with the student. An ITS needs to interface with two categories of users: 1) instructors developing the course and monitoring the students; and 2) the students learning the material. This issue necessitates the design of two models of communication processes: an authoring facility for the instructor and an access tool for the student.

Fig. 1 shows the general architecture of the tutoring framework. The architecture embodies a representation technique based on semantic networks and a direct manipulation graphics interface [12] for user interaction. The architecture is comprised of two major functional blocks: 1) the knowledge representation environment; and 2) the student interface modules. Instructors and students working with an instance of this framework operate on the same knowledge network. The instructor can make changes to the network configuration or modify node contents. These changes are reflected to all or some of the students, and can be controlled via student models. In the next section, the general characteristics of the underlying knowledge representation and the user interface paradigms are explained. A description of the major functional blocks is then given.

A. The Knowledge Representation Environment

Domain knowledge about the structure and operation of a simulator is encoded into a network using NRL, a network representation language [1], [3], which was specifically designed to provide the representation support needed in a tutoring system. NRL provides a set of representation primitives of nodes and links, with which a user can build semantically consistent representations of a specific simulator. The network is composed of associated concept nodes and attribute nodes. Complete textual and graphical descriptions that explain a concept or function of the simulator are stored at each concept node. Supplementary information and examples pertaining to that concept are provided by attribute nodes that are aggregated within a context structure assigned to a concept node. A context structure provides a mechanism by which different sets of attributes can be defined about a concept for various user categories. A student in the initial stages would need to only have cursory information about a particular aspect of the simulator; the details would be relevant at a later stage. Thus, contexts provide a mechanism for selective information hiding so that the user is not overwhelmed initially with excess information about the entire simulator.

NRL also provides an interactive graphics -based -browser editor. This interface serves as a knowledge acquisition aid by which an expert (in this case, the instructor) can interactively encode the domain model into an associative network. The underlying representation is presented to the user as a labeled graph. The nodes are represented by labeled icons, and links by directed arcs. This interface modality is based on the hypertext [9] user interface paradigm and has the following features.

- Permits viewing the entire network representation visually.
- Provides navigational facilities to browse through the network by tracing associations to other nodes.
- Stores and displays annotated descriptions at nodes.
- Allows embedded simulation examples and tests at various sections in the network.
- Allows interactive alteration of the network structure by creating new links or deleting existing links.

The graphics-based interface simplifies the task of knowledge acquisition and enhances the level of communication with the user. The use of a graphics-based dialog schema, implemented via a direct manipulation interface, provides a simple alternative to a natural language interface in communicating with students.

B. Paradigms Used in the User Interface Design

Central to any ITS is the user interface and the style of presentation. If an interface is easy to learn and simple to use, instructors and students can concentrate on developing the course curricula or using the system without expending effort in learning the programming environment in which the system is implemented. A user interface design paradigm that has been rapidly evolving due to the availability of high-resolution graphics workstations is the *direct manipulation interface* [12], [13]. Direct manipulation interfaces make up an essential component of the *Hypertext systems* [9]. Characteristics of these two paradigms described in the next two paragraphs are incorporated in the general architecture.

Hypertext systems provide a general framework for information management. This technology provides an environment in which information can be stored and retreived from a nonlinear database in a more or less random manner. Hypertext systems are made up of two components: a database facility and a convenient user interface for access. The hypertext database is comprised of nodes containing chunks of textual or graphical information. These nodes have within themselves links to other nodes, thereby giving the database a directed graph structure. The database structure can be hierarchical, graph-based [nonhierarchical], or a mixed structure depending on how the information is to be organized. A typical hypertext database can be viewed as a loosely fragmented network.

The hypertext interface modality centers around sophisticated browsers and editors. The browser provides navigational facilities through link tracing, where links are expressed as visual cues, in order to search through the database. The entire database, or parts thereof, is displayed as a directed graph of nodes as icons, and links between nodes as directed arcs. From this visual description, the user can initiate various operations as required by the application in any portion of the database. Editing functions allow the user to set up or modify the underlying database. By altering the displayed graph structure, the corresponding changes are made to the database network. This technique of visually editing graphically displayed objects by using a pointing device, such as a mouse, constitutes a direct manipulation interface [12]. Direct manipulation interfaces free the users, both instructors and students, from the burden of learning a programming language. The user can put together applications by using a library of sufficiently general visual objects, such as icons representing nodes. These objects can be further augmented by ordinary text or graphics images. Contents are added to the nodes by invoking text or graphics editors at the desired nodes. Another feature is procedural attachment to the displayed graph in the browser/editor. This feature enables assigned procedures to be invoked depending on the user interaction, i.e., the mouse selection at a node would cause the node contents to be displayed or a query to be initiated, or a section of the network to be zoomed.

