ARI Research Note 97-39

# Designing an Interactive Multimedia Environment for Learning and Aiding Troubleshooting

**Janet Kolodner and Mimi Recker**
Georgia Institute of Technology

**Research and Advanced Concepts Office**
**Michael Drillings, Chief**

**September 1997**

**United States Army**
**Research Institute for the Behavioral and Social Sciences**

**U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES**

**A Field Operating Agency Under the Jurisdiction of the Deputy Chief of Staff for Personnel**

**EDGAR M. JOHNSON**
**Director**

---

Research accomplished under contract
for the Department of the Army

Georgia Institute of Technology

Technical review by

Michael Drillings

**NOTICES**

# REPORT DOCUMENTATION PAGE

| | | |
|---|---|---|
| 1. REPORT DATE<br>1997, September | 2. REPORT TYPE<br>Final | 3. DATES COVERED (from. . . to)<br>June 1990-September 1994 |

| | |
|---|---|
| 4. TITLE AND SUBTITLE<br>Designing an Interactive Multimedia Environment for Learning and Aiding Troubleshooting | 5a. CONTRACT OR GRANT NUMBER<br>MDA903-90-K-0112 |
| | 5b. PROGRAM ELEMENT NUMBER<br>0601102A |
| 6. AUTHOR(S)<br>Janet Kolodner and Mimi Recker (Georgia Institute of Technology) | 5c. PROJECT NUMBER<br>B74F |
| | 5d. TASK NUMBER<br>2901 |
| | 5e. WORK UNIT NUMBER<br>C06 |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Georgia Institute of Technology<br>College of Computing<br>Atlanta, GA 30332-0280 | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>U.S. Army Research Institute for the Behavioral and Social Sciences<br>ATTN: PERI-BR<br>5001 Eisenhower Avenue<br>Alexandria, VA 22333-5600 | 10. MONITOR ACRONYM<br>ARI |
| | 11. MONITOR REPORT NUMBER<br>Research Note 97-39 |

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

COR: Michael Drillings

14. ABSTRACT *(Maximum 200 words)*:

The need for effective troubleshooting is rapidly becoming ubiquitous in our increasingly technological society. However, troubleshooting is a complex process both to learn and perform. This report examines the prospects for designing an interactive learning environment that helps users acquire and engage in effective troubleshooting. This work is informed by two important strands of related research. First, we draw upon research focused on the design and development of interactive learning environments. We are interested both in work focusing on theory-driven design on multimedia, and work focusing on how students learn in apprenticeship learning situations. The research summarized forms the basis for a prototype design of an interactive multimedia environment. The prototype is designed for the task domain of help-desk troubleshooting of computer systems problems for a large computer company.

15. SUBJECT TERMS

Troubleshooting    Vyasa    Problem solving

| SECURITY CLASSIFICATION OF | | | 19. LIMITATION OF ABSTRACT | 20. NUMBER OF PAGES | 21. RESPONSIBLE PERSON (Name and Telephone Number) |
|---|---|---|---|---|---|
| 16. REPORT<br>Unclassified | 17. ABSTRACT<br>Unclassified | 18. THIS PAGE<br>Unclassified | Unlimited | 62 | |

DESIGNING AN INTERACTIVE MULTIMEDIA ENVIRONMENT FOR LEARNING AND AIDING TROUBLESHOOTING

CONTENTS

LIST OF TABLES

**Preceding Page Blank**

Page

## LIST OF FIGURES

# DESIGNING AN INTERACTIVE MULTIMEDIA ENVIRONMENT FOR LEARNING AND AIDING TROUBLESHOOTING

## 1 Introduction

The need for effective troubleshooting is rapidly becoming ubiquitous in our increasingly technological society. People with a variety of educational backgrounds, involved in many different kinds of jobs, are using a growing number of technologically sophisticated tools. These tools frequently require troubleshooting.

However, effective and efficient troubleshooting of devices and systems is a complex process to perform. And, for several reasons, it is equally difficult to learn. First, troubleshooting requires detailed knowledge of the device or system in question, making it hard to separate domain knowledge from troubleshooting knowledge and strategies. Second, troubleshooting may occur in many different situational contexts, under different resource bounds, each requiring different reasoning strategies (Towne and Munro, 1988; Towne, 1993; Munro, 1993). For example, a diagnosis situation may require that the failed device be repaired immediately, or a longer turn-around time may be permitted; and, replacement parts may be plentiful and cheap, or they may be scarce and expensive; and, certain diagnostic tests may be difficult or time-consuming to perform.

Our research examines the prospects for designing interactive systems that help users learn effective troubleshooting skills and strategies. Furthermore, as the user gains expertise, we examine how the role of the interactive environment may change to become more of a personal assistant that aids and structures the troubleshooting process. This research thus provides an interesting case study of how cognitive science research can be used to inform the design of tools intended for authentic and complex work domains.

This research is informed by two important strands of related work. First, we draw upon work aimed at understanding people as they engage in complex troubleshooting tasks and, in addition, as they learn how to troubleshoot. In this arena, we draw upon results from both empirical and analytical studies (for example, AI and cognitive science models). We began by observing car mechanics as they diagnosed malfunctioning engines and, based upon that, developed a model of an ideal learner of troubleshooting skills (Redmond, 1992). This model tells us much about the kinds of situations that enable succsessful learning in active apprenticeship learning situations.

However, we still lacked a complete model of the troubleshooting problem solving. To this end, we performed studies of students diagnosing faults using a computer-based, dynamical simulation of an oil-fired marine steam power plant, called Turbinia, coupled with an intelligent tutoring system, called Vyasa (Vasandani and Govindaraj, 1993; Vasandani and Govindaraj, 1994). The domain of study is diagnosis of large systems, a complex activity that requires detailed knowledge of the device or system in question, along with a repertoire of troubleshooting strategies (Lancaster and Kolodner, 1988; Redmond, 1992). The Turbinia simulation environment provides a tractable means for analyzing the range of student diagnostic strategies. Our analysis of student troubleshooting activities is performed in the context of a theoretical framework for analyzing diagnostic tasks. In this framework, diagnosis is characterized as a process of generating hypotheses for explaining the observed faults, and testing these hypotheses by conducting experiments. Such reasoning has been described as *dual problem space search*, where processing alternates between search in the hypothesis problem space and search in the experiment problem space (Klahr and Dunbar, 1988).

The second important strand of research draws upon the design and development of interactive multimedia learning and aiding environments. We are interested in both work focusing on theory-driven design of multimedia, and work focusing on how students learn in apprenticeship learning situations (Collins et al., 1989; Lajoie and Lesgold, 1989). We review literature that highlights the important processes that comprise active learning. In addition, we present a theory for the design of interactive systems, based upon cognitive considerations of users and tasks.

This paper presents the empirical and analytical studies underlying the design of an interactive system for troubleshooting. Specifically, results from studies of troubleshooting provide the theoretical underpinnings for knowledge communication. Similarly, prior research on the design of interactive multimedia learning environments provide a theoretical basis for design. In tandem, these provide the motivation for the design of an interactive multimedia learning environment to help and aid people as they troubleshoot complex systems. The target domain for our system is customer help-desk employees. These employees, called help-desk specialists, work for a large computer company and receive telephone calls from customers experiencing problems with their computer systems. Through conversing with the customer, asking questions, and drawing upon information resources, they diagnose the problem and offer solutions.

This real-world work domain provides an ideal test-bed for our design. It is prototypical of troubleshooting tasks. Help-desk operators engage in complex troubleshooting, under difficult and time-constrained conditions. Customers typically want immediate solutions to their problems, yet they are not always able to correctly identify and articulate problems and symptoms. In addition, help-desk operators are limited in the kinds of diagnostic tests that they can perform, since they are primarily limited to conversations and question-asking. Finally, they have at their disposal a wide array of information resources. They perform diagnosis in an information-rich environment, drawing upon resources such as on-line databases, libraries of solved cases, manuals, and other specialists. They must make rapid decisions as to which resources will provide the best information in the shortest amount of time.

## 2 Troubleshooting Problem Solving

We begin by presenting the theoretical framework we have developed to characterize troubleshooting tasks. This section describes this theoretical framework for troubleshooting tasks and describe a computational model, which implements the framework.

## A Theoretical Framework for Diagnosis

We define diagnosis as identifying the component that is causing the faulty condition. We propose that this process involves *identifying and clarifying* the initial symptoms, *generating hypotheses* to explain the symptoms, *running diagnostic tests* to confirm or disconfirm hypotheses, and *evaluating* test results. Within cognitive science, the alternation between hypothesis generation and testing has been characterized as *dual problem space search* (Klahr and Dunbar, 1988). In such models, search alternates between (1) the *hypothesis* space, which contains all possible hypotheses for the task, and (2) the *experiment* space, which contains all possible experiments for the task. This framework has been proposed as a model applicable to any problem solving situation in which hypotheses are proposed and data are collected (Klahr and Dunbar, 1988)[1].

Search in the hypothesis space entails proposing components whose failure best explains the observed symptoms. The search is guided by both prior knowledge and results from experiments. Search in the experiment space entails conducting diagnostic tests whose results, in turn, may confirm or disconfirm particular hypotheses under consideration. Search in the experiment space may be guided by currently active hypotheses, or may serve to gather information for formulating hypotheses.

Klahr and Dunbar (1988) developed this framework to characterize problem solvers' actions as they engaged in process of scientific discovery. The framework enabled them to characterize problem solvers in terms of how they chose to search the two problem spaces. *Theorists* describe problem solvers who first attempt to generate hypotheses that are then confirmed or disconfirmed through experiments. This top-down approach begins with search in the hypothesis space, and the search in the experiment space is constrained by the hypotheses under consideration. *Experimenters* describe problem solvers who search the experiment space without explicit hypotheses. In this bottom-up strategy, hypotheses are induced from experimental results.

Thus, strategies are crucial for determining where and how the two problem spaces are searched. For example, when searching the hypothesis space, are several hypotheses considered, or only one? When searching the experiment space, are tests conducted to 1) confirm the leading hypothesis, 2) disconfirm hypotheses, or 3) maximize information gain? In our empirical studies, we were interested in determining students' strategies for coordinating search in the two problem spaces, whether students could be characterized as theorists or experimenters, and the effects of resource bounds on students' search strategies. These studies provide a baseline that tells us how the novices we are addressing see the problem and which strategic pieces of troubleshooting the system needs to help them learn.

[1]It is also interesting to note that some theories of scientific discovery are inspired by AI-based models of troubleshooting (Darden, 1990)

## TS: A Computational Model

In order to carefully separate and analyze the effects of knowledge, strategies, and control during troubleshooting, we created TS (TroubleShoot), a computational model of the knowledge and strategies involved in troubleshooting an automobile engine.

TS was implemented using the Soar cognitive architecture (Laird et al., 1987). Soar is a general problem solving architecture, which includes an experience-based learning mechanism, called *chunking*. Information processing in Soar involves search in problem spaces through the application of operators in order to achieve a particular goal, with knowledge influencing both the structure and efficiency of the search process (Newell, 1990).

The Soar architecture has two kinds of memories. The first memory, a recognition or content-addressable memory, stores long-term knowledge in production rules (or situation-action pairs). The second memory is called working memory, and contains a set of objects that represents the current problem solving situation. Changes to working memory trigger parallel access to recognition memory, where all productions whose conditions match to elements currently stored in working memory are fired.

### General Modelling Framework

Within TS, successful troubleshooting is defined as identifying the component that is causing the faulty condition. As suggested by the theoretical framework, the process involves identifying and clarifying the complaint, generating hypotheses to explain the symptoms, and running tests to confirm or disconfirm hypotheses. This process may be iterated several times until the faulty component is located.

### Implementation

The above theoretical framework is implemented in the TS Soar model by dividing the troubleshooting problem solving into multiple problem spaces. In addition, the TS model consults an external simulation of the malfunctioning engine in order to collect the initial complaint and results from diagnostic tests.

The problem solving begins by requesting the complaint from the external simulation. This is also an opportunity for problem re-description. For example, a generic complaint may be re-interpret in terms of the more specific underlying causal model.

After collecting the complaint, the system begins problem solving in one of three problem spaces: 1) the hypothesis generation space, 2) the hypothesis testing space, and 3) the hypothesis evaluation space. The order in which problem spaces are visited and re-visited depends on the particular students' problem solving strategy.

**The hypothesis generation problem space.**

The role of this problem space is to formulate a hypothesis about which system component is faulty. Hypotheses can be generated in many different ways. The easiest method (the least computationally expensive) for the generation of hypotheses occurs via the application of relevant symptom-fault pairs. These are acquired through experience. Hypotheses can also be generated by specializing a previous, more general hypothesis. They can be generated through causal reasoning, where previous hypotheses and test results are considered. Finally, they can be generated by analogies to previous problem solving episodes.

In addition, hypothesis generation can be either serial or parallel. That is, several hypotheses can be considered at once, and one selected as a the leading contender, or a single hypothesis can be generated at a time.

After the set of hypotheses has been generated, one is selected. Several heuristics are available for doings this, e.g., choosing the easiest to test, choosing the most probable, or choosing the most familiar.

**The hypothesis testing problem space.**

Two main processes occur here: Proposing a test and carrying it out. Strategies help select the kinds of tests selected. Heuristics help to identify tests that are cheap to conduct, yet are maximally informative. For example, visual inspections of parts are much quicker to perform than using expensive test equipment. Tests can serve to confirm or disconfirm active hypotheses. Once a test is chosen, a prediction of its result is made.

Next, the test is conducted and the external simulation returns the test result. The test, the prediction, and the test value are all recorded.

**The hypothesis evaluation problem space.**

The role of the evaluation problem space is the evaluate the currently active hypothesis in light of the current data (hypotheses considered, tests conducted, and tests results). The test result is first compared to the prediction. Depending on the comparison, TS will do one of several things: (1) it might halt with its final diagnosis, (2) it might reject the current hypothesis and return to the hypothesis space, (3) it might propose that the current hypothesis be refined and return to the hypothesis space, and (4) it might decide that further tests results are inconclusive and return to the test problem space.

## Protocol Simulations

In order to determine coverage of the model, we used TS to interpret the protocols of two students and an instructor in an auto mechanics vocational education class (Lancaster and Kolodner, 1988). These protocols, collected during the previous contract period, were analyzed informally and formed the primary motivation for the model of learning to troubleshoot (Redmond, 1992). These same protocols were revisited and modeled more formally in a model of troubleshooting problem solving.

In the model, we were interested in determining which heuristics, knowledge, and strategies led to their decisions. Input to TS was the troubleshooting behavior of participants. TS constructs a chain of inferences that provides a plausible explanation of their behavior.

For the current model, protocols were drawn from a troubleshooting session in which a piece of tape had been wrapped around the outgoing terminal of the fuel pump fuse. The car's symptom was that it would turn over but would not start. This problem was successfully solved by all participants.

We added knowledge about automobile fuel systems to TS to allow it to simulate the behavior of participants. In general, the primary differences between the participants were the number of hypotheses generated, whether or not they were generated in parallel, heuristics for choosing the best hypothesis (e.g. a bias toward the "easiest" hypothesis to test), and types of tests selected (confirming vs. disconfirming).

### First Student: Novice

The first student was considered intermediate in that he was in his second quarter at the school. Upon beginning the problem, he suspects a fuel pump problem almost immediately. Then, in sequence, he considers a number of subcomponents of the fuel pump and tests each one in turn. His search is hampered by a difficulty in locating the fuel pump relay. In sum, this protocol is characterized by a sequential generation of hypotheses, each a refinement of the previous. Tests are also proposed sequentially, in an attempt to confirm the active hypothesis. The only real difficulty the student encounters occurs due to lack of topographic knowledge.

Early in his reasoning, the student uses a symptom-fault pair to go from non-starting car to hypothesizing the fuel pump as the faulty component. Clearly this is an idiosyncratic pair, but, due to the fault distribution probability of most cars, it has a fairly high degree of reliability.

### Second Student: Advanced

The second student was considered advanced in that he had more background that the previous student. He locates the failure easily once he decides to check the fuses. But first, he incorrectly applies a procedure by using a wrong piece of equipment during a test, which slows his progress. All of his tests act to confirm the active hypothesis.

The protocol begins a parallel generation of hypotheses. The generation appears to be driven by a symptom-fault pair:

> What are the 3 things we need for a car to run? Air, fuel, and heat. The first thing I'm gonna check and make sure we got fuel.

Note that the trigger, "car to run", is fairly vague at this point. We suspect that such triggers, with experience, become more attuned to relevant car features. It is unclear what prompts the ordering of these hypothesis such that one is selected over the others.

This student uses causal reasoning to make some of his decisions to interpret a test result, and to generate or refine a hypothesis:

So, what we're gonna think about now is why that fuel pump is not running when it's got 12 volts back there to it.

Okay, the ECM controls the injector so what we have – the fuel pump's running – we tested it we ran straight to that bypass wire went back there and heard it. And we know we got fuel up to the injector and what the injector does is – the ECM – the computer puts 13 volts to the red wire to the injector and then the other side goes to ground so when the key is on you have ignition going to the red wire to the injector so you take the test light, hook it to the ground and the light should come on and it's not.

This student also displays an awareness of some potential pitfalls of tests. As a backup, he proposes more reliable (and also more "expensive") tests. Thus, in this example, tests are ordered according to how difficult they are to conduct, with an accompanying awareness of the reliability of their results.

So we want to do now is check the fuse to the ECM and check it. It looks okay, but that doesn't mean it's right – 'cause in the back of it a wire could be loose. What I need to do though is to check it with the light.

## Instructor: Expert troubleshooter

The instructor (an expert) generates many hypotheses in parallel and then employs a hypothesis-elimination strategy. He also appears to select tests on the basis of how easy they are to conduct. Finally, he, on occasion, selects a test that acts to *disconfirm* a test. We also observe frequent episodes of causal reasoning in which hypotheses are proposed or refined (but not rejected).

In this first example, we see a refined symptom-fault pair pointing to three hypotheses. The instructor, based on a non-starting car, immediately identifies *non-combustion* (a higher-level perceptual feature) as the symptom. Notice that he does not say "the car doesn't start." On the basis of the perceived symptom, three possible faults are hypothesized. Then, one is selected on the basis of ease of investigation.

First thing I start thinking about is what? The three things that you have to have for combustion – fuel, heat, and air. Ok? Now – what you gotta think about there is what is the easiest thing to check? The fuel.

This excerpt shows an example of a test that is conducted to *disconfirm* a hypothesis:

Now as I power the ECM down for 10 seconds I'm gonna turn the key to the ON position and listen...And I hear no pump run...But it should run for 2 seconds, so now I – now wait a minute – you the hear fuel pump relay energize, you'll hear the pump run. Now, I know that the pump is not running. I know that I don't have any fuel.

## Summary

The theoretical framework we developed helps address and characterize the inherent complexity present in troubleshooting tasks. In addition, the model and simulation of protocols demonstrate the usefulness of the framework for understanding the troubleshooting process. They serve to highlight differences between the troubleshooting processes of the students and the instructor. First, the students typically generate hypotheses in a serial fashion and generally only one hypothesis is considered at a time. The instructor, on several occasions, generates hypotheses in parallel. Second, differences are observed in how selections are made. For the instructor, the hypothesis that is deemed the *easiest* to investigate is typically the one that becomes active. The students appear to select on the basis of familiarity.

Third, differences exist in how testing strategies are selected. Often the instructor selects a test whose outcome will disconfirm active hypotheses. The students tend to select tests that will confirm their hypothesis. The technique of selecting a measurement that produces maximal elimination of hypotheses is frequently used in AI models of troubleshooting (deKleer, 1990). While it is certainly an efficient technique, it was never observed in the students' protocols. We note that there is also probably a relationship between the method used to generate hypotheses (i.e. in parallel) and whether a test was selected that served to disconfirm multiple hypotheses.

Finally, and quite obviously, knowledge differences affect the process. The students possess a smaller and less refined symptom-fault set and were less able to effectively use causal reasoning in order to focus their search, compared to the expert.

## Empirical Studies of Troubleshooting

The previous section presented a theoretical framework for troubleshooting, a computational model, and a small protocol study of people troubleshooting an automobile engine. The model and the protocol analyses were intended as initial explorations of troubleshooting problem solving. In particular, these preliminary studies revealed the importance of generation hypotheses, and testing those hypotheses. Strategies were also suggested as important for coordinating these processes in these two problem spaces.

In order to further examine the role of hypothesis generating and testing and how strategies coordinate these processes , we conducted empirical studies of students' troubleshooting a complex, dynamic system. In our studies, students diagnosed faults in a computer-based simulation of an oil-fired marine steam power plant, called Turbinia, coupled with an intelligent tutoring system, called Vyasa (Vasandani and Govindaraj, 1993; Vasandani and Govindaraj, 1994). The domain of study is diagnosis of large systems, a complex activity that requires detailed knowledge of the system, along with a repertoire of troubleshooting strategies (Lancaster and Kolodner, 1988; Redmond, 1992).

An important benefit of conducting empirical studies in the context of computer-based simulation systems is that they offer invaluable opportunities for studying student problem solving activities and strategies. Specifically, the simulation context provides a more constrained environment for analyzing learning and performance in complex, problem solving domains. In addition, the simulation context allows tractable means for recording the problem solving of a large number of people, thus establishing greater reliability. Results from such studies have several important implications. Theoretically, they provide insights into the nature of cognition and learning in the context of interactive educational technologies. Practically, they provide guidance on the design of such systems.

In the first study reported, we analyzed students' diagnostic goals and strategies in terms of the theoretical framework. We were interested in determining students' strategies for conducting and coordinating search in the two problems space. In addition, we wished to characterize these strategies with respect to students' overall diagnostic performance.

In addition, diagnosis can occur in many different situational contexts, under differing resource bounds (Towne and Munro, 1988). For example, a diagnosis situation may require that the failed device be repaired immediately, or a longer turn-around time may be permitted. Moreover, replacement parts may be plentiful and cheap, or they may be scarce and expensive. Finally, certain diagnostic tests may be difficult or time-consuming to perform.

Again, the Turbinia simulation environment provides means for analyzing diagnostic strategies under different situational contexts. These analyses can be conveniently performed using the simulation, while avoiding some of the costly and inherent difficulties present in performing field studies. In particular, the second study we report investigates the effects of resources bounds on students' diagnostic strategies. Two resource bounds were investigated: time bounds and cost bounds. The effects of time constraints on diagnosis performance were investigated by manipulating the time available for diagnosing faults. The effects of cost were investigated by limiting the number of diagnostic tests that students were permitted to conduct.

The results from these two studies show how simulation environments offer methodological advantages for studies of student learning and performance. Moreover, these results provide important feedback and implications for the design of software simulation and learning environments. For example, the results raise issues in the kinds of *fidelity* that are designed in a computer simulation in order to model the corresponding external system (Collins, in press). In one common approach, the system designer attempts to maintain an *epistemic* fidelity with the external system (Wenger, 1987). In this view, the representations used in the simulation capture the structural organization of knowledge assumed to be held by the expert. An implicit instructional assumption is that the learner internalizes the expert's mental model through interacting with the simulation. In Turbinia, the organization of systems, subsystems, and components were modeled on an expert's understanding of the marine power plant (Govindaraj and Su, 1988). In another approach, the simulation maintains a *dynamic* or temporal fidelity with the external system. That is, as the external system changes through time, the corresponding simulation reflects these changes. Turbinia also attempts to maintain dynamic fidelity, as the effects of faults propagate through the simulation with time.

As we will argue, the results from our studies show that students develop strategies that are tuned to particular constraints present in the simulation environment. In short, students' strategies develop from exposure to a range of situation contexts exhibited by the system. These results, then, suggest a third, important kind of fidelity to be considered during design – one that we have termed *fidelity of interaction.* In this view, the design of simulations of dynamical domains should attempt to preserve the costs and resource bounds of target situations. This will enable students to develop strategies that are faithful to the activity demands of real-world situations.

## Turbinia-Vyasa

In our empirical studies, we used a computer-based simulation, called Turbinia-Vyasa. This simulation is an instructional system that trains operators in diagnostic problem solving in the domain of marine power plants. It is comprised of a steam power plant simulator and an intelligent tutoring system. Turbinia-Vyasa is implemented in Macintosh Common Lisp with Common Lisp Object System and runs on Apple Macintosh II computers. The simulator, Turbinia, is based on a hierarchical representation of subsystems, components, and primitives together with necessary physical and logical linkages among them. Turbinia can simulate a large number of failures in a marine power plant. Approximately 100 components have been modeled to achieve fairly high degrees of structural and dynamic fidelity even though the physical fidelity of the simulator is rather low. Vyasa is the computer-based tutor that teaches the troubleshooting task using Turbinia. The simulator, an interactive, direct manipulation interface, and the tutor (with its expert, student, and instructional modules) comprise the instructional system.

A student interacts with Turbinia-Vyasa by choosing a schematic icon, a component, a gauge, or icons representing functions of the tutor. The boiler schematic, along with various icons, is shown in Figure 1.

## Study 1: Diagnostic Strategies

As discussed, the first study provides an analysis of students' diagnostic strategies in the context of Turbinia. In particular, we analyzed students' diagnostic goals and strategies in terms of our theoretical framework. We were interested in determining students' strategies for conducting and coordinating search in the two problems space. In addition, we wished to characterize these strategies with respect to students' overall diagnostic performance.

In this study, students learned to troubleshoot using one of three training methods. The training phase was followed by a test phase, in which students diagnosed a series of faults using the simulator only. Details and results of students' performance during the

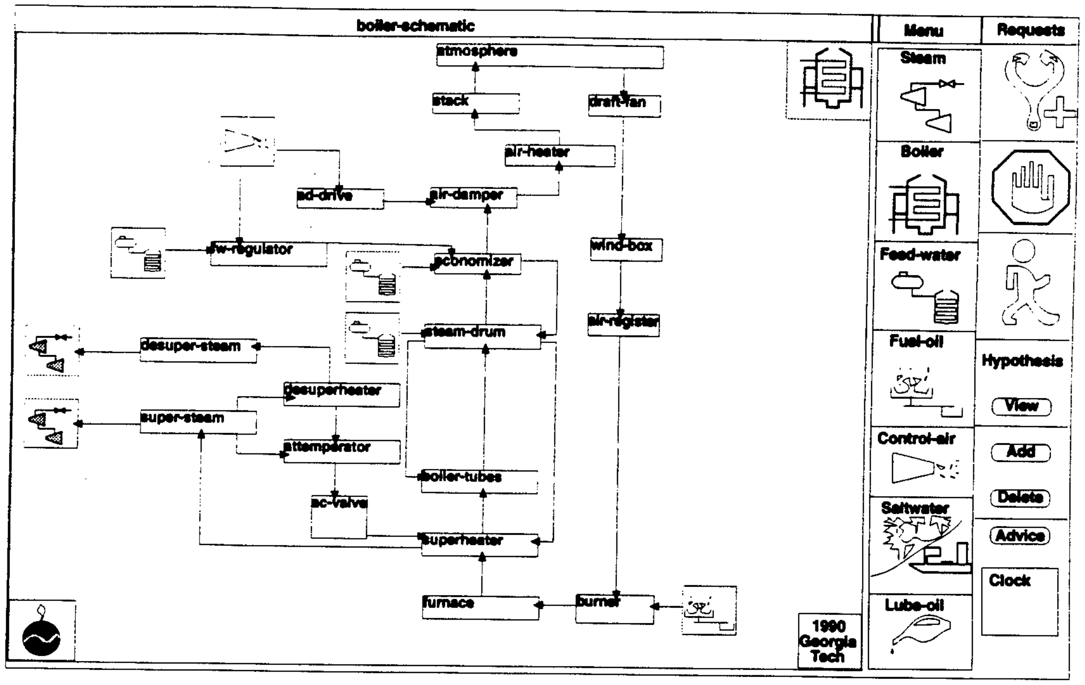


Figure 1: The boiler schematic in Turbinia.

training phase are reported elsewhere.(Vasandani and Govindaraj, 1994). In the analysis presented here, we focus on students' diagnostic strategies after the training phase and while diagnosing faults using only the simulator. More details on the results of the study can be found elsewhere (Recker et al., 1994).

## Method

Thirty undergraduate Georgia Tech Naval ROTC cadets served as students in the study. The study consisted of a training phase (10 sessions) and a testing phase (two sessions). All sessions lasted approximately 1 hour. During the training phase students were randomly assigned to one of three training conditions : (a) training using the simulator alone (Turbinia), (b) training with the aid of a passive tutor (passive Vyasa), and (c) training with the aid of an active tutor (active Vyasa). The training phase spanned 10 sessions, with students typically attending one session per consecutive day. Students were never allowed to attend more than one session per day.

During each subsequent training sessions, students solved three faults. Overall, students diagnosed a total of 28 faults during the training phase. The fault ordering was randomly determined, and identical for all students.

The training sessions were followed by a test phase, consisting of two sessions, which was identical for all students. Five faults were new, while five faults had been presented during

| Mouse Category | Percent | Overall Mean |
|---|---|---|
| Gauges | 39 | 25.19 |
| Components | 28 | 18.26 |
| Schematics | 9 | 5.78 |
| Symptoms | 2 | 1.62 |
| Diagnoses | 6 | 4.00 |

Table 1: Percentage and mean number of mouse actions per category.

the prior training sessions.

**Results**

The Turbinia system kept a permanent record of each students' mouse actions. Each mouse action served one of the following functions: 1) a request to view the initial fault symptoms 2) a request to view a schematic, 3) a request to view a component, 4) a request to view a gauge, and 5) a request to make a diagnosis.

The first type of mouse action was assumed to be problem formulation or elaboration. The second and third types of mouse action were assumed to involve hypothesis formulation. Mouse actions in the fourth category were assumed to involve hypothesis testing, while actions in the the fifth category were considered to be hypothesis evaluation.

**Mouse Actions.** Overall, the mean number of mouse actions per fault condition varied widely, and can be seen as reflecting fault difficulty. The high variability also suggested that there was no learning effect during the test phase. For this reason, in the analyses reported below, faults were sorted by difficulty. Fault difficulty was determined by sorting on the mean number of mouse actions per fault presented.

Table 1 shows the overall percentage and mean number of mouse actions in each category for all faults. Viewing gauges was the most common activity, accounting for 39% of students' mouse actions. Viewing components accounted for 28% of students' mouse actions. Calls for viewing a new schematic accounted for 9% of the mouse actions, while evaluating diagnoses accounted for 6%. The most infrequent action was viewing the initial symptoms (2%). Thus, over a third of the students' actions involved conducting experiments. These results also suggested that, overall, students did not appear to engage in much problem formulation or elaboration.

The large number of tests suggested that students were following a strategy of attempting to confirm their hypothesis. The ubiquity of the positive-test strategy is a robust finding in the scientific discovery literature. While the positive-test strategy is generally acknowledged to be a less-efficient strategy than a negative-test strategy (Freedman, 1992), its optimality is, in reality, a function on the distribution of positive and negative instances (Klahr and Dunbar, 1988). For example, if the probability of confirming a hypothesis is high, then a positive test result does not add much new information. However, in the Turbinia simulation,

| Mouse Category | Mean Quick | Mean Slow | Main Effect $F(1,28)$ | p-value | Group X Trials p-value |
|---|---|---|---|---|---|
| Gauges | 16.76 | 33.42 | 6.85 | .0001 | .0001 |
| Components | 12.24 | 24.28 | 27.22 | .0001 | .02 |
| Schematics | 4.78 | 6.77 | 9.69 | .005 | n.s. |
| Symptoms | 1.57 | 1.68 | - | n.s. | n.s. |
| Diagnoses | 3.57 | 4.42 | - | n.s. | n.s. |

Table 2: ANOVA results.

the majority of the gauges had normal levels. Therefore, students had a low probability of encountering abnormal gauges, which would serve to confirm their current hypothesis. As such, the use of a positive-test strategy is a quite reasonable heuristic.

**Diagnostic Efficiency.** Students were divided into two groups, *Quick* vs. *Slow*, which was intended to capture students' troubleshooting efficiency. The split was based on a post-hoc median split of the mean number of mouse actions performed when diagnosing novel faults during the test phase. In performing this split, we were interested in characterizing differences between efficient (Quick) and less-efficient (Slow) troubleshooters.

Analyses of variance were conducted with the number of mouse actions in each category per fault (sorted by difficulty) as the repeated measure, and group as the between-students factor. There was a strong main effect of trials (all $p$'s $< .0001$) in *all* mouse action categories. This result again reflected fault difficulty. On harder faults, students were simply making many more mouse actions in all categories.

In terms of the number of gauges viewed by students in the two groups (Quick or Slow), there was a strong main effect of group and an interaction of group with trials (see Table 2). Moreover, there was a significant linear trend showing increased viewing of gauges with increased fault difficulty, $F(1, 28) = 24.66$, $p = .0001$. Thus, as faults increased in difficulty, the less-efficient group viewed significantly more gauges

Similar significant differences were found in the number of components viewed by students in the two groups. There was a main effect of group and an interaction of group (Quick vs. Slow) with trials (see Table 2). Not only did the less-efficient students make significantly more component checks, their number increased significantly as faults became more difficult. As with gauge checking, there was a significant linear trend showing increased viewing of component with increased fault difficulty, $F(1, 28) = 13.07$, $p = .001$.

**Summary.** By definition, the less-efficient students made more mouse clicking actions. However, a significant difference between the less-efficient and efficient troubleshooters was found only in the number of gauges and components viewed. These results suggest that the primary difference between efficient and less-efficient troubleshooters was in the number of diagnostic tests. The less-efficient students conducted significantly more tests, and this difference became more pronounced as faults increased in difficulty. These students could thus be characterized as *experimenters* in that they were attempting to induce the failed component by searching for abnormal gauges. The more-efficient students conducted significantly fewer experiments, suggesting a better search of the hypothesis space.

## Study 2: Effects of Resource Bounds

In the real world, many kinds of external constraints operate during diagnosis. The second study was addressed at investigating the effects of resource bounds during diagnosis. The central questions in this study were the role of resource bounds and their resulting impacts on students' diagnostic strategies.

We investigated the effects of two kinds of resource bounds on students' diagnostic strategies: time and cost. Time limits within Turbinia were implemented by restricting the time available to diagnose faults. Cost limits within Turbinia were implemented by adding a "Cost" window to the interface, next to the window that kept track of elapsed time. Upon startup, this window displayed a number that was decremented by one each time a diagnostic test was conducted. In the case of Turbinia, this corresponded to consulting a gauge attached to a component.

### Method

Twenty-four Georgia Tech Naval ROTC cadets served as students in the study[2]. The study consisted of three sessions, each lasting approximately two hours. The first session was a training phase, in which students diagnosed 8 faults using Turbinia and the intelligent tutor, Vyasa.

The next two sessions were test sessions. In these sessions, students were randomly assigned to one of four conditions: 1) time and cost bounds, 2) cost bound only, 3) time bound only, 4) no bounds. In each test session, students diagnosed 8 faults using only Turbinia. The fault ordering was identical for all students in all sessions, and all faults were novel.

**Design.** There were two main independent variables of interest: diagnosis time (bounded, unbounded) and diagnosis cost (bounded, unbounded), resulting in a 2 X 2 between-subjects design.

Bounds on time and costs for each fault were determined by using baseline data from Study 1. From these data, we calculated the mean solution time and mean number of gauges viewed per fault condition. For diagnosis time, students in the *unbounded* time condition were given 10 minutes to diagnose each fault. In the *bounded* time condition, for each fault condition, students were given the mean time to solution from Study 1, rounded up to the nearest minute. For diagnosis costs, students in the *unbounded* cost condition were given 100 cost units for diagnosing each fault (an ample amount). In the *bounded* time condition, for each fault condition, the cost window was initialized with the mean number of gauge

[2]The data from one student were discarded due to an experimenter error.

| | | | COST | |
|---|---|---|---|---|
| | | | yes | no |
| | N | yes | 6 | 6 |
| | | no | 6 | 5 |
| | Proportion Diagnosed | yes | .75 | .56 |
| | | no | .68 | .90 |
| | Time (mins) | yes | 4.03 | 4.76 |
| | | no | 6.14 | 4.13 |
| | Mouse Actions | yes | 39.37 | 44.79 |
| | | no | 49.95 | 59.35 |
| TIME | Gauge Actions | yes | 9.95 | 14.56 |
| | | no | 14.00 | 17.85 |
| | Component Actions | yes | 11.33 | 13.06 |
| | | no | 16.18 | 18.05 |
| | Schematic Actions | yes | 4.20 | 3.62 |
| | | no | 5.72 | 4.40 |
| | Symptom Actions | yes | 1.66 | 1.83 |
| | | no | 2.18 | 1.70 |
| | Diagnosis Actions | yes | 4.31 | 3.79 |
| | | no | 3.64 | 5.65 |

Table 3: Proportion of faults diagnosed, time, and mean number of mouse actions per condition (yes=bounded; no=unbounded).

consultations from Study 1, rounded up to the nearest integer.

**Results**

In our analysis, we considered two performance measures: the number of faults diagnosed and the time to solution per fault successfully diagnosed. We also considered several process measures: the total number of mouse actions and the number of mouse actions in the five mouse action categories. We conducted ANOVA on the performance and process measures, with COST and TIME as the independent variables. Means for each condition are shown in Table 3.

Our results indicate that students with no resource bounds exhibited the most successful diagnostic performance. Bounds on *time* allowed for diagnosing faults led to a reduction in the overall number of actions performed and components viewed, without appearing to affect performance. Bounds on the number of diagnostic tests (*COST*) reduced search in the experiment space, which appeared to negatively affect diagnostic performance. As suggested by results from the first study, testing was greatly relied upon by students (perhaps due to the structure of the Turbinia environment), and removing this capability adversely affected performance. Students with no time bounds but with cost bounds appeared to adopt a conservative diagnostic strategy, which in the end did not prove beneficial. Taken together, these suggest results that students' diagnostic strategies were sensitive to constraints in the external task environment.

## Discussion

In this paper, we have presented analyses of students' diagnostic strategies in the context of an interactive multimedia simulation environment, called Turbinia. We have argued that the use of such simulation systems provide opportunities for analyzing student learning and performance in complex domains.

Results from the first study show that the students performed a large number of diagnostic tests, suggesting that they were primarily engaged in search of the experiment problem space. Unlike other studies where troubleshooters used strategies such as "half-split" or symptomatic search (White and Frederiksen, 1990), the students in the present study did not seem to engage in much symptom evaluation, and did not seem to rely on symptom-fault pairs. This is perhaps due to their lack of experience in the domain, the complexity of the Turbinia environment, and the great ease of conducting tests in Turbinia.

Students also appeared to rely on a strategy of confirming leading hypotheses, rather than selecting tests that served to *disconfirm* a maximal number of possible hypotheses. The technique of selecting a measurement that produces maximal elimination of hypotheses is frequently used in AI models of troubleshooting (de Kleer, 1990). As discussed, while the negative-test strategy is generally acknowledged to be an efficient technique, its applicability is, in reality, a function on the distribution of positive and negative instances. The distribution of abnormal gauges in Turbinia is such that the positive-test strategy is a reasonable heuristic. In addition, the scarcity of the negative-test strategy may also be due to students' relative inexperience in the domain.

The results of our analyses of students' diagnostic efficiency are consistent with those found in scientific discovery (Klahr and Dunbar, 1988). We found that the primary difference between the diagnostic strategies of efficient and less-efficient students was in the number experiments conducted. The less-efficient students appeared to adopt a highly data-driven strategy of searching for abnormal gauges. This strategy could be viewed as a kind of abductive process, in which finding abnormal gauge readings helped to induce the faulty component. The fact that the more efficient students performed significantly fewer diagnostic tests suggested that they engaged in a better search of the hypothesis problem space. It is possible that differences in prior knowledge about steam engines accounted for students' ability to formulate and effectively search the hypothesis space.

The results from second study suggest that learning in the context of a simulation environments impact the kinds of strategies adopted by students. In particular, we examined the effects of imposing resource bounds on students' diagnostic strategies. Students with no resource bounds exhibited the most successful diagnostic performance. Bounds on *time* allowed for diagnosing faults led to a reduction in the overall number of actions performed and components viewed, without appearing to affect performance. Bounds on the number of diagnostic tests (*COST*) reduced search in the experiment space, which appeared to negatively affect diagnostic performance. As suggested by this first study, testing was greatly relied upon by students (likely due to the structure of the Turbinia environment), and removing this capability adversely affected performance. Students with no time bounds but with cost bounds appeared to adopt a conservative diagnostic strategy, which in the end did not prove beneficial.

In our study, the cost factor (bounds on the number of experiments) appeared to have the largest effects on diagnostic efficiency and accuracy. We are currently working on augmenting the DPSS model to account for these results. This requires analyzing the role of experimentation within both DPSS and the task environment. In DPSS, experiments are conducted to generate or test hypotheses, or to gather data (Klahr and Dunbar, 1988). In Turbinia, experiments consisted of reading gauges. Gauges had a fairly high density and, in the experimental conditions with no cost bounds, their access was cheap and easy. Our modeling approach involves adding an additional component to the search framework. Specifically, our hypothesis is that when considering a diagnostic test, the reasoner first estimates the cost of a particular diagnostic test against the expected information gain. As the reasoner gains expertise, these estimates better reflect the cost structure of the task environment. We anticipate that a model based on this approach will better capture the decisions and complexity faced by troubleshooters in real-world, resource-bounded situations.

**Designing for Fidelity.** Software simulation systems are built upon some underlying model of a real-world phenomena. The kind of model embodied in the system is defined as the *fidelity* of that simulation (Collins, in press). In one common approach, the system designer maintains *epistemic* fidelity with the external system (Wenger, 1987). That is, the representations used in the simulation represent the knowledge structures of the mental model held by the expert. Within Turbinia, the organization of systems, subsystems, and components were modeled on an expert's understanding of the marine power plant (Govindaraj and Su, 1988). The learner is assumed to internalize that mental model through interacting with the system.

Results from our studies offer feedback on the design of such systems. The results show that students' diagnostic strategies were clearly sensitive to features and constraints present in the diagnostic situation. For example, results from the first study indicated that students relied highly on gauge testing. This is perhaps due to the great ease of consulting gauges in Turbinia. In contrast, tests in real-world troubleshooting situations may be expensive or time-consuming. Moreover, there may be a high penalty for mis-diagnosis.

Therefore, we argue that there exists an additional aspect of fidelity in the design of dynamical simulations. This fidelity, which we term the *fidelity of interaction*, captures the costs and resource bounds of the corresponding situation. For example, in the medical domain, the cost of a mis-diagnosis can be very expensive. Similarly, conducting many expensive and time-consuming tests is not always the best course of action. These kinds of factors should be captured in the simulation. Preserving this fidelity will enable students to develop strategies that are faithful to the activity demands of real-world situations.

## 3 Learning to Troubleshoot

CELIA (Redmond, 1992) models the memory and reasoning capabilities of a novice trouble shooter. The novice has very little in the way of a domain model to start with, but with experience, it both acquires cases and updates its domain model. What is particularly interesting about CELIA is that it inserts case-based reasoning and the use of cases in reasoning into a broader model of memory that reasons using several different methods and several different kinds of knowledge.

CELIA acts as an apprentice mechanic. It solves problems by itself, and more importantly, it learns by watching and listening to a teacher explain his reasoning about particular cases, and integrating those experiences and what is learned from them with what it already knows. CELIA uses case-based reasoning, but it is not a case-based reasoner per se. That is, it uses several different kinds of knowledge to reason, among them cases. While it learns new cases and new indexes, it also learns several other things and uses a variety of different learning methods.

CELIA takes as input several sequences of teacher actions and explanations collected while observing teachers in an area vocational technical school. CELIA's memory, which is meant to hold the same sorts of knowledge a student might have, holds several different kinds of knowledge: an (impoverished and possibly incorrect) causal model of how an automobile engine and all of its parts work, a model (also impoverished and incorrect) of the reasoning process employed in troubleshooting, and a set of trouble-shooting experiences (both those of the student and those of the student watching the expert).

As a result of its learning experiences, CELIA's knowledge improves in several ways, allowing it to perform better: it refines and elaborates its models of the device and of the diagnosis process, it collects new troubleshooting experiences, and it learns the applicability of its troubleshooting experiences.

CELIA is an *active* intentional learner. As it watches and listens to a teacher performing some troubleshooting task, it *predicts* what the teacher will do or say next. It then *observes* or listens to the teacher. When predictions don't match what the teacher says or does, it attempts to *explain the discrepancy* and learn from it. Case-based reasoning's major roles are in aiding the prediction and explanation processes. Predictions that the student makes often come from previous troubleshooting cases remembered in the course of reasoning. Explanations of how it could have avoided a mistake often are in the form of cases it could have recalled at the time that would have allowed it to make the correct prediction. In addition, CELIA internalizes its experience observing the teacher into a troubleshooting case and uses those in later problem solving.

CELIA's prediction process makes use of all the different kinds of knowledge it has available. It predicts what a teacher might do next by retrieving cases that can make a prediction and by using its models, preferring the predictions made by cases over other predictions. This is because its models are known to be incomplete and buggy, while its cases are known to be instances of troubleshooting sequences that have been carried out. In essense, it predicts the thing that it would be most likely to do next were it doing the problem solving itself.

After discovering descrepancies between what it was expecting and what happened, CELIA uses cases again to help it explain its mistake and repair its knowledge. Of particular importance in this step is its process of figuring out how it could have avoided its mistake. In this step, it asks itself whether it already knew what it needed to know to make the correct prediction, and if so, why it didn't make the prediction. Often, making the right prediction depends on recalling a different case than the one it recalled. When that happens, it considers why it did not recall that case. In general, it is because it had it indexed poorly. CELIA refines the indexes on that case so that it would have been recalled in the current instance. It also refines the indexes of the case that made the poor prediction so that it will not be recalled under similar circumstances in the future. In this way, CELIA learns the applicability of its cases.

One other piece of CELIA related to cases is its process of making its experience watching the teacher into a troubleshooting case. Though on the surface this may seem like a trivial process, it is in fact, quite complex. The reason is this. CELIA is trying to learn how to do a task. Every cognitive task is composed of several steps, each of which has a cognitive subgoal attached to it (e.g., I need to make a hypothesis, I need to explain something). The better the student can do at explaining why the teacher did what he did and recording that, the more useful a case the student remembers will be during later reasoning. But explaining someone else's motivations is difficult. We know our own reasoning subgoals, but we have to guess at someone else's. Making the teacher's experience troubleshooting a car into a useful case for the student requires the student to analyze the things the teacher does, explain why the teacher did what he did, and connect together the different things the teacher said and did in a coherent way. CELIA's prediction process is key to doing this. CELIA makes sense of what the teacher is doing and sets itself up for active learning at the same time.

A major conclusion of this work is that early in learning, one of the most useful results of apprenticeship is the acquisition of cases of successful problem solving. These cases, if acquired, can guide later problem solving and prediction. Experimentation with CELIA shows that early in learning, remembering cases is the most powerful form of learning and other forms do not provide a major advantage. This is due partly to the limited domain knowledge the student has. With little domain knowledge, he can't form explanations easily about what is going on. Acquisition of cases, however, allows him to store new domain knowledge associated with the kinds of situations it is useful in. Later in the learning process, when many cases have been acquired, that domain knowledge can be migrated out of the cases to form a general model of the domain. Early on, however, the student does not know enough about applicability of the new pieces of knowledge he is acquiring to build a coherent model. After much experience, applicability of knowledge can be learned.

Acquisition of cases is more than just recording the sequence of events of some situation. A useful case is far more organized than being just a sequence. In particular, a case is most useful when the goals of the reasoner are associated with each step of the sequence of events. Associating goals with actions can be very hard. CELIA shows that it can be done as a natural consequence of the process of following along with what the instructor is doing as he solves a problem. Part of that process involves predicting the goals of the instructor, and based on those goals, what his actions will be. This understanding process provides the goals that need to be associated with steps in a sequence of events.

CELIA provides a model of learning that happens without strong knowledge of the domain. CELIA's learning processes allow it to bootstrap that knowledge from its experiences watching and listening to an expert problem solver. Through remembering those experiences, the student gets a good start toward being able to function in the domain. At the same time, he identifies places where he is lacking knowledge. The student interacts with the expert instructor to acquire some of that missing knowledge. Learning helps both future problem solving and future learning.

The instructor plays a crucial role in apprenticeship learning, but using the instructor to directly implant knowledge in the novice is unreasonable. CELIA shows that demands on an instructor don't have to be as excessive as that if the student takes responsibility for actively following the instructor's example. The instructor's job is to solve problems in front of students, to provide explanations at times when he thinks students may not know all the knowledge they need to follow him, to answer questions, and to give hints when students are having trouble. Effective apprenticeship requires the student to ask questions and the instructor to provide the right level of hints and explanations. Or course, more effort must be put into giving more concrete guidelines about what those hints and explanations need to look like.

In essense, CELIA models the ideal apprentice. It doesn't model the student who is having trouble, nor does it model the one who doesn't follow what the teacher is doing. As a model of the ideal, however, it can tell us much about the kinds of situations that enable successful learning. In particular, it makes the following testable cognitive predictions:

- Learning by observing and listening to an expert leads to significant increases in performance after only limited exposure to examples if the student plays an active role in understanding what the expert is doing.

CELIA's performance increased considerably as a result of its experiences observing the teacher solve problems. We cannot claim, however, that merely collecting sequences of actions is what made CELIA's performance improve. Key to its improvement was that it analyzed those cases, explaining to itself (as best it could) the teacher's actions, and making those experiences its own. That is, it always explained to itself *why* each step of the process was being carried out. This is consistent with research on self-explanation (Chi et al., 1989; Pirolli and Recker, 1994).

- During early learning of a task and domain, collecting cases is the single most useful type of knowledge acquired. Refining and adding to models of the domain and task result in far less improvement in capabilities.

Recall that CELIA adds to its models and refines its indexes in addition to adding cases to its case library. Recall also that it has access to all of that knowledge when it reasons. Cases provide only one of its knowledge sources.

Several ablation studies were run to see where CELIA's increase in performance was coming from, and the most significant increases were coming from having more cases available. When its ability to use its cases during problem solving was removed, so was much of its capability. Note that CELIA is not biased toward use of cases or collection of case knowledge such that this result is built in. It learns as much as it can from each of its experiences, collecting cases and adding to its models each time. While it prefers to make predictions based on cases, it is able to use its other knowledge fully. Early in learning, that knowledge is very buggy and very incomplete. It grows much slower than the case library does.

- It is difficult to improve the accuracy of case retrieval early on. Attempts to make students aware of when their knowledge is applicable may not succeed because they are missing crucial domain knowledge. As domain knowledge acrues, this problem abates.

This result says that early in learning students will put poor indexes on cases because they do not have enough knowledge to be able to determine the applicability of their knowledge. One could point it out, but they may not have enough knowledge to understand the explanation. As they make mistakes and are forced to analyze the indexes on their cases, indexing becomes better. And as students learn more, they are better able to index cases appropriately the first time.

- It is better to present a variety of types of problems early on rather than concentrating on several very similar ones.

- Repeated presentation of the same problem will not lead to long term improved performance. Variations on a problem work better.

These two results come simply from the statistics of experiments that were run, but they make a lot of sense. Experience with a wide variety of problems seeds the space of experiences broadly, implying that cases will be used often to make predictions. The use of cases to make predictions, even if the wrong predictions are made, is important, because it gives the reasoner an opportunity to refine its indexing and learn when its experiences and the things it knows are applicable. Seeding the case library with many similar cases means that later in problem solving, cases will be recalled to make predictions less often. At the same time, experience with variations of a problem allows a particular piece of the space to be examined from many different perspectives, allowing the applicability of the knowledge acquired to be learned. Are these two results contradictory? No. The first says that broad exposure should be given early on. The second says that if a student is having trouble with some concept, then give variations of the problem to the student to work on as drills, not the same one over and over.

## 4 Theory-Based Design of Interactive Multimedia

In order to design our interactive environment, our theoretical and empirical studies of troubleshooting needed complementary research in the area of design of interactive computing technologies for learning and aiding. Our ultimate goal is to combine these two threads in order to design an interactive multimedia environment for supporting the acquisition of troubleshooting skills and strategies.

Indeed, recent advances in computing technologies have resulted in the proliferation of multimedia repositories, libraries, and databases. These offer the potential for providing students with access to a wide variety of interconnected information resources and activities. Moreover, network-accessibility allows these resources to be distributed over wide distances.

However, access to multimedia system does not, by itself, guarantee *knowledge.* Multimedia systems instead should provide access to information and activities that support effective knowledge construction and learning by students. Unfortunately, little research has been conducted into how students will actually use such systems, what kinds of usage actually improves learning, and what types of educational materials these systems should provide.

To address these issues, we have developed a preliminary theory for designing educational multimedia systems based upon the cognitive and learning needs of students. This theory argues that the design of multimedia systems should not be based on the physical properties of the information (e.g., pictures, audio, video, etc.) Research on the effects of media types on learning have been inconclusive in demonstrating the superiority of one physical media over another reflects the fact that many factors, above and beyond simple media, affect a student's learning process. These factors include, for example, students' background knowledge, their motivation and interests, their learning strategies and goals, and the overall learning context (Recker and Pirolli, 1995). Therefore, rather than base the design of a hypermedia system on the physical properties of the information contained in the system, we propose that the indices and structure of the system should be based on cognitive aspects of the *users* of that information. By this, we mean that the access methods in a hypermedia systems should be "cognitively relevant" to the learning and information seeking goals of the user.

These *cognitive media types* form the basic building blocks during system design (Recker and Ram, 1994). They are based on an understanding of the learning and constructive processes of students, and encapsulate different methods or strategies for problem solving and learning. These strategies rely on specific media characteristics that facilitate specific learning activities, which in turn are enabled by specific kinds of physical media.

Text is an example of a physical media type. It can be used to represent several cognitive media types. For example, text can be used to present abstract, general instructions. Text can also be used to display instantiations of these concepts within examples, or to provide explanations and annotations. Animations and pictures are other examples of physical media types. Animations can be used to exemplify general or specific instantiations of dynamic displays of processes. Pictures can be used to display graphical relations among concepts. Each of these cognitive media facilitate different learning inferences.

Cognitive media are characterized in terms of the inferential processes of the human user rather than physical properties of the computer representation. Cognitive media encapsulate different kinds of problem solving information which might, in turn, be composed of many different physical media.

We believe that it is essential to focus on cognitive media in order to understand how best to design multimedia systems that can support novices in learning or training situations as well as aiding experts in on-the-job situations. We are developing a theory and taxonomy of cognitive media types that we believe are useful in learning situations. In addition, we are attempting to specify at which point during learning they may prove to be more advantageous. For example, is a general principle more useful at the beginning of a learning session, or after the student has gained some experience in a domain? Finally, we are interested in determining the physical media types that can best represent the different cognitive media types.

## Cognitive Media Types

We view multimedia information as composed of three layers: (1) media types, (2) media characteristics, and (3) cognitive media types.

At the lowest level are media types, for example, text, video, animation, etc. They are defined by characteristics of the physical (on-screen) media used to represent different kinds of information. Although often distinguished based on perceptual modalities (for example, visual vs. aural), they may be also characterized by the types of inferences that they facilitate. For example, figures (or diagrammatic representations) can facilitate spatial inferences (Larkin and Simon, 1987; Larkin, 1988) Thus, media types can be classified based on "cognitive" distinctions that depend not on physical characteristics of media but on the reasoning processes of users.

For example, consider the following example from Boden (Boden, 1991). A 20-foot rope

| Examples of Physical Media | Examples of Cognitive Media Types |
|---|---|
| Text | Abstract principles<br>Specific instructions<br>Annotated cases or examples<br>Explanations<br>Annotations |
| Animations | Dynamic display of process |
| Pictures | Graphical display of relation among processes<br>Examples<br>Diagrams |
| Sound | Voice-over of text<br>Warnings<br>Summaries |

Table 4: Partial taxonomy of physical media and cognitive media types

is tied to and hanging between two buildings that are some distance apart. Given that the lowest point of the rope is 10-feet below the tethered ends, how far apart are the buildings? In this example, most people tend to draw a diagrammatic representation of the problem showing the two buildings and a rope hanging in some sort of arc between them—this representation enables all kinds of fancy mathematical (geometric) reasoning which in this case is absolutely the wrong way to solve the problem. Boden points out that mathematically-inclined people tend to take a long time to solve this problem because they usually start by drawing a picture of the hanging rope, whereas less math-sophisticated people find it easier to solve because they do not rely on diagram-based geometric reasoning. The point is that a type of physical media—here, a diagram—can affect the course of problem solving by facilitating certain kinds of inferences and making others harder.

At the next level are the media characteristics (Kozma, 1991). These are a characterization of the kinds of problem-solving actions that people might perform during a task. For example, *zooming* is a problem-solving action that focuses attention on and highlights the details of a problem situation; such an action is easier in a diagrammatic representation than in a symbolic one.

Finally, at the level above media characteristics, are cognitive media types. For example, *zooming* is a particular problem solving action—more precisely, a particular schema for a problem solving action which results in certain inferences—whereas at the higher level one might have schemas for problem solving strategies as a whole. So, for example, one may resort to case-based reasoning (Kolodner, 1993), go back to first principles using basic definitions and equations, or reason using constructive simulations (Soloway et al., 1992).

A case is a type of media (characterized at the cognitive level) that facilitates the former (case-based reasoning) problem solving strategy. Using a case requires (among other things) zooming into the differences between the case and the problem situation so that the differences can be characterized and the case suitably adapted. Zooming in and adaptation are specific problem solving actions that are facilitated by different media characteristics; these characteristics, in turn, are enabled by specific physical media.

To take another example, consider *Emile*, a multimedia environment in which subjects learn physics by constructing physics simulations (Soloway et al., 1992). While the physical media characteristics - animations - are similar to those of many other animation-based learning environments, the crucial difference lies in the reasoning processes used by the student—here, constructive experimentation rather than passive observation.

We are developing a taxonomy of the kinds of cognitive media types used in learning and how these are best represented within of physical media types. We are also interested in determining the kinds of learning inferences they facilitate. Table 4 shows a partial taxonomy and examples of types of physical media we are using, and the kinds of cognitive media that can be represented within each physical media.

It is important to note that physical media types and media characteristics can be defined without a strong appeal to a cognitive theory of reasoning and learning. This is not true of cognitive media types—the structure, content, and use of cognitive media types is a function of the researcher's theory of cognition. This has two implications. First, the design of effective hypermedia environments must take into account a cognitive theory of how the user interacts with the environment. In particular, the design must be based on considerations of how the user will think, use, interact with, and learn from the environment. The second implication is that such cognitively-based hypermedia environments are not just likely to be more effective, but in addition might be used to test the validity of the cognitive principles on which they are based.

## 5 Implications for Design

The research and results we have presented thus far have many implications for the design of an interactive learning environment designed to help students learn and engage in troubleshooting. The research has implications for (1) how to structure the troubleshooting problem solving environment, (2) how to help learners acquire troubleshooting skills, (3) how to provide access to information in support of the task, and (4) how to support troubleshooters during the troubleshooting process.

## Learning to Troubleshoot

Much research in learning theory stresses the active involvement of learners during the process. In CELIA, the active involvement primarily takes the form of predicting and explaining the teacher's actions (Redmond, 1992). This conclusion is also supported by other research on the *self-explanation* effect. Engaging in self-explanations has been shown to reliably correlate with positive learning learning outcomes (Chi et al., 1989; Fergusson-Hessler and de Jong, 1990; Pirolli and Recker, 1994; Recker and Pirolli, 1995). Furthermore, CELIA suggests that students need to notice failures between their predictions and actual outcomes, as these can serve as triggers for useful learning opportunities (Redmond, 1992).

The teacher models the task for the student, thereby providing an opportunity for students to learn by example (LeFevre and Dixon, 1986; Lewis and Anderson, 1985; Pirolli and Anderson, 1985; Sweller and Cooper, 1985). The teacher, on the other hand, tries to encourage active student involvement by structuring the task into prediction and explanation phases. Furthermore, during the explanation phase, the teacher facilitate the learner's task by making goals and actions explicit and by only considering one hypothesis at a time. This process may lead the student toward acquiring better troubleshooting strategies.

In sum, the system should:

1. Encourage active, intentional learning.

2. Encourage prediction of teacher's next step.

3. Highlight prediction failures.

4. Encourage explanation of discrepancies between what it was expecting and what happened. This can lead to a re-indexing of cases and an understanding of the applicability of cases.

5. Model problem solving for learners. This may allow the learner to acquire new cases through observation. The steps performed by the system should be presented in a way that allows easy derivation and understanding of cases.

6. When possible, the system should model an initial set of examples or cases that cover a wide range of the problem solving to be learned.

## Structuring Troubleshooting

The theoretical framework suggests ways in which the troubleshooting process may be structured. Specifically, the process should be structured into (1) problem elaboration, (2) hypothesis generation, (3) hypotheses testing, and (4) hypothesis evaluation phases.

The TS cognitive model and the specific protocol simulations suggest coaching and modelling tactics that could be used by the system. For example, during coaching and modelling, the teacher can explicitly structure the troubleshooting process so as to communicate the general framework of hypothesis generation, testing, and evaluation (Merrill and Reiser, 1994). During hypothesis generation, the learner can be given the opportunity to acquire larger symptom fault-sets by engaging in problem solving, and by engaging in periods of problem solving reflection. Reflecting over problem solving episodes has been suggested to lead to improved learning (Pirolli and Recker, 1994; Lajoie and Lesgold, 1989). Similarly, students can be given support for developing better causal models as these help to focus the search (Merrill and Reiser, 1994; White, 1993).

As suggested by the results from our empirical studies, learners should be given the opportunity to acquire a wider range of troubleshooting *strategies*. These experiences can help them explore the time and cost constraints present in different troubleshooting contexts. For example, as they gain experience, students should be encouraged to generate multiple hypotheses. Similarly, coaching tactics can be used to help students learn to adapt to the cost structure of the particular device and troubleshooting situation by helping them develop strategies for identifying tests that are cheap to conduct, yet are maximally informative. They should also learn to select tests that serve to *disconfirm* a maximal number of possible hypotheses.

In sum, the system's interface should

1. Structure troubleshooting process into hypothesis generation and testing phases.
2. Act as external scratch pad to keep track of processes.
3. Encourage reflection over problem solutions.
4. Encourage acquisition of troubleshooting strategies.

## Designing for Fidelity

The results from the empirical studies using Turbinia provide important feedback and implications for the design of software simulation and learning environments. For example, the results raise issues in the kinds of *fidelity* that are designed in a computer simulation in order to model the corresponding external system (Collins, in press). In one common approach, the system designer attempts to maintain an *epistemic* fidelity with the external system (Wenger, 1987). In this view, the representations used in the simulation capture the structural organization of knowledge assumed to be held by the expert. An implicit instructional assumption is that the learner internalizes the expert's mental model through interacting with the simulation. In Turbinia, the organization of systems, subsystems, and components were modeled on an expert's understanding of the marine power plant (Govindaraj and Su, 1988). In another approach, the simulation maintains a *dynamic* or temporal fidelity with the external system. That is, as the external system changes through time, the corresponding simulation reflects these changes. Turbinia also attempts to maintain a kind of dynamic fidelity, as the effects of faults propagate through the simulation with time.

We argue the results from our studies show that students develop strategies that are tuned to particular constraints present in the simulation environment. In short, the problem solving strategies develop from exposure to a range of situation contexts exhibited by the system. These results, then, suggest a third, important kind of fidelity to be considered during design – one that we have termed *fidelity of interaction.* In this view, the design of simulations of dynamical domains should attempt to preserve the costs and resource bounds of target situations. This will enable students to develop strategies that are faithful to the activity demands of real-world situations.

In sum, the system should:

1. Encourage acquisition of troubleshooting strategies that are sensitive to resource bounds of situations.

2. Design for fidelity of interaction.

## Intelligent Support and Aid

In our approach, the troubleshooting environment structures the task for the user, focus on the current context, manage details, and perform simple bookkeeping functions in order to reduce complexity. The interface attempts to keep salient the currently available options.

Given our view of troubleshooting as a complex process, with substantial strategic and monitoring components, the pedagogical goals go beyond simply communicating facts and rules. As such, the apprenticeship learning paradigm is more appropriate than simple tutoring. Students need to actively predict outcomes of the actions, and, equally importantly, to notice failures between their predictions and actual outcomes. Such failures can be serve as triggers for useful learning opportunities. Failures indicate either incorrect knowledge or knowledge gaps that need to be filled.

Support for the process can be offered by the interface in the following ways: by structuring the task, by focusing on the current context, by managing details, and by performing simple bookkeeping functions by acting as an external memory.

The troubleshooting task can be structured and focused as follows. The interface can support two context: hypothesis generation and test. The user can be one of these two contexts. The interface should keep track of useful symptom-fault pairs, which the user can consult. As the user gains experience, and becomes attuned to device behavior, this catalogue of indices can be added to.

The interface can also allow different views on the system, allowing either causal or topographic problem solving strategies. Cause knowledge allows reasoning via fluid flows to identify possible faulty components. Topographic knowledge uses component location and neighborhoods to identify potential faulty components.

The interface also keeps track of all hypothesis that have been considered and all tests that have been conducted. This capability provides an external memory for the user.

In sum, the system should:

1. Act as external memory to reduce cognitive load on user.
2. Manage complexity of process by breaking the problem down into a series of sub-steps.
3. Provide structured access to repository of information resources that can be used to help solve the problem. These information resources include libraries of solved cases, models of the system, and on-line manuals.

### Communication

Communication of the process takes place within an apprenticeship paradigm, and draws upon a model of the apprenticeship process (Collins et al., 1989). This process can be drawn upon to suggest interface features that best supports apprenticeship learning. For example, the troubleshooting goals and actions should be made explicit, as should be the transition between goals. Complexity can be reduced by only considering active hypotheses.

The interface can also support a period of reflection, by allowing the learner to reconsider his/her solution, including incorrect paths. The solution can also be compared to one from an expert.

In addition, since we view the troubleshooting as a dual process, the environment and the learner can alternate role playing. For example, the learner could choose to act as theorist while the simulation tutor as as experimenter, or vice versa.

## 6 Designing for a Help-Desk Domain

Based upon the work distilled in the previous sections, we have been implementing a prototype interactive environment for learning and aiding troubleshooting. Initially we intended our target domain to be car engine troubleshooting. However, it quickly became obvious that we lacked access to the necessary experts and to usable cases. At the same time, we had the unique opportunity of investigating diagnosis and troubleshooting in the context of the telephone help-desk work environment. In this domain, help-desk specialists for a large computer company receive telephone calls from customers experiencing problems with their computer systems. Through conversing with the customer, asking questions, and drawing upon information resources, they diagnose the problem and propose solutions.

The help-desk work domain is an ideal test-bed for our design. The help-desk specialist engages in complex troubleshooting, under difficult and time-constrained conditions. The customers (and the company) want immediate solutions to their problems. However, the customers are sometimes not expert enough to correctly articulate problems and symptoms. In addition, the help-desks are limited in the kinds of diagnostic tests that they can perform, since they are primarily limited to conversations and question-asking.

Finally, with the wide use computer networks, computers, databases, and information technologies, the specialist has access to a wide variety of information resources that can be used to help diagnose the problem. These resources are stored in various formats, are accorded different kinds of status, and are retrieval through a variety of means. While answers to assist a current diagnosis problem are highly likely to be contained within these information resources, the precise answer is difficult to obtain quickly. Information flows from 3 general sources. First, a specialist may ask another specialist. This can be a reliable means, if the appropriate specialist is know. On the other hand, it can be intrusive, and it can be difficult to consult another person while in the middle of a phone call.

Second, manuals for the various software and hardware products can be consults. These manuals are often paper-based and thus do not have electronic indexing. In addition, they are typically not written in a way that supports diagnosis.

Third, specialists have access to a variety of on-line databases. These databases store prior calls handled by specialists, newsgroup articles that discuss problems, and official problem reports and solutions

The specialist thus performs diagnosis in an information-rich environment. While there are certainly a multitude of advantages accompanying the new ease in accessing information, there also disadvantages. For example, with ubiquitous information, the specialist must carefully consider their means for searching information in order to avoid an overwhelming response to queries. They must balance the benefits of information retrieval against the costs of processing that information. They must consider which sources to search. They must more carefully consider the origin of the information in order to retrieve the highest quality information. They must more carefully filter what is retrieved.

These problems become particularly salient in time-critical situations. Users performing information-rich tasks with limited time to make decisions and consider information sources must develop *satisficing* strategies (Card et al., 1993; Simon, 1973). That is, they must quickly decide the most relevant source, devise means for querying these sources while avoiding a large query return, then filter and make-sense of the returned information.

In the description of the prototype that follows, our scenarios are drawn from the help-desk situation. However, we note that the design of the support environment is sufficiently generic to transfer to other help-desk troubleshooting situations.

## The Prototype

Figure 1 shows the initial screen of the help-desk troubleshooting system (HD-TS). From this screen, the learner can select the troubleshooting mode desired. Currently, three modes are supported.

Insert Figure 1 about here

1. *Observing Cases.* In this mode, the novice can observe an articulate teacher model the troubleshooting process via a series of cases. Here we assume that the learner does not know anything about the process on engaging in troubleshooting, and that a productive starting point is to observe an expert model the process.
2. *Interactive Observation: Learning by predicting and explaining.* In this mode, the more advanced learner engages in and is actively involved in playing out the case studies. The learner engages in the same activities that CELIA, our computational model of the ideal apprenticeship learner, performs.
3. *Troubleshooting.* In this mode, the troubleshooter solves diagnosis problems, with the assistance and support of the environment.

Figure 2 shows the generic, start-up screen after selecting any of the modes. The top part of the screen shows the mode – in this case "Case Studies." In addition, the initial complaint is listed. The middle part of the screen contains real-estate for managing troubleshooting steps, scratch pads for keeping track of proposed hypotheses, conducted tests, and the ongoing problem solving context. The bottom of the screen helps the learner keep track of troubleshooting steps. The right part of the screen provides access to various informational media. Currently implemented cognitive media are:

Insert Figure 2 about here

1. *Manuals.* Access to relevant pages in manuals.
2. *Experts.* Access to on-line human experts via video teleconferencing. They can be used for advice and consulting
3. *Company Cases.* Access to repository of cases solved by other staff members.
4. *Personal Cases.* Access to repository of cases solved by the learner.

In the next sections, we describe the three troubleshooting modes in more detail.

## Observing Cases

In this mode, the user is assumed to be a rank novice. Here, the learner can observe an articulate teacher as s/he engages in troubleshooting. As suggested by CELIA, the user learns by observing the articulate teacher model the troubleshooting process. In the upper right hand part of the screen is a button labeled "Continue." The learner repeatedly clicks on this button to step through the process. With each mouse click, the expert performs the next step. Explanations and heuristics for each step are displayed in the window labeled "Strategies."

A sample case study scenario is as follows. With the first mouse click on "Continue," a button labeled "What's Wrong?" appears. A mouse click on "Continue" causes the "What's Wrong" button to be clicked, which, in turn, displays the complaint. In addition, a button labeled "Verify Complaint" appears. (Simultaneously, the troubleshooting step is highlighted at the bottom of the screen. Similarly, the strategy window explains the heuristics underlying each step). Figure 3 shows the screen at this point. A mouse click on the "Continue" button and the "Verify" button is clicked to display that the complaint has been verified.

At the same time, the "What's Causing It?" button appears. This represents the hypothesis generation phase. A mouse click on the "Continue" button causes several hypotheses to be proposed. These are stored in the "Active Hypotheses" window in the middle of the screen. In addition, the "Select Hypothesis" button appears. Figure 4 shows the screen at this point. Note that a relevant page from the manual and an additional company case have appeared, selected because they matched the active hypothesis.

Insert Figures 3, 4, 5 about here

A mouse click on the "Continue" button, and a leading hypothesis is selected. This hypothesis is also highlighted in the "Active Hypotheses" window. At the same time, the "How to Check?" button appears. This represents the hypothesis testing phase. Figure 5 shows the screen at this point. In addition, note in this screen that the user has chosen to view a matching company case.

A mouse click on the "Continue" button, and the "How to Check?" button is clicked, displaying several testing methods. These methods are stored in the "Conducted Test" window, for easy referral by the learner. A diagram of the relevant area is also displayed.

At the same time, the "What Happened?" button appears. The represents the verification/evaluation phase, in which test results are reviewed to suggest further actions. Figure 6 shows the screen at this point.

Figure 7 shows the screen after a second iteration through hypothesis selection and testing. The problem solving context, hypothesis, and testing boxes have been updated.

Insert Figures 6, 7 about here

In this mode, we expect students to learn by observing the system model the troubleshooting process. In particular, since the system acts as an articulate and explicit expert, we expect students to understand the phases of troubleshooting, the processes involved in each phase, the kinds of strategies applicable at each phase, and general metacognitive awareness of how diagnosis works.

## Interactive Observation

As suggested by CELIA, we want to encourage the learner to take an active role in learning by engaging in cycles of prediction and explanation as the instructor models troubleshooting. CELIA modeled this using previous troubleshooting cases remembered in the course of reasoning to generate a prediction of the instructor's next step. When predictions don't match what the teacher says or does, it attempts to *explain the discrepancy* and learn from it. Explanations of how it could have avoided a mistake often are in the form of cases it could have recalled at the time that would have allowed it to make the correct prediction. In addition, CELIA internalizes its experience observing the teacher into a troubleshooting case and uses those in later problem solving (Redmond, 1992).

We implemented this by segmenting each step into phases where the students generate their own predictions. These are then matched against the instructor's next step. If the predictions do not match, the student is prompted to produce an explanation or to access information resources.

The scenarios in the "Interactive Case Study" are similar to the case study scenarios, except in three regards:

In particular, in this mode, the learner is asked:

1. to selected each subsequent troubleshooting step, rather than relying on the "Continue" button.
2. to predict the instructor's steps
3. to compare predictions with the instructor's actions
4. if they do not match, to engage in explanation of the discrepancy
5. to review and reflect on each completed scenario. The on-line review phase then becomes input in the personal case library.

A sample case study scenario is as follows. With the first mouse click, a button labeled "What's Wrong?" appears. A click on the "What's Wrong" button displays the complaint. In addition, a button labeled "Verify Complaint" appears. (Simultaneously, the troubleshooting step is highlighted at the bottom of the screen. Similarly, the strategy window explains the heuristics underlying each step). A click on the "Verify" button causes a display that the complaint has been verified. A relevant company case appears at this point, selected because it matched the complaint.

At the same time, the "What's Causing It?" button appears. This represents the hypothesis generation phase. A click on this button and several hypotheses are proposed. These are stored in the "Active Hypotheses" window in the middle of the screen. In addition, the "Select Hypothesis" button appears. A click on this button, and the learner is asked to predict the instructor's next step in a pop-up window.

After the learner's prediction, a leading hypothesis is selected. This hypothesis is also highlighted in the "Active Hypotheses" window. At this point, a relevant page from the manual and an additional company case appear, selected because they matched the selected hypothesis.

At the same time, the "How to Check?" button appears. This represents the hypothesis testing phase. A click on this button, and the learner is asked to predict the instructor's next step in a pop-up window. After entering the predictions, the "How to Check" button is clicked, displaying several testing methods. These methods are stored in the "Conducted Test" window, for easy referral by the learner. A diagram of the relevant area is also displayed.

At the same time, the "What Happened?" button appears. The represents the verification/evaluation phase, in which test results are reviewed to suggest further actions.

After completing the interactive case study, the learner is asked to reflect on the episode, by clicking on the "Reflect" button. A pop-up window appears in which the learner enters the important symptoms, hypotheses, and tests in the episode. These data are then entered and indexed in the personal case library. They can thus be retrieved during subsequent troubleshooting episodes.

## Troubleshooting: Trying out what's learned

In this mode, the help-desk environment acts as a problem solving aid. Here, some of the interface scaffolding is left in place, while other aspects are faded.

The interface retains the structure that breaks the troubleshooting process into its several sub-components. Thus, students still engage in the processes of complaint elicitation and complaint verification, followed by hypothesis generation, testing, and evaluation. This structure helps the user plan and manage the details of the task. However, the user has control of the path through the process.

In addition, the environment continues to act as an external scratch pad. As before, a window is used to keep track of the complaint and its elaborations. Windows are used to keep track of active hypotheses, and tests that have been conducted.

Our on-sight studies of help-desk staff as they perform their job suggest a crucial aspect of job performance: managing and filtering the vast array of information resources that is available. Their job is performed in a world of truly ubiquitous information. This information is available from several sources: digital, paper, or human.

We observed help-desk staff consulting the following resources while handling calls:

- Digital. Help-desk staff have a variety of on-line repositories of information that they can consult during a call. Access is typically performed via keyword search, though the help-desk staff can consult these off-line. These repositories include databases of other calls handled by help-desk staff, informal database of notes posted by other specialists, and and official databases of notes posted by experts.

- Paper. Help-desk staff may consult manuals for software and hardware that a customer has problems with

- Human. Help-desk staff may ask other staff members for assistance during a call, or after.

The access and use of such information is difficult to analyze and characterize because of its ubiquity. In our initial approach, we have focused on digital information simply because of its availability. In addition, many summary statistics are available describing use by employees during the course of performing their work.

## 7 Conclusion

We have described a prototype interactive, multimedia, learning environment that helps users learn and engage in effective troubleshooting. The system is intended to help specialists learn how people perform diagnosis in the context the telephone help-desk work environment. As the user gains expertise, the system switches its role into one that helps aid and structure the process.

The design of the environment draws upon two important strands of related research. First, we reviewed studies on how people perform complex troubleshooting tasks. In addition, we reviewed results from a model of how people learn to troubleshoot in an apprenticeship learning situation. This research thus provides an interesting case study of how cognitive science research can be used to inform the design of tools intended for authentic and complex work domains.

The research on learning troubleshooting suggests that learners benefit from observing experts model the troubleshooting process. The benefits are the greatest if the learner attempts to predict and explain the instructors steps. With the process, learners can acquire experiences (or cases) that may be helpful during subsequent problem solving (Kolodner, 1993; Redmond, 1992).

Our research on troubleshooting problem solving suggests that users vary in the kinds of strategies employed during troubleshooting. In particular, similar to results found in the domain of scientific discovery, some are primarily *experimenters*, while the more successful troubleshooters are primarily *theorists*. Finally, the strategies employed are clearly sensitive to the task resources available. This suggests that an interactive learning environment support *fidelity of interaction* so that students learn strategies that are sensitive to the constraints of the particular troubleshooting context.

Second, we described research that focuses on the theory-based design of interactive learning environments. We argued that access to multimedia information should be structured in terms of the cognitive and learning processes that it supports, rather than on the physical format and properties of actual piece of information.

We then described a prototype of an interactive, multimedia learning environment, called HD-TS. The system supports users in learning and performing troubleshooting in the telephone help-desk task domain.

In future work, we plan to continue the implementation the troubleshooting environment. The development work will be complemented by field studies of the help-desk work environment, and walk-throughs of the system with members of the help-desk staff. These data from our formative evaluations will enable "user-centered design," will allow us to resolve design decisions, and to allow us early feedback on the quality and effectiveness of our design.

A particular focus of our future work is on incorporating digital data from the real work environment into the HD-TS system. As discussed, the help-desk specialists have access to a wide variety of information resources that they draw upon in order to help during their diagnosis process. In fact, they have access to much more information that can reasonably be used within the time-constraints of a telephone call and diagnostic episode. Therefore, we wish to develop computationally feasible methods that serve to index, retrieve, and filter real-world information and cases in a manner that is useful for the specialists. The filtering and presentation of the information must be made in a way that is easily incorporated in to the current problem solving situation. This problem is becoming increasingly prevalent in most information-rich tasks, as we enter in the the Age of Information.

## Acknowledgements

This research is supported by the Army Research Institute for the Behavioral and Social Sciences under Contract No. MDA-903-90-K-112 to Janet Kolodner. The development of Turbinia-Vyasa was sponsored by the Office of Naval Research and contract N00014-87-K-0482 from Manpower, Personnel, and Training R & D Program.

## References

Boden, M. (1991). *The Creative Mind: Myths and Mechanisms.* Abacus, London.

Card, S., Pirolli, P., and Mackinlay, J. (1993). The cost-of-knowledge characteristic function: Display evaluation for direct-walk dynamic information visualizations. In *CHI'94: Human Factors in Computing Systems*, pages 238–244, Boston, MA. ACM.

Chi, M., Bassok, M., Lewis, M., Reimann, P., and Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13:145–182.

Collins, A. (in press). Design issues for learning environments. In Vosniadou, S., deCorte, E., Glaser, R., and Mandl, H., editors, *International Perspectives on the Psychological Foundations of Technology-Based Learning Environments.* Springer Verlag, New York.

Collins, A., Brown, J., and Newman, S. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In Resnick, L., editor, *Knowing, Learning and instruction: Essays in honor of Robert Glaser.* Lawrence Erlbaum, Hilldale, NJ.

Darden, L. (1990). Diagnosing and fixing faults in theories. In Shrager, J. and Langley, P., editors, *Computational Models of Scientific Discovery and Theory Formation*, pages 319–354. Morgan Kaufmann Publishers, San Mateo, CA.

de Kleer, J. (1990). Using crude probability estimates to guide diagnosis. *Artificial Intelligence*, 45:381–391.

Fergusson-Hessler, M. and de Jong, T. (1990). Studying physics texts: Difference in study processes between good and poor solvers. *Cognition and Instruction*, 7(1):41–54.

Freedman, E. (1992). Scientific inductions: Individual versus group processes and multiple hypotheses. In *Proceedings of the Annual Conference of the Cognitive Science Society*, Cambridge, MA. Erlbaum.

Govindaraj, T. and Su, Y. (1988). A model of fault diagnosis performance of expert marine engineers. *International Journal of Man-Machine Studies*, 29:1–20.

Klahr, D. and Dunbar, K. (1988). Dual space search during scientific reasoning. *Cognitive Science*, 12:1–48.

Kolodner, J. L. (1993). *Case-Based Reasoning.* Morgan Kaufman Publishers, San Mateo, CA.

Kozma, R. (1991). Learning with media. *Review of Educational Research*, 61(2):179–211.

Laird, J., Rosenbloom, P., and Newell, A. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64.

Lajoie, S. and Lesgold, A. (1989). Apprenticeship training in the workplace: Computer-coached practice environment as a new form of apprenticeship. *Machine-Mediated Learning*, 3:7–28.

Lancaster, J. and Kolodner, J. (1988). Varieties of learning from problem solving experience. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

Larkin, J. H. (1988). Display-based problem solving. In Klahr, D. and Kotovsky, K., editors, *Complex Information Processing: Essays in honor of Herbert A. Simon.* Lawrence Erlbaum, Hillsdale, NJ.

Larkin, J. H. and Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:64–100.

LeFevre, J. and Dixon, P. (1986). Do written instructions need examples? *Cognition and Instruction*, 3:1–30.

Lewis, M. and Anderson, J. (1985). Discrimination of operator schemata in problem solving: Learning from examples. *Cognitive Psychology*, 17:26–65.

Merrill, D. and Reiser, B. (1994). Scaffolding effective problem solving strategies in interactive learning environments. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Hillsdale, NJ. Erlbaum.

Munro, A. (1993). Authoring interactive graphical models for instruction. In Towne, D., de Jong, T., and Spada, H., editors, *Simulation-based experiential learning*, volume 122, pages 33–45. NATO ASI Series F, Springer Verlag, Heidelberg.

Newell, A. (1990). *Unified Theories of Cognition.* Harvard University Press, Cambridge, MA.

Pirolli, P. and Anderson, J. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, 39(2):240–272.

Pirolli, P. and Recker, M. (1994). Learning strategies and transfer in the domain of programming. *Cognition and Instruction*, 12(3):235–275.

Recker, M., Govindaraj, T., and Vasandani, V. (1994). Troubleshooting strategies in complex domains. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 739–744. Lawrence Erlbaum, Hilldale, NJ.

Recker, M. and Pirolli, P. (1995). Modelling individual differences in students' learning strategies. *Journal of the Learning Sciences*, 4(1):1–38.

Recker, M. and Ram, A. (1994). Cognitive media types as indices for hypermedia learning environments. In *Proceedings of the AAAI Workshop on Indexing and Reuse in Multimedia Systems*, Menlo Park. AAAI.

Redmond, M. (1992). *Learning by Observing and Understanding Expert Problem solving.* PhD thesis, Georgia Institute of Technology.

Simon, H. A. (1973). *The Sciences of the Artificial.* MIT Press, Cambridge, MA.

Soloway, E., Guzdial, M., Brade, K., Hohmann, L., Tabak, I., Weingrad, P., and P., B. (1992). Technological support for the learning and doing of design. In Jones, M. and Winne, P., editors, *Adaptive Learning Environments.* NATO ASI Series, Springer Verlag, Berlin.

Sweller, J. and Cooper, G. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 7:1–39.

Towne, D. (1993). Teaching and learning diagnostic skills in a simulation environment. In Towne, D., de Jong, T., and Spada, H., editors, *Simulation-based experiential learning*, volume 122, pages 149–164. NATO ASI Series F, Springer Verlag, Heidelberg.

Towne, D. and Munro, A. (1988). The intelligent maintenance training system. In Psotka, J., Massey, L., and Mutter, S., editors, *Intelligent Tutoring Systems: Lessons Learned*, pages 479–531. Lawrence Erlbaum Associates, Hillsdale, NJ.

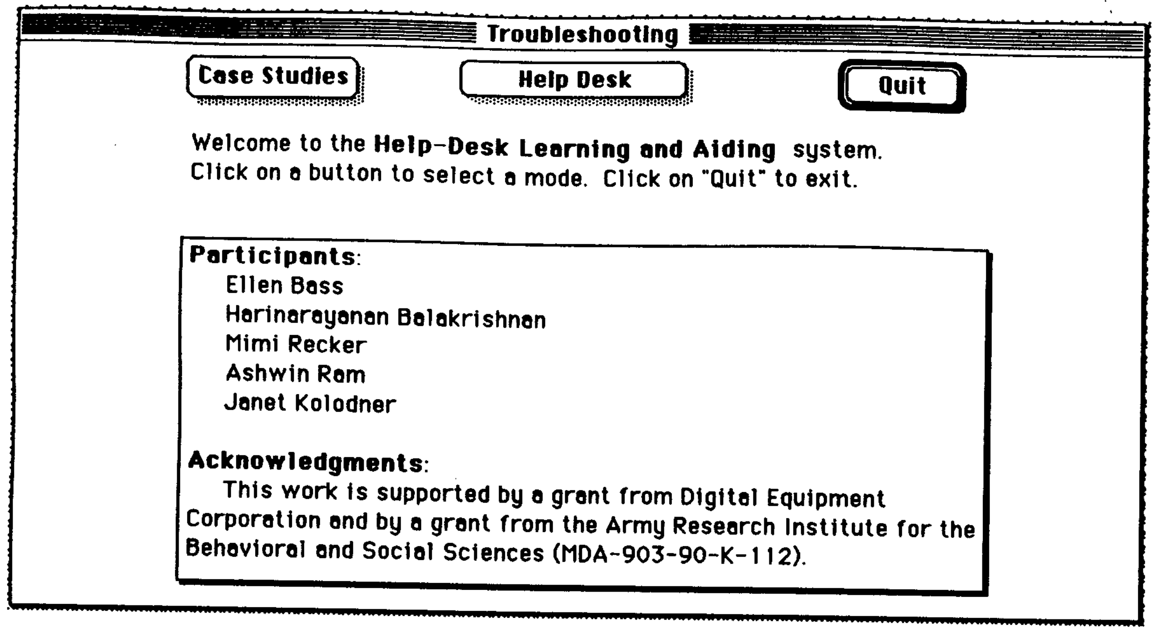
Vasandani, V. and Govindaraj, T. (1993). Knowledge structures for a computer-based training aid for troubleshooting a complex system. In Towne, D., de Jong, T., and Spada, H., editors, *Simulation-based experiential learning*, volume 122, pages 17–32. NATO ASI Series F, Springer Verlag, Heidelberg.

Vasandani, V. and Govindaraj, T. (1994). Knowledge organization in intelligent tutoring systems for diagnostic problem solving in complex dynamic domains. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-25.

Wenger, E. (1987). *Artificial Intelligence and tutoring systems.* Morgan Kaufmann Publishers, Los Altos, CA.

White, B. (1993). Thinkertools: Causal models, conceptual change, and science education. *Cognition and Instruction*, 10(1).

White, B. and Frederiksen, J. (1990). Causal model progression as a foundation for intelligent tutoring systems. *Artificial Intelligence*, 42(1):99–157.

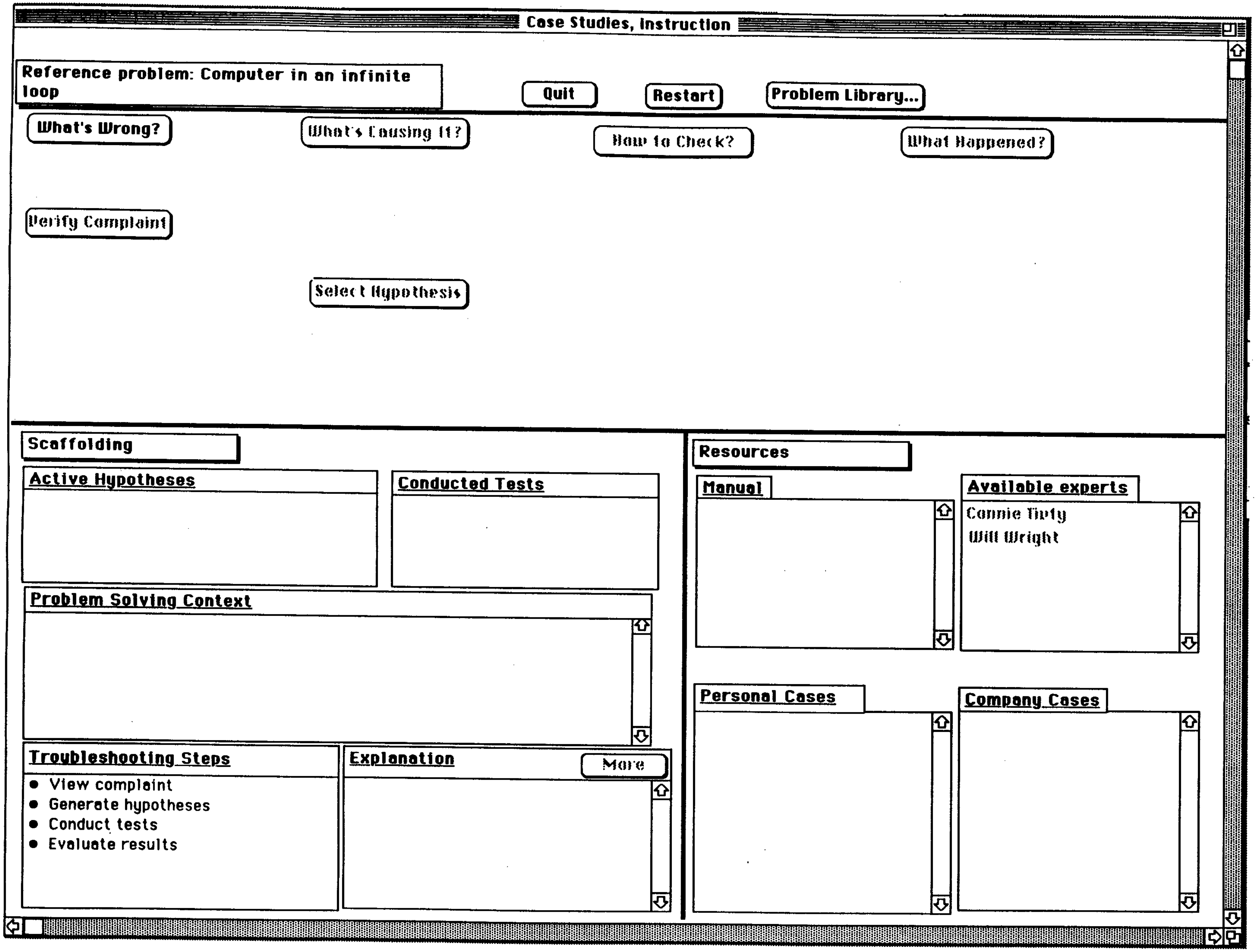


Welcome to the **Help-Desk Learning and Aiding** system.
Click on a button to select a mode. Click on "Quit" to exit.

**Participants**:
Ellen Bass
Harinarayanan Balakrishnan
Mimi Recker
Ashwin Ram
Janet Kolodner

**Acknowledgments**:
This work is supported by a grant from Digital Equipment Corporation and by a grant from the Army Research Institute for the Behavioral and Social Sciences (MDA-903-90-K-112).

Figure 1

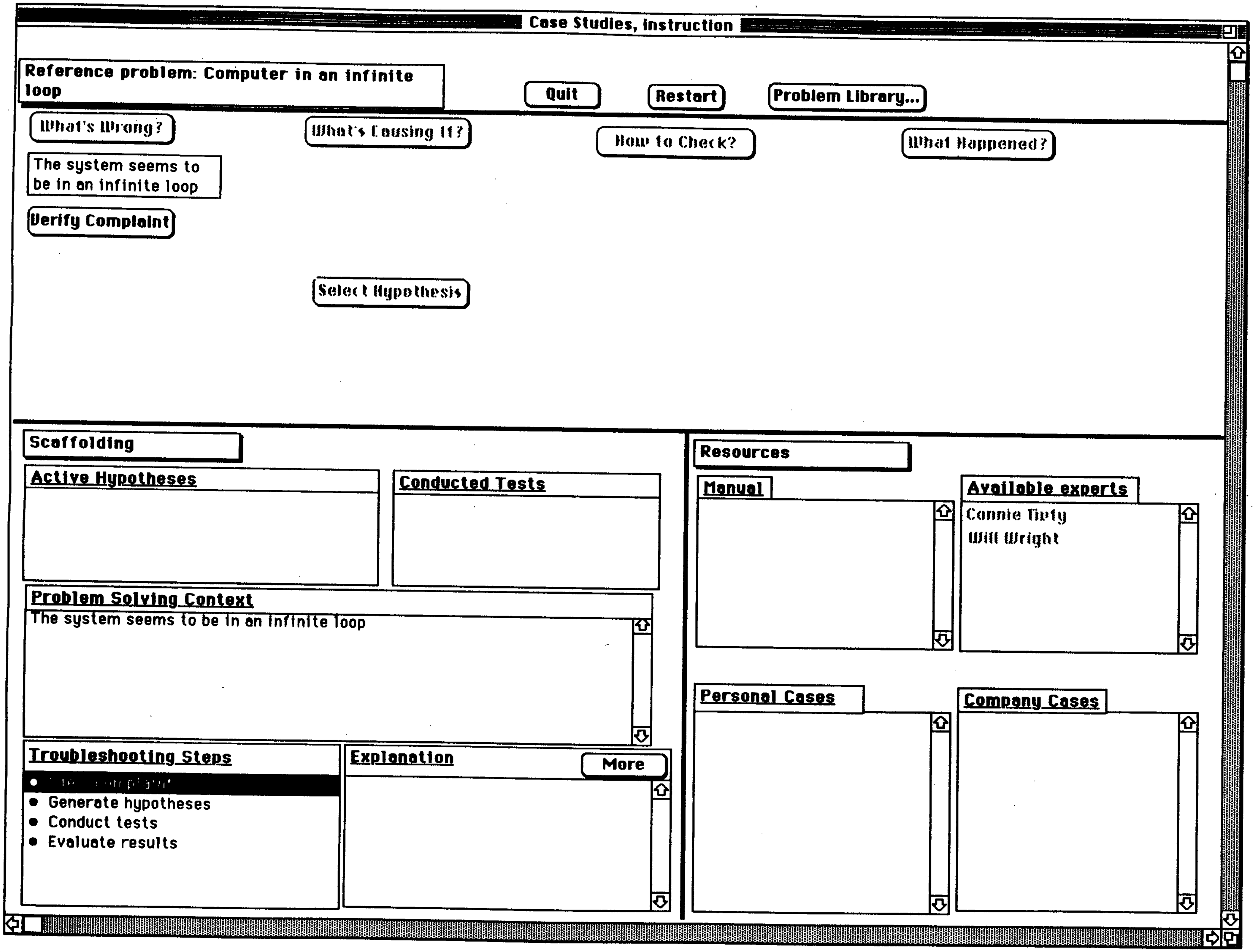


Reference problem: Computer in an infinite loop

Troubleshooting Steps
- View complaint
- Generate hypotheses
- Conduct tests
- Evaluate results

Available experts
Connie Tirty
Will Wright

FIG. 22



Reference problem: Computer in an infinite loop

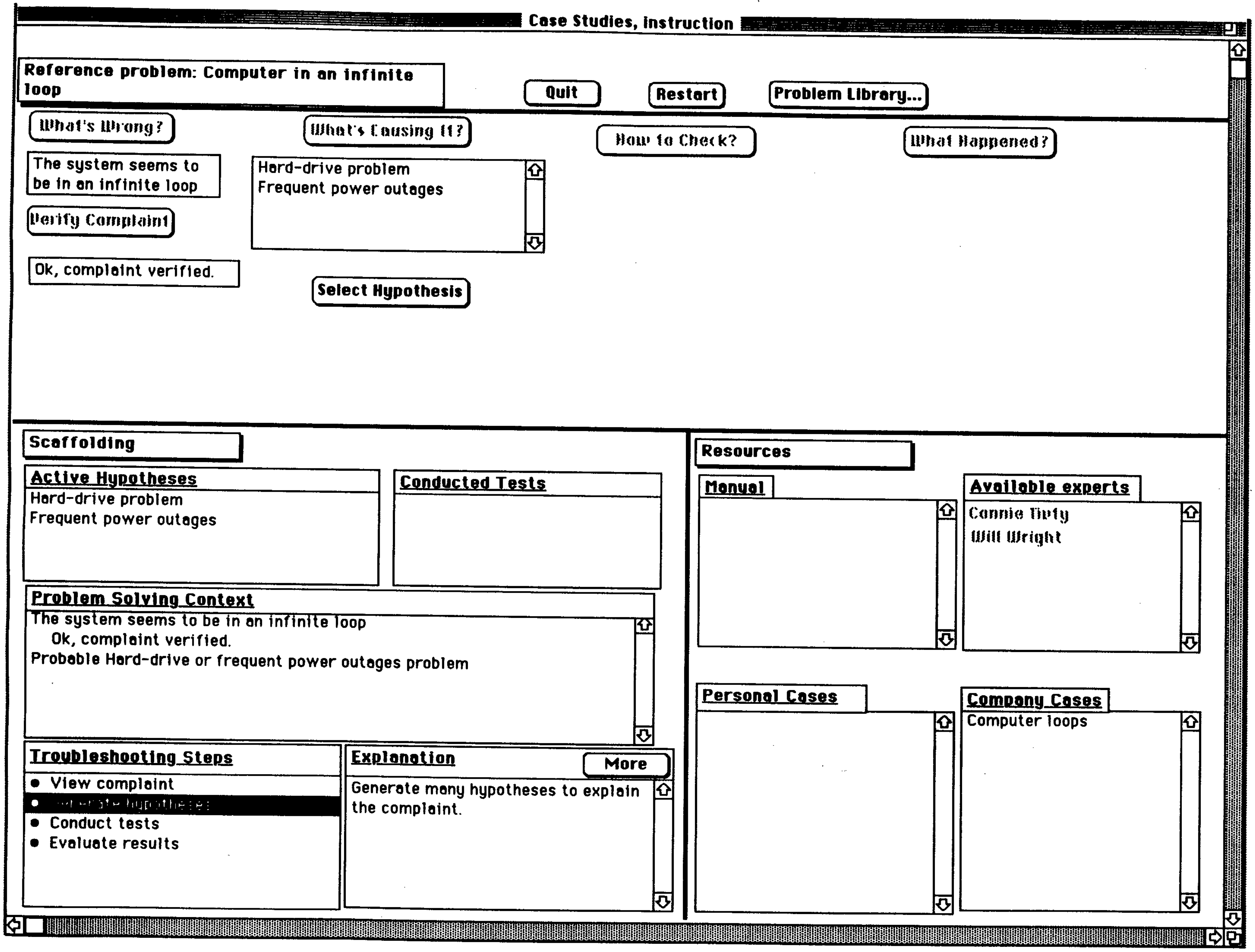
The system seems to be in an infinite loop

The system seems to be in an infinite loop

- [illegible]
- Generate hypotheses
- Conduct tests
- Evaluate results

Connie Tirty
Will Wright

FIGURE 3



Case Studies, instruction

Reference problem: Computer in an infinite loop

The system seems to be in an infinite loop

Ok, complaint verified.

Hard-drive problem
Frequent power outages

**Scaffolding**

**Active Hypotheses**
Hard-drive problem
Frequent power outages

**Conducted Tests**

**Problem Solving Context**
The system seems to be in an infinite loop
Ok, complaint verified.
Probable Hard-drive or frequent power outages problem

**Troubleshooting Steps**
- View complaint
- Generate hypotheses
- Conduct tests
- Evaluate results

**Explanation**
Generate many hypotheses to explain the complaint.

**Resources**

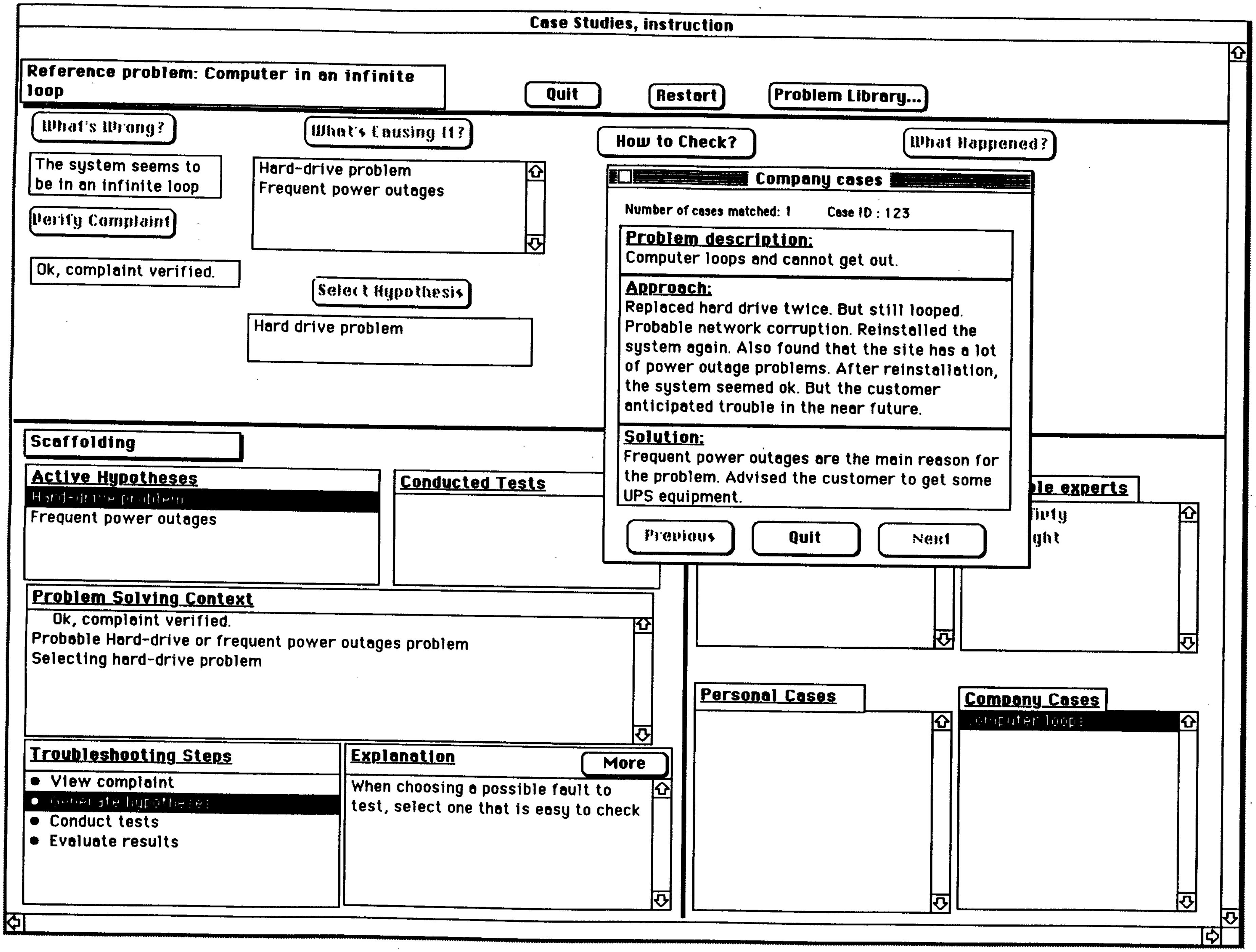
**Manual**

**Available experts**
Connie Tiety
Will Wright

**Personal Cases**

**Company Cases**
Computer loops

FIG 7


Case Studies, instruction

Reference problem: Computer in an infinite loop

Quit   Restart   Problem Library...

What's Wrong?   What's Causing It?   How to Check?   What Happened?

The system seems to be in an infinite loop

Verify Complaint

Ok, complaint verified.

Hard-drive problem
Frequent power outages

Select Hypothesis

Hard drive problem

**Company cases**

Number of cases matched: 1   Case ID : 123

**Problem description:**
Computer loