Chi, M., Feltovich, P., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5,* 121-152.

Eisenstadt, M., Laubsch, J., & Kahney, H. (1981). Creating pleasant programming environments for cognitive science students. In *Proceedings of the Third Annual Cognitive Science Society Conference.* Berkeley, CA: Cognitive Science Society.

Garlan, D. B., & Miller, P. L. (1984, April). GNOME: An introductory programming environment based on a family of structure editors. In *Proceedings of the Software Engineering Symposium on Practical Software/Development Environments.*

Johnson, W. L. (1986). Intention-based diagnosis of errors in novice programs. Palo Alto, CA: Morgan Kaufman.

Johnson, W. L., & Soloway, E. (1985) Micro-PROUST. Yale University Computer Science Department Research Report No. 402. New Haven, CT: Yale University.

Mayer, R. E. (1979). A psychology of learning BASIC. *Communications of the Association for Computing Machinery, 22*(11), 589-598.

Miller, L. A. (1981). Natural language programming: Styles, strategies, and contrasts. *IBM Systems Journal, 20,* 184-215.

Reiser, B., Anderson, J., & Farrell, R. (1985). Dynamic student modeling in an intelligent tutor for LISP programming. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* 8-14.

Resnick, L. (1983). A new conception of mathematics and science learning. *Science, 220,* 477-478.

Spohrer, J., Soloway, E., & Pope E. (1985). A goal/plan analysis of buggy PASCAL programs. *Human-Computer Interaction, 1,* 163-207.

Soloway, E., & Ehrlich, K. (1984). Empirical studies of programming knowledge. *IEEE Transactions of Software Engineering, SE-10,* 595-609.

In Psotka, J., Massey, L. D., & Mutter, S.A. (eds), Intelligent Tutoring Systems: Lessons Learned. Lawrence Erlbaum, Inc., 1988.

Feurzeig, W., & Ritter, F. (1988). Understanding reflective problem solving. In J. Psotka, L. D. Massey & S. A. Mutter (Eds.), Intelligent tutoring systems: Lessons learned. 435-450. Hillsdale, NJ: Erlbaum.

16

Understanding Reflective Problem Solving

Wallace Feurzeig
Frank Ritter
BBN Laboratories

STUDENT MODELING AND THE DIAGNOSIS PROBLEM

How can an ICAI system make plausible hypotheses concerning a student's knowledge state about a problem domain? The task of the diagnostic module of the system is to make intelligent inferences about the student's knowledge, knowledge gaps, surface bugs, and, if possible, the associated underlying misconceptions. The essential starting point for such deep inferences is the observations of the student's task performance — the sequence of actions taken by the student as he or she works on a problem. This constitutes the initial knowledge base of the student diagnostic module. This surface performance information is necessary for diagnosing and characterizing faulty behaviors, but it is not sufficient even for diagnosing student difficulties in relatively simple intellectual tasks such as the performance of simple arithmetic computations using prescribed algorithmic procedures.

The diagnosis problem has been addressed by a number of ICAI systems, including WEST, DEBUGGY, SOPHIE, and QUEST. The concept of a differential student model was developed in WEST (Burton & Brown, 1982), a computer board game designed to teach computational skills through computation-based game-playing strategy. The approach to diagnosis in WEST is to model the problem performance of an expert player and to contrast that with the observed performance of the student working on the same problem. This kind of performance analysis can identify weaknesses in the student's play, but not the underlying difficulties responsible for them. For example,

a poor move might be due to the student's failure to consider an alternative move or to an incorrect computation of a move, two distinctly different kinds of difficulties calling for qualitatively different instructional treatments.

DEBUGGY (Burton, 1982) is the instructional form of the well-known BUGGY system for modeling the procedural bugs accounting for most student subtraction errors. In DEBUGGY, the diagnosis of a student's procedural bugs is done using a pattern-matching scheme. DEBUGGY incorporates a substantial data base of subtraction problem bugs—faulty subtraction procedures obtained from empirical studies of subtraction problem work across large student populations. If a student's performance across a set of representative problems is identical to that of a buggy procedure in the data base, the system identifies the buggy procedure as the student's bug. This approach is severely limited to those relatively simple types of problems for which there is a small enough set of distinct types of bugs to permit their explicit enumeration. It is not a feasible diagnostic methodology for the problems of typical interest and complexity such as tactical military tasks.

SOPHIE (Brown, Burton, & Bell, 1974), a pioneering ICAI system for electronics troubleshooting training, used a general circuit simulation program as a dynamic knowledge base for evaluating the behavior of the circuit under working or faulted conditions. A substantial part of the understanding capabilities in the SOPHIE ICAI system was based on its use of this mathematical simulation model, a general purpose circuit simulation called SPICE (Nagel, 1975), together with a LISP-based functional simulator incorporating circuit dependent knowledge. These facilities were essential for inferring complex circuit interaction sequences such as fault propagation chains. SOPHIE's capabilities for modeling and understanding causal chains of events formed the basis for its powerful explanation and question-answering facilities.

SOPHIE used the simulator to make powerful deductive inferences about hypothetical, as well as real, circuit behavior. For example, it determined whether the behavior of the circuit was consistent with the assumption of specified faults and whether a student's troubleshooting inferences were warranted, that is, whether the student had acquired information of the voltage and current states of relevant circuit components sufficient to unambiguously isolate the fault.

SOPHIE could infer what the student should have been able to conclude from his or her observations at any point. For example, it determined the currently plausible hypotheses and those that were untenable. However, because SOPHIE did not determine the reasons for the student's tests and measurements, the hypotheses the student

was actually considering, it could not tell whether his or her conclusions were based on logically complete and consistent reasoning. It was unable to diagnose the student's specific misconceptions or difficulties in understanding circuit behavior or in troubleshooting faults. Despite these deficiencies, SOPHIE was one of the first ICAI systems capable of supporting compelling and educationally effective instructional interactions.

There is a straightforward approach to improving a system's knowledge of a student's thinking during a problem-solving interaction. Instead of basing the system's inferences solely on the external actions taken by the student (for example, voltage measurements or continuity tests in electrical troubleshooting), the system can also attempt to elicit associated information concerning the student's hypotheses, goals, and plans. This approach has been developed in the context of the QUEST instructional system.

AN OVERVIEW OF QUEST

QUEST (Qualitative Understanding of Electrical System Troubleshooting) is a current ICAI system for teaching electrical system troubleshooting (White & Frederiksen, 1986; 1987). QUEST uses qualitative simulation methods (White & Frederiksen, 1985; Ritter, 1986) to teach knowledge-based reasoning about circuit behavior and troubleshooting. Humans think about the behavior of phenomena and systems in a qualitatively different way from that used to describe such behavior in mathematical simulation models. Experts in a domain (not only beginning students) use qualitative modes of thought and qualitative models to reason about system behavior. Thus, though it is necessary to employ mathematical simulations to obtain precise detailed descriptions of system behaviors, we also want to teach conceptually sound qualitative reasoning. The use of qualitative simulation models is valuable for producing understandable explanations and for generating animated displays to show dynamic behavior in a clear manner. This facilitates learning by fostering the student's development of effective mental models for understanding and reasoning about system behavior.

QUEST employs a qualitative simulation model for reasoning about the behavior of simple electrical circuits composed of batteries, wires, resistors, coils, condensers, lamps, switches, and testlights. An example of a QUEST circuit is shown in Figure 16.1. The qualitative simulation includes a description of the circuit topology, a runnable function model for each device in the circuit, rules for evaluating

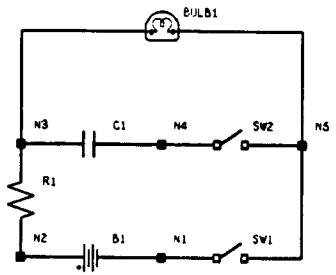

FIGURE 16.1 A typical QUEST circuit.

device states at each time increment, and circuit-tracing procedures to aid in evaluating conditions for device states. It is designed to support a dynamic presentation environment by generating graphical representations of circuit operation. An expert troubleshooting program is provided to demonstrate troubleshooting concepts and strategy within that environment. The expert troubleshooter can be called upon to solve problems and to explain its reasoning along the way. QUEST also provides an instructional mode that allows students to practice troubleshooting (Feurzeig, 1985). In this mode the instructional system provides students with a problem-solving environment within which circuits can be built, tested, and modified. When requested, the program generates qualitative explanations of circuit operation in both working and faulted states. Circuit problems given to students include predicting circuit behavior and troubleshooting faults within circuits.

When solving problems, students can call upon the qualitative simulation and expert troubleshooter programs to explain reasoning about circuit operation or troubleshooting logic. Each tutorial program utilizes a model that articulates reasoning at a level of explanation that is appropriate for the particular stage of instruction. The circuit simulation program can explain to students the operation of circuits in either faulted or working condition. Explanation of troubleshooting logic are produced by the troubleshooting expert and are coordinated in level of complexity with the explanations of circuit behavior offered by the circuit simulation.

KNOWLEDGE ACQUISITION FOR STUDENT MODELING IN QUEST

The distinctive diagnostic feature of QUEST that sets it apart from the other ICAI systems is its facility for eliciting explicit information from the student about the intended purpose of his or her actions before they are performed and about his conclusions afterward. This

interaction is carried out throughout the detailed course of the troubleshooting activity. An example of the use of the QUEST instructional monitor is shown in Figure 16.2. The scenario shows a student troubleshooting a simple circuit consisting of a battery, two resistors, a wire, and a bulb to illustrate the interaction (QUEST is also capable of modeling the dynamic behavior of capacitors and inductors in relatively complex circuits.)

This scenario represents just a few minutes of interaction during the problem-directed explanation mode of QUEST. The querying of the student proceeds effortlessly. The student is asked before each action (for example flipping a switch or inserting a test light) what he or she hopes to learn through taking this action. After his or her response, the action is carried out by the system. The student may also take other actions at this point, call the simulation to be run, and see the effects of these actions. After this, the student is asked what he has learned. Following his response, the entire process resumes with the student's next troubleshooting action. In addition to gathering data, this procedure helps shape a student's thinking and approach to circuit problems by providing an explicit model and an analytic framework.

The interface is easy to use. The student answers a question by choosing from an appropriate range of responses on a display window, clicking the mouse when it points to the response selected. Some answers require more than one response; for example, when the student wants to add a new fault, he must also select the component of the subcircuit suspected of being faulty and designate the fault. The instructor can set up the system to require that the student answer all the questions posed. At the opposite extreme, the system can be set up so that it is possible for the student to bypass the entire monitoring process.

The Quest Instructional Monitor (QUIMON) is invoked each time the student takes an action. This elicitation procedure is designed to be non-intrusive and unforced. The student is not required or even requested to be deliberative about every single action taken along the way. A more sophisticated procedure, incorporating knowledge of the circuit and utilizing the information elicited from the student about his current plans, could be designed. This kind of procedure would need to be invoked less frequently, at points corresponding to completion of a global operation sequence or to a shift in the student's current focus of attention.

The session produces a substantial knowledge base of the student's plans and goals with minimal interference to his troubleshooting activity. We believe that such finegrained information about the student's intentions, expectations, and conclusions can be uniquely

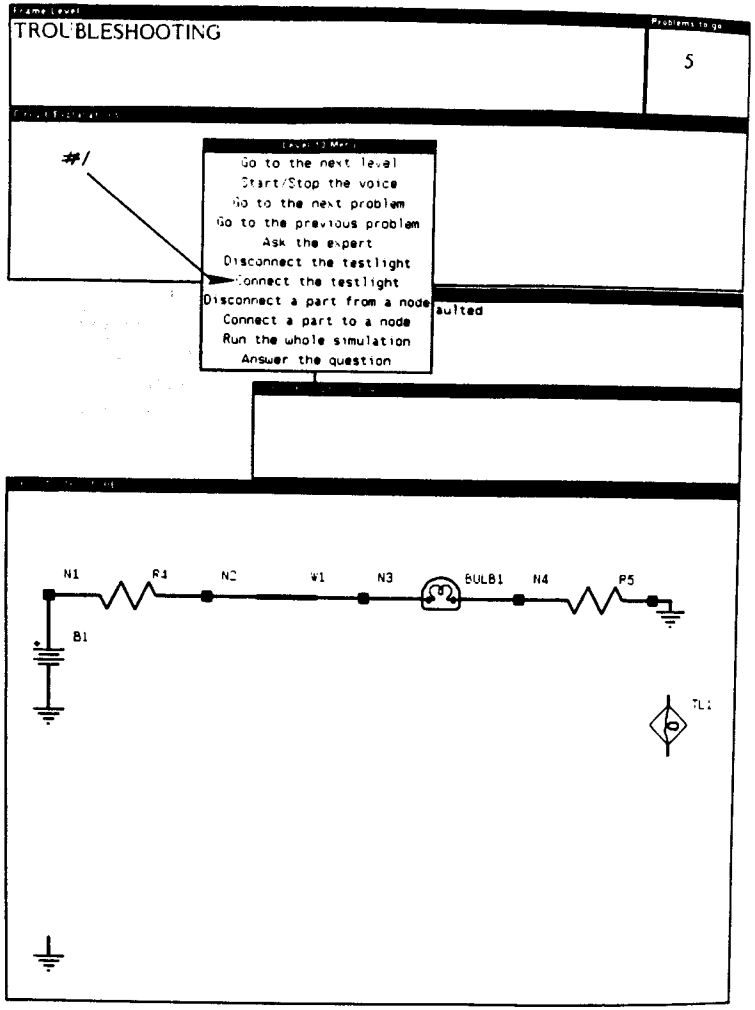

FIGURE 16.2a A student interaction with Quest Instructional Monitor.

valuable for understanding his performance and making plausible diagnoses of his misconceptions and difficulties. Moreover, such information can only be elicited from the student—it is, at the very least, extremely difficult for an ICAI system based on present AI methods to infer the student's mental states from his surface behaviors. Thus we believe that QUIMON work provides an effective starting point for development of more competent student diagnostic models.

A sample operation of QUIMON follows.

QUEST has created a circuit for the student to troubleshoot. The student has a menu from which to choose several possible actions. In Figure 16.2a, the student points an arrow (1) to "Connect the test light."

In order to model the student's behavior, a menu pops up to ask the student why he wants to insert a test light. The student is allowed to answer simply "Don't know." In Figure 16.2b, however, the student clicks where arrow 2 is pointing, "To explore general circuit behavior," indicating a lack of specificity to his actions. A student who would like to be more specific can go as far as specifying that he is testing the feed to a device. In that case, the system would prompt him for the specific device.

The system responds by entering the test light, pointed out by arrow 3, (see Figure 16.2c) into the circuit at the place where the student indicates (not shown here), and brings the student back to the action menu. The student then clicks where arrow 4 is pointing, to run the simulation of the current circuit.

The tail end of the written explanation of the qualitative simulation in QUEST, detailing the state changes of parts, and explanations of why the parts changed state, is visible in the Trace Output Window, arrow 5 (see Figure 16.2d). The spoken output is not shown, nor is the system's use of flashing in inverse video to show the paths that the simulation used in computing voltage drops. The student is asked if he wishes to continue the simulation for another clock cycle. He does not (see Figure 16.2d) arrow 6, and clicks on "Stop."

After the simulation has run, the student is asked what he or she has learned—for example, what new possible faults he has discovered. Arrow 7 in Figure 16.2e points to the list of faults identified by the student. The list of possible faults ruled out by the student is also displayed. The student can indicate that he has not drawn any conclusions from the current action, and can proceed directly back to the action menu. In this scenario, the student clicked at arrow 8 (Figure 16.2e) indicating that he or she suspects a new fault in the circuit.

The student now is asked to indicate the exact part that he or she thinks is faulted by clicking on it with the mouse. The student does so by clicking on "BULB1" (arrow 9, Figure 16.2f). The system knows the set of faults each part can have, and now pops up the menu for bulb faults. The student picks "Open" (arrow 10, Figure 16.2f).

The new fault is now displayed in the window containing possible faults (arrow 11, Figure 16.2g). The student also believes he learned

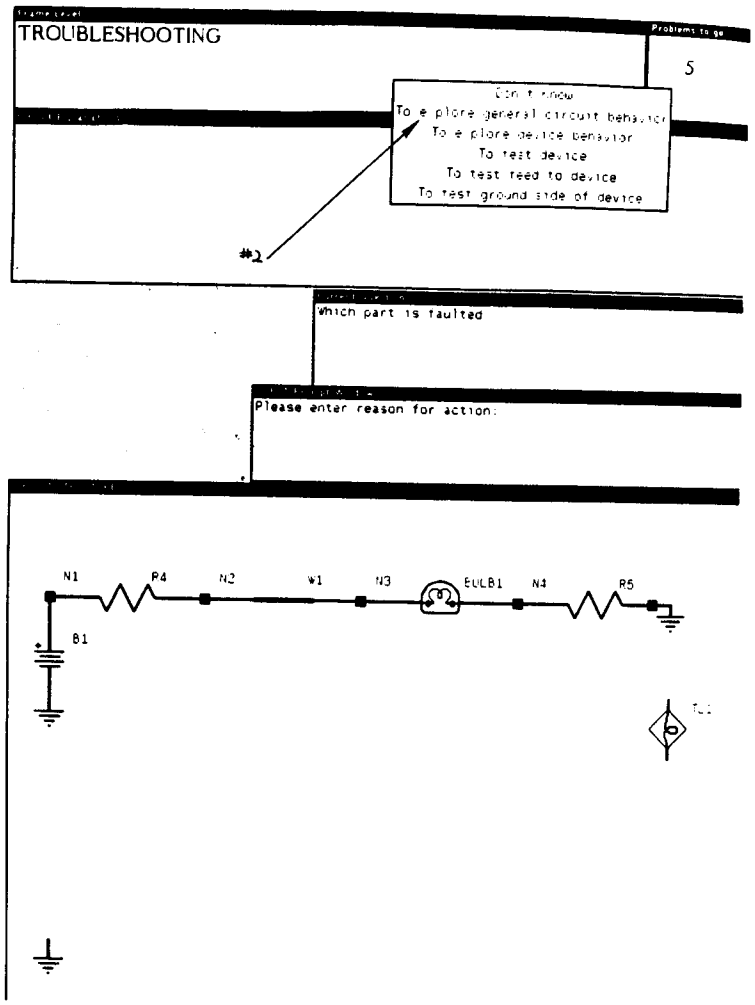

FIGURE 16.2b

that another part is unfaulted and indicates this by clicking (where arrow 12 points): "Rule out possible fault."

The student is then prompted for the part that is to be marked as not faulted. He clicks on the wire "W1," pointed to by arrow 13 in Figure 16.2g.

The choice of W1 as an unfaulted part is shown in the appropriate window. The student is returned to the menu that allows him to

specify additional information, for example other faulted or unfaulted parts. The student declines to do so, by clicking where arrow 14 indicates (see Figure 16.2h), and is returned to the action menu to continue troubleshooting.

The final figure shows the end of the session and displays the fault. The student was correct in choosing BULB1 being open as the fault (arrow 15, Figure 16.2i).

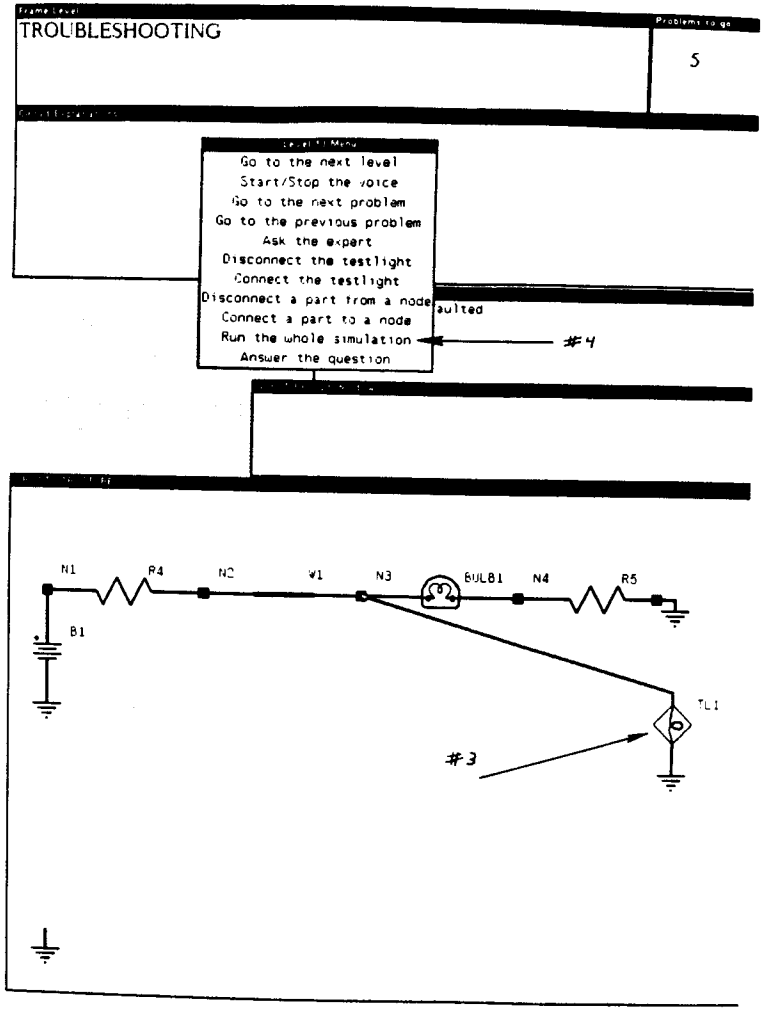

FIGURE 16.2c

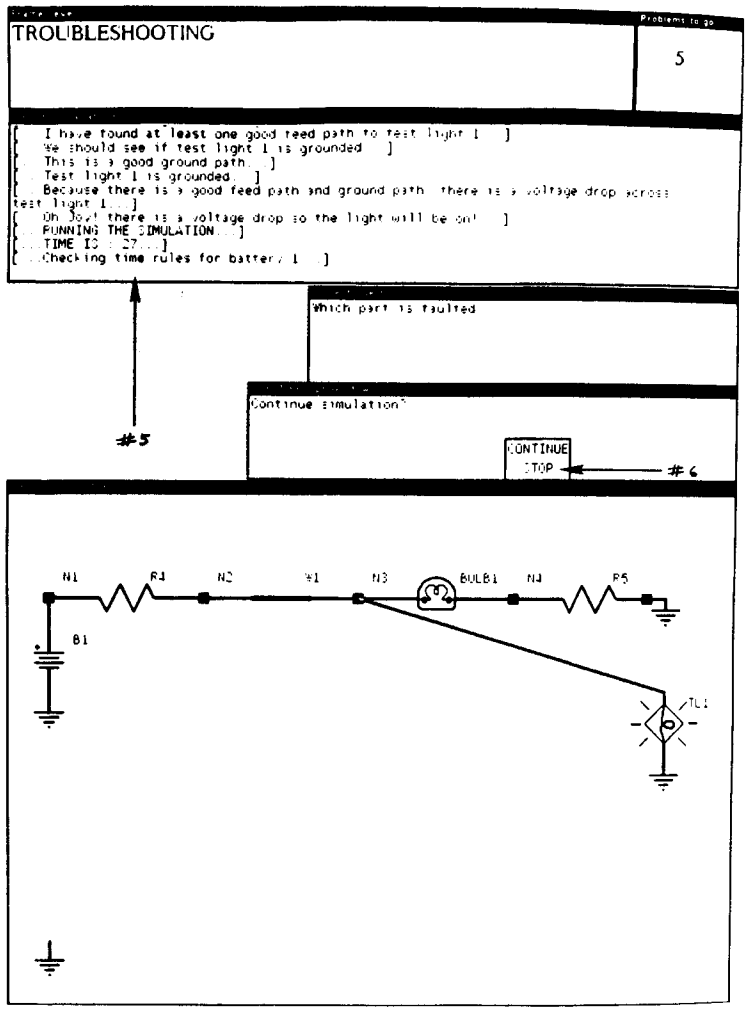

FIGURE 16.2d

CONCLUSIONS

The addition of information about the student's intentions, expectations, and plans, as well as his observed actions is, we believe, essential to making informed and insightful diagnostic hypotheses. This approach to diagnosis integrates commonsense principles from cognitive science with powerful AI inferencing methods. It

substantially enhances the power and reliability of ICAI inferencing capabilities, and it has importance for a wide range of applications to complex systems maintenance and troubleshooting training.

This approach does have limitations. In common with other approaches, it is ineffective with noncooperative students. Also, it assumes the principle of rationality, that problem-solving behavior, whether correct or not, is always rational behavior even when based

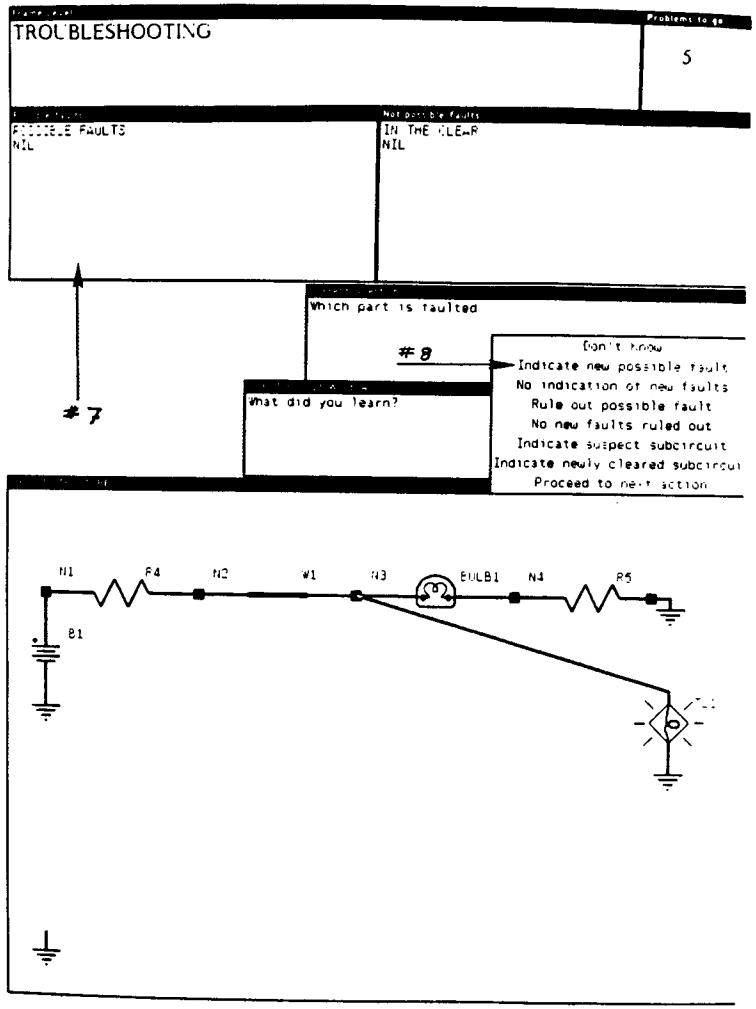

FIGURE 16.2e

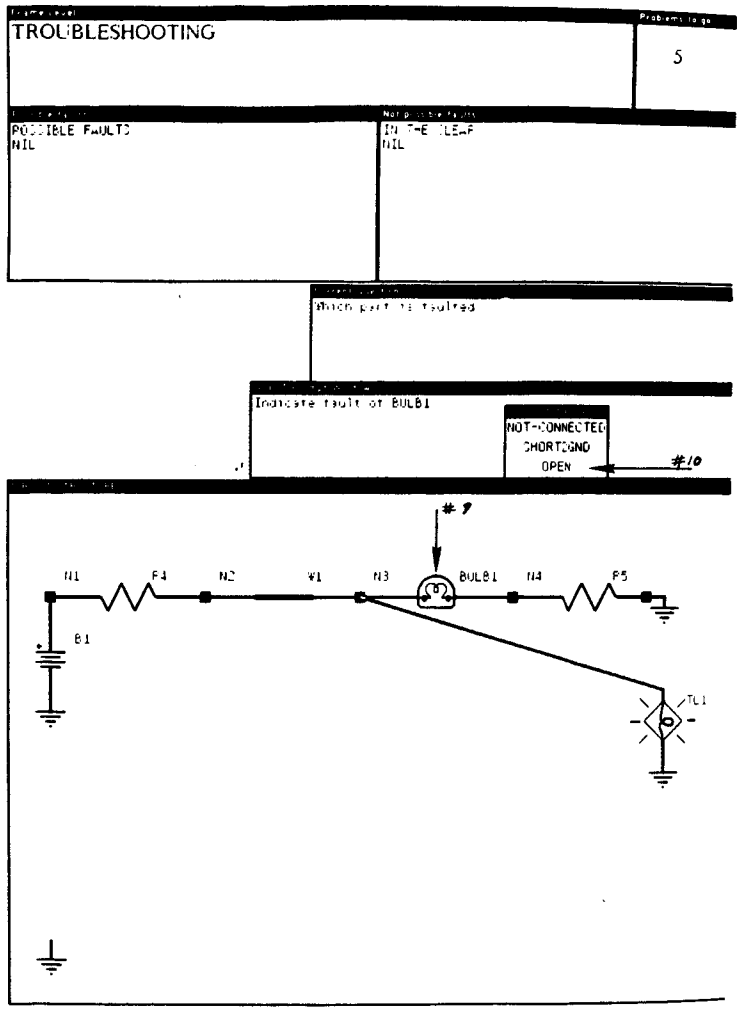

FIGURE 16.2f

on incorrect knowledge. Further, the elicitation procedure that is central to it will not be applicable in problem-solving situations where students lack either the knowledge or an appropriate vocabulary or language for talking about their problem-solving plans and goals. For example, beginning students of mathematics may be ill-equipped to discuss their strategy, or even their step-by-step actions, in performing tasks such as solving equations. This problem can be

mitigated in QUIMON by placing items the student should be considering on the menu to prompt his or her thinking. Finally, in certain real-time situations, where tasks have to be performed on the fly, there is little time or attention available for the kind of interventions required to discuss actions, much less their antecedents and consequences. Other kinds of intelligent instructional methods must be developed to deal with such real-time interactions.

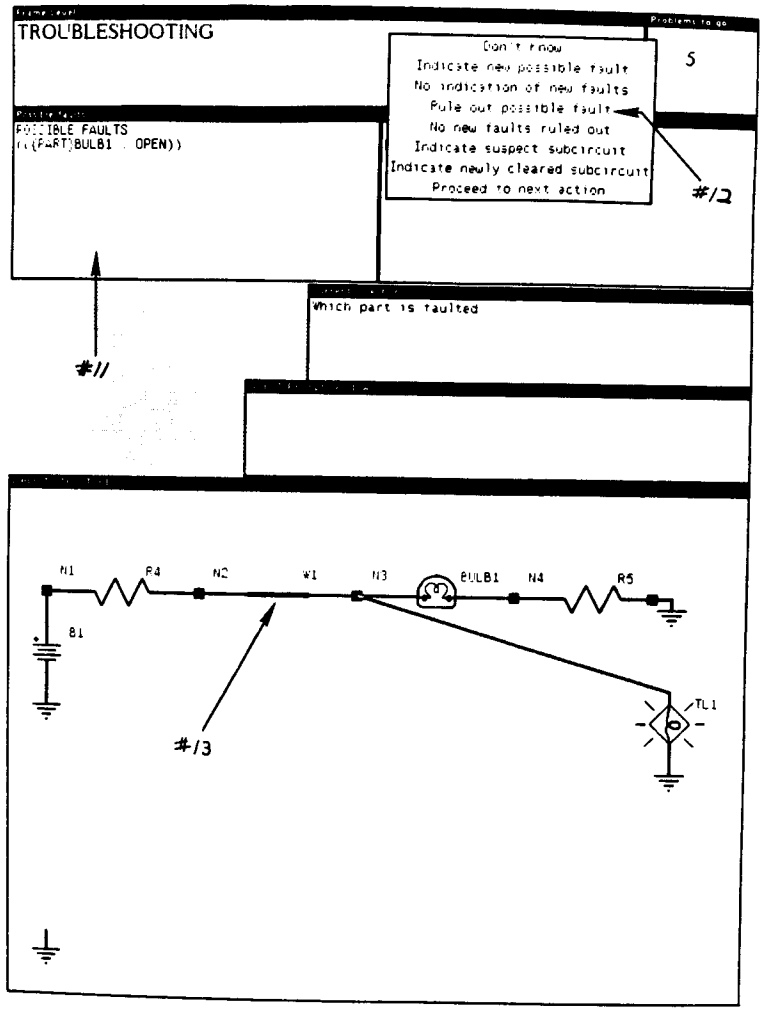

FIGURE 16.2g

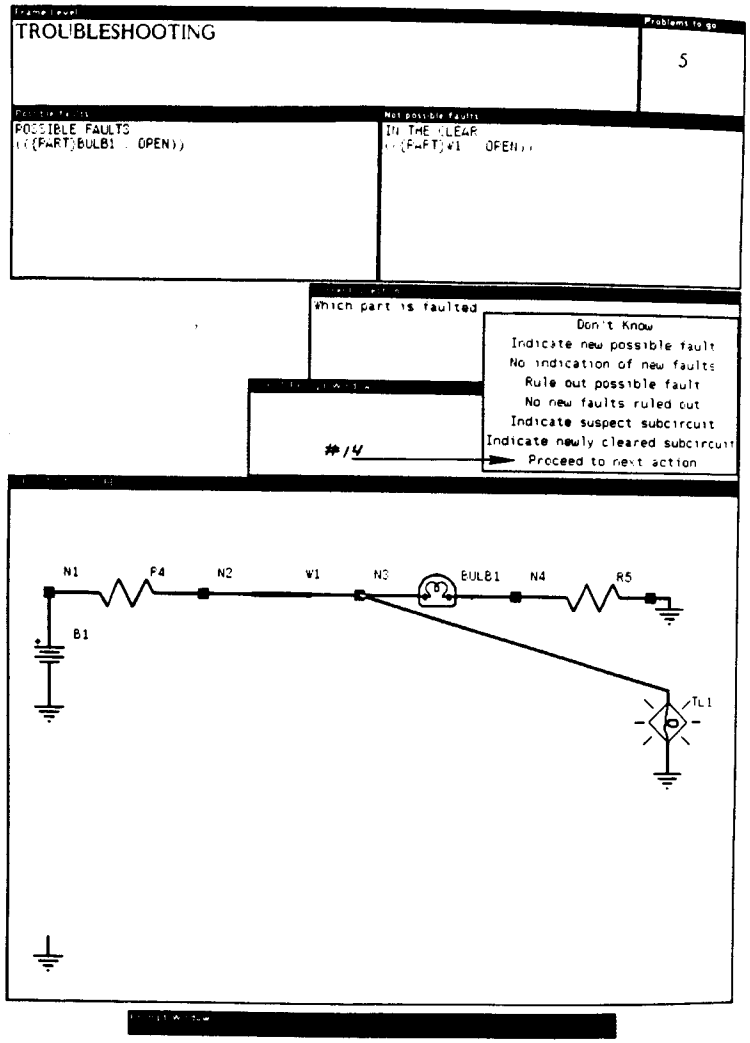

FIGURE 16.2h

Characterizing a student's problem-solving knowledge solely or primarily by observations of his overt actions has unnecessarily limited the power of student models. Soliciting the student's assistance in the attempt to determine the intentions and the reasoning, correct and faulty, that underlie his actions, is not at variance with the purpose or spirit of the intelligent instructional enterprise. Inviting students to rationalize their actions and to identify their hypotheses and goals

can enormously enrich a program's knowledge of the student. The use of a mouse, menu, and window-based interactions on current machines enables rapid and nonintrusive elicitation during problem solving in intelligent tutoring systems that acquire and use this new rich source of data.

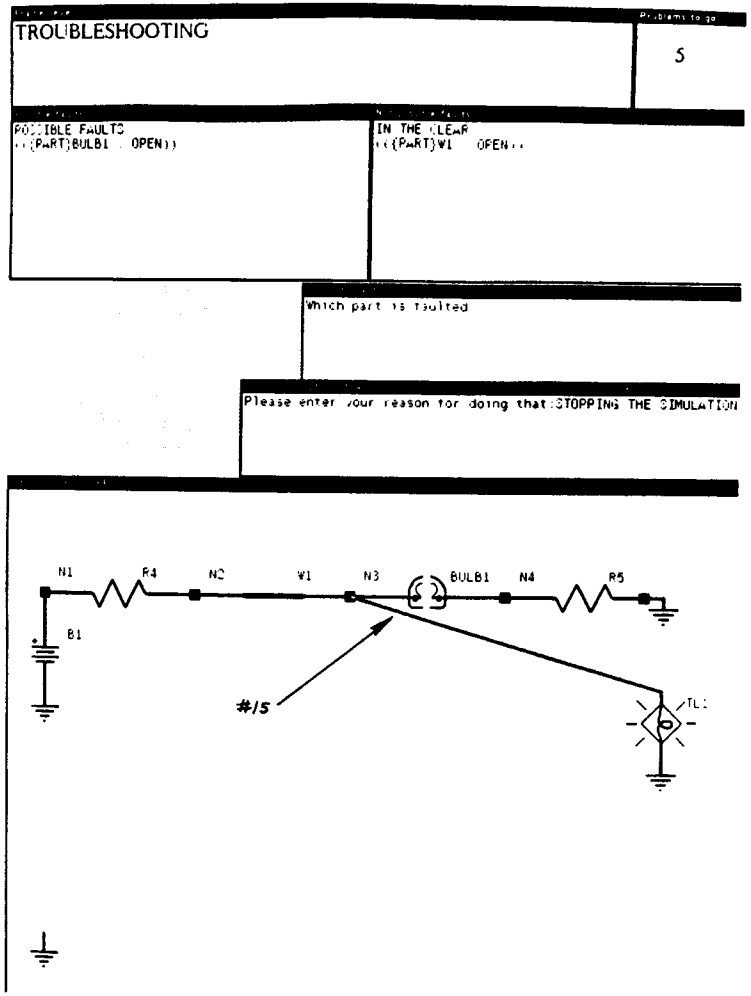

FIGURE 16.2i

ACKNOWLEDGMENT

This research was supported by the U. S. Office of Naval Research and the U. S. Army Research Institute under Contract N0014-82-C-0580.

REFERENCES

Brown, J. S., Burton, R. R., & Bell, A. G. (1974). *SOPHIE: A sophisticated instructional environment for teaching electronic troubleshooting (an example of AI in CAI) Final Report* (Tech. Rep. 2790). Bolt, Beranek & Newman.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring systems* (pp. 157-199). London: Academic Press.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (PP. 79-98). London: Academic Press.

Feurzeig, W. (1985, January). *Student modeling in QUEST*. Presentation at ONR program review on AI and education. Georgia Institute of Technology.

Nagel, L. W. (1975, May). SPICE2. *A computer program to simulate semiconductor circuits* (Tech. Rep. ERL Memo No. ERL-M520). Berkeley, CA: University of California.

Ritter, F. (1986). OREO. Orienting electrical circuits for qualitative reasoning. Unpublished manuscript.

White, B. Y., & Frederiksen, J. R. (1986). *Progressions of qualitative models as a foundation for intelligent learning environments* (Technical Report 6277). BBN Laboratories, Cambridge, MA.

White, B. Y., & Frederiksen, J. R. (in press). Qualitative models and intelligent learning environments. *AI and Education.*

White, B. Y., & Frederiksen, J. R. (1985). QUEST: Qualitative understanding of electrical system troubleshooting. *ACM Sigart Newsletter, 93*, 34-37.

17

The Next Wave of Problems in ITS: Confronting the "User Issues" of Interface Design and System Evaluation

Douglas Frye
David C. Littman
Elliot Soloway
Yale University

MOTIVATION AND GOALS

The standard architecture of an Intelligent Tutoring System (ITS) is depicted in Figure 17.1. By and large, the majority of research effort goes into developing the student modeling module, the expert module, and the tutorial module, just as almost all effort is put into building these modules, since they do form the core of the ITS. However, we have found that developing an effective interface and developing an effective evaluation may require as many resources as did the development of the three core modules. We are not alone in this experience. Yet, there are few guidelines for how to build an effective interface for educational software, and how to design an effective evaluation for intelligent tutoring systems. Even though human-computer interaction is a topic of growing interest, that field has not focused on the special properties of educational software. Similarly, even though educational evaluation is an established field, methodologies have not been developed for evaluating educational systems that attempt to teach students to "understand"—rather than simply to get the right answer.

In this chapter, we present two case studies, one dealing with our efforts to understand the special properties of interfaces for educational software, and one dealing with our efforts to evaluate an intelligent tutoring system. Our intent is more to spotlight the problems than to present a coherent, well-worked-out theoretical framework in which these issues can be viewed and resolved. The bottom line is this: As ITS become usable entities, and not merely