C. User Interface Modules

1) Authoring Shell for Instructors: The instructor interface in the SPICE tutor is shown in Fig. 2. This interface allows a preauthorized instructor to add more information, modify the course structure, or to encode the complete description for a new simulator. The instructor editor also has a student record inspector that allows the instructor to monitor a student. The student record is a history mechanism that stores various attributes about a student: the portion of the course completed, performance on the given tests, time spent, etc. The authoring shell user interface is a composite window structure made up of four subwindows, each being an editor. The major portion of the shell is occupied by the main network editor, in which an instructor can develop an outline of the course contents as a network. The contents subwindow provides a means to add text contents to each of the nodes; text files generated using other sources can be linked to the nodes. The questions editor subwindow allows an instructor to assign tests to a node made up of multiple choice questions. Additionally, a set of hidden nodes can be linked to a node. These hidden nodes serve to provide extra cues or supplementary information that would be useful to the student.

All the operations an instructor needs to use while setting up the course are presented in the form of hierarchical pull-down menus. This direct manipulation interface obviates the need to define a syntax. Fig. 2 also includes a snapshot of an instance of the menu which defines all the actions that can be used to build a network. The operations are categorized under Net, Node, Link, and Student operations. The first three categories of operations enable the instructor to create new knowledge networks, modify specific components such as a specific node, or redefine a new link. Selecting the menu option Node provides the user with the two possible categories of operations: 1) network operations include functions for creating new nodes or altering the existing network; and 2) Edit/Display contents allow an instructor to add or modify

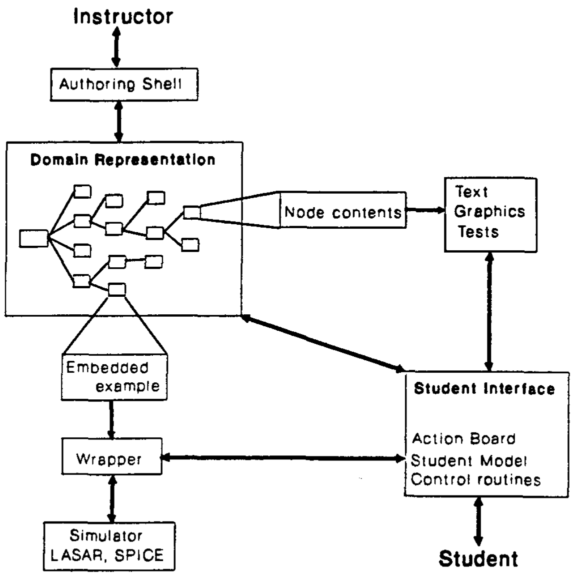
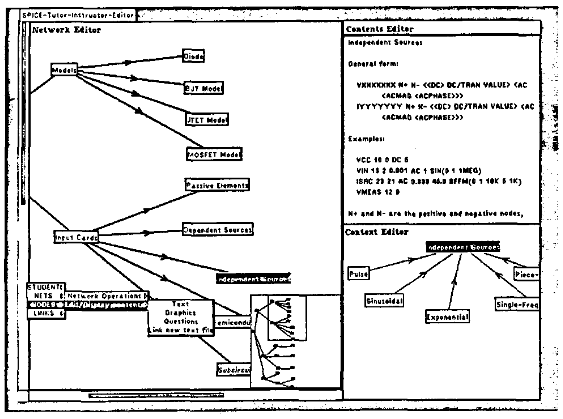


Fig. 1. General architecture.

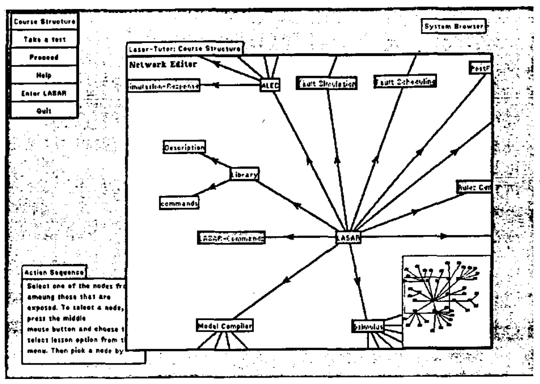


```
Independent Sources

General form:

   VXXXXXXX N+ N- <<DC> DC/TRAN VALUE> <AC
         <ACMAG <ACPHASE>>>
   IYYYYYYY N+ N- <<DC> DC/TRAN VALUE> <AC
         <ACMAG <ACPHASE>>>

Examples:

   VCC 10 0 DC 6
   VIN 13 2 0.001 AC 1 SIN(0 1 1MEG)
   ISRC 23 21 AC 0.333 45.0 SFFM(0 1 10K 5 1K)
   VMEAS 12 9

N+ and N- are the positive and negative nodes,
```

Fig. 2. Instructor interface.

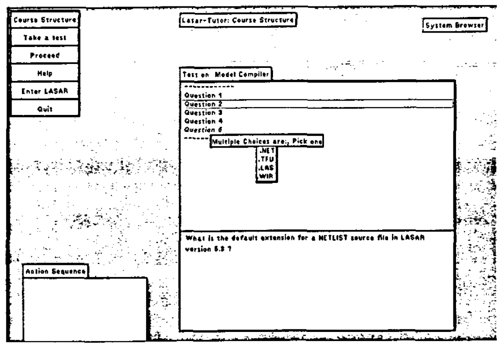
contents of a desired node. Student operations provide a means of inspecting and browsing the performance of each student Histograms are used to relate various parameters such as nodes completed, time taken, and scores on tests, for each student. Histograms for comparative evaluations of students as a group are also made available.

2) Student Interface: Fig. 3. shows the student interface. The control strategy is student driven, with the system monitoring the actions in the background to ensure that the actions taken are within the permissible boundaries. The student is provided with various functional options via an action board. The action board is a panel of buttons, with procedures attached to each button. These procedures operate on the current student model, and update the displays accordingly by exposing new nodes for example. A loose control flow of the tutoring strategy is programmed into the action board. The control flow ensures that the student does not preempt any of the necessary steps,



```
Select one of the nodes fr
ameung those that are
exposed. To select a node,
press the middle
mouse button and choose t
select lesson option from t
menu. Then pick a node by
```

Fig. 3. LASAR tutor student interface.



What is the default extension for a NETLIST source file in LASAR version 5.3 ?

Fig. 4. LASAR tutor student test interface.

such as taking all the embedded tests. Tests are made up of a set of multiple choice questions, and are presented to the user in a direct manipulation type of interface. Fig. 4 shows the test interface. In this figure, the top subwindow is a selection-in-list window, which displays a list of questions. The student goes through selecting a question using the mouse, and the multiple choices are presented in the form of a pop-up menu. The student is also given the freedom to select any node from among those exposed. This form of an open interface also allows the student to inspect more than one node at a time.

The interface to the actual simulator underlying an instance of the tutoring system is provided by a module called *the wrapper* [19]. The wrapper is an intelligent interface between a simulation tool and a user environment. Example nodes that require the student to develop actual simulations invoke the wrapper whenever the student explores that node.

D. Implementation

Throughout this paper, we have presented screen dumps of the actual systems implemented for LASAR and SPICE. This general framework is designed in an object-oriented manner, and implemented in the programming environment Smalltalk-80 [11]. Smalltalk-80 runs on workstations (Sun, Hewlett-Packard) and personal computers (Apple and 386-based machines), requiring a minimum of 8 mbytes of memory. Smalltalk-80's portability allows a single implementation to be easily ported across all of these computer systems. The wrapper has remote process communication features that enable the actual simulation process to be run remotely, either in the background or on a remote server, thereby enabling the user to proceed with other tasks.

V. Operational Characteristics

A. User Driven Control

The operational principle underlying the tutoring system architecture is based on a continuous mixed initiative dialog with the user. This mode of operation allows the user a certain degree of freedom to intuitively explore a restricted space. The limited access ensures that attention is focused on a certain aspect of the simulator. The user's actions are monitored within this space. Systems like SCHOLAR [7] review knowledge about a student by using a mixed initiative dialog conducted through a natural language interface. A similar strategy is adopted in this framework, except that communication is through manipulation of visual objects. Such a strategy also allows for localized remedial actions to be attached to each node.

A student proceeds with the tutorial by navigating through the knowledge network. A segment of the network is presented to the user and the performance of the user within this segment is measured. Once the performance crosses a predefined threshold, signifying a measure of confidence or familiarity with the current aspect, more links to other parts of the network are exposed. The student's access to the course contents is visually guided by coloring the nodes. Nodes that have been successfully completed are grayed lightly, the nodes that need to be completed in order to progress to the next section are fully displayed, and nodes not yet accessible are blacked out.

B. Learning by Doing

The goal of the system is to train a user to generate simulations with a minimum of difficulty. A pedagogical approach towards achieving this goal is to adopt a teaching strategy wherein the user has to generate working simulations during the learning process. Thus, hands-on experience is provided while the student is learning about the simulator. These examples are presented as tests at various stages of the course structure. Since the examples are stored in nodes, access to the examples is controlled in the same manner as for lessons. As the student completes the task of generating a simulation for a simple circuit, based on a particular aspect of the simulator, other nodes become accessible. These nodes include the other features of the simulator as well as more advanced examples. The practice obtained in generating a complete working simulation gradually, through these various stages, ensures that a student fully understands every aspect of the simulation.

C. Performance Monitoring and Comparative Evaluation

Access to the tutorial is restricted by assigning a password to each student. A student record is maintained for each registered student. The student record includes relevant personal information and performance parameters. The performance parameters are the number of nodes completed, the scores obtained on the local embedded tests, and time spent on each lesson and within a level. These parameters provide useful information in making inferences on how the student is progressing. The student record serves as a model of the current student. The control routines tied into the student interface continuously update the student record to reflect the new state. The student record is also referenced to make inferences as to when it is appropriate to advance to a higher level of course contents. The instructor is provided with various histogram tools to analyze each student, as well as to obtain a comparative view of the group as a whole. The forms of evaluations, both individual and comparative, are useful in developing various group models. These models can be generalized according to student categories such as sophomores, juniors, etc. For each category, an instructor can define an overlay structure, which is superimposed over the knowledge base, and acts as a filter to allow the appropriate level of information to be presented to a student group. This monitoring strategy also provides a way of individualizing the instruction to specific student needs. The instructor can post remedial hints and offer suggestions to each student on the basis of how he is doing.

VI. Conclusions and Future Directions

The research efforts reported in this paper were directed towards the architectural development of the framework for building ITS's for simulation systems. This framework was demonstrated by building ITS for LASAR and SPICE. The use of these simulators in undergraduate engineering education has provided a useful method for instruction. Quantitative evaluation of the effectiveness of the ITS's in this paper have not as yet been made. Effective evaluations require the careful construction of controlled experiments, which has been difficult to achieve in our environment. The ITS's have been used by various populations, including industrial engineers, graduate students in electronics, and undergraduates. Initial responses from those using these tutorials have encouraged us to pursue the development of this system. Knowledge is more easily accessible than through the use of the traditional instruction manual. An oft-cited negative is the need to learn the complexities of the direct manipulation interface. After the initial learning sessions, these complaints disappear, though efforts are being made to further simplify the user interface. The increasing popularity of windowing environments should also help to ameliorate these criticisms.

We have also begun applying the framework to other domains besides simulators. Future efforts are directed towards enhancing the student model, and the diagnostic inference procedures which would help in providing more individualized instruction. Another direction open to further research is making diagnostic inferences on the simulations developed by the student in comparison to an ideal model, and offering hints that would help in improving the quality of the simulation.

Acknowledgment

The authors would like to acknowledge the efforts of S. Tso of Northern Telecom, Inc., who helped in developing

the knowledge base for LASAR, and H. Van Der Molen, who developed the interface to the simulator—the *wrapper*. Teradyne Corporation provided their LASAR simulator.

References

[1] B. Antao, "NRL: A network representation language for representing knowledge in advisory systems," M.S. thesis, Dept. Elect. Eng., Vanderbilt Univ., Dec. 1988.
[2] B. Antao *et al.*, "An advisory system for digital logic simulation," presented at Proc. Sec. IEA/AIE Conf., June 1989.
[3] B. Antao, "NRL: Implementation description," tech. rep. CIS-89-01, Center for Intelligent Systems, Vanderbilt Univ., Nov. 1989.
[4] P. Antognetti and G. Massobrio, *Semiconductor Device Modeling with SPICE.* New York: McGraw Hill, 1988.
[5] W. Banzhaf, *Computer-Aided Circuit Analysis Using SPICE.* Englewood Cliffs, NJ: Prentice-Hall, 1989.
[6] J. R. Bourne *et al.*, "Intelligent hypertutoring in engineering," *Acad. Comput.*, Sept. 1989.
[7] J. R. Carbonell, "AI in CAI: Artificial intelligence approach to computer assisted instruction," *IEEE Trans. Man-Mach. Syst.*, no. 4, Dec. 1970.
[8] W. J. Clancey, *Knowledge-Based Tutoring: The GUIDON Program.* Cambridge, MA: M.I.T. Press, 1987.
[9] J. Conklin, "Hypertext: An introduction and survey," *IEEE Comput.*, pp. 17–41, Sept. 1987.
[10] N. V. Findler, *Associative Networks: Representation and Use of Knowledge by Computers.* New York: Academic, 1979.
[11] A. Goldberg and D. Robson, *Smalltalk80: The Language and Its Implementation.* Reading, MA: Addison Wesley, 1983.
[12] E. L. Hutchins, J. D. Hollan, and D. A. Norman, "Direct manipulation interfaces," *User-Centered System Design: New Perspectives on Human-Computer Interaction,* D. A. Norman and S. W. Draper, Eds. Lawrence Erlbaum Associates, 1986, pp. 285–313.
[13] B. Shneiderman, "The future of interactive systems and the emergence of direct manipulation," *Behavior and Information Technology,* vol. 1, no. 3, 1982.
[14] D. Sleemen and J. S. Brown, *Intelligent Tutoring Systems.* New York: Academic, 1982.
[15] Teradyne Corporation, *LASAR Version 6 Software,* 1988a.
[16] Teradyne Corporation, *LASAR Version 6 Software Users Guide,* 1988b.
[17] Teradyne Corporation, *Windows,* vol. 1, no. 2, Apr. 1988.
[18] A. Vladimirescu *et al., SPICE Version 2G Users Guide,* 1981.
[19] H. Van der Molen, "The LASAR wrapper: An intelligent interface for a digital simulation tool," M.S. thesis, Dep. Elect. Eng., Twente Univ., May 1989.
[20] E. Wenger, *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge.* Morgan Kaufman, 1987.

Brian A. A. Antao (S'85) was born in Bombay, India, on June 13, 1964. He received the B.E. (Hons.) degree in electrical engineering from the Victoria Jubilee Technical Institute at the University of Bombay, Bombay, India, in 1986, and the M.S. degree in electrical engineering from Vanderbilt University, Nashville, TN, in 1988.

Since 1986, he has been a Research Assistant with the Center for Intelligent Systems, Department of Electrical Engineering, Vanderbilt University, where he has been involved in research and consulting in the area of applied artificial intelligence. His research interests include all aspects of design automation for VLSI circuits (specifically the synthesis, simulation, and optimization of analog and mixed mode circuits, CAD tool frameworks, and object-oriented techniques) and artificial intelligence (specifically design methodologies).He is currently working towards the Ph.D. degree in electrical engineering at Vanderbilt University.

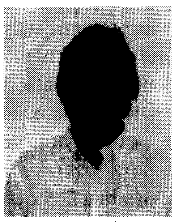
Mr. Antao is a member of the American Association for Artificial Intelligence (AAAI).

Arthur J. Brodersen (SM'76) received the B.S., M.S., and Ph.D. degrees from the University of California, Berkeley, in 1961, 1963, and 1966, respectively, all in electrical engineering.

He was on the faculty at the University of Florida, Gainesville, from 1966 to 1974. He has been at Vanderbilt since 1974 and has served as Chairman of Electrical and Biomedical Engineering and Associate Dean of the School of Engineering. He is currently Professor of Electrical Engineering and a member of the Center for Intelligent Systems. In recent years, his research interests have been centered in applying expert system technology to the diagnosis and repair of electronic systems and in intelligent training systems for engineering education.

John R. Bourne (S'66–M'70–SM'77) received the B.E. degree from Vanderbilt University, Nashville, TN, in 1966, and the M.S. and Ph.D. degrees from the University of Florida, Gainesville, in 1967 and 1969.

He has been a member of the faculty of Vanderbilt University since 1969, where he is currently Professor of Electrical Engineering and Director of the Center for Intelligent Systems. During 1982, he was a Visiting Professor at Chalmers University, Gothenburg, Sweden. He is Editor of the *CRC Critical Reviews in Biomedical Engineering* and has served in various other editorial capacities.



Jeffrey R. Cantwell (S'82–M'85) received the B.A. degree from the University of North Carolina, Greensboro, in 1979, and the M.S. degree in educational psychology from Florida State University, Tallahassee, in 1984.

He is a Research Instructor in the Department of Electrical Engineering at Vanderbilt University, Nashville, TN, and has been a member of the affiliated faculty with the Center for Intelligent Systems since 1985. His research interests include intelligent tutoring systems, hypermedia, industrial training, object-oriented programming, and user modeling. He has served as a reviewer for *IEEE Expert.*

Mr. Cantwell is a member of the American Association for Artificial Intelligence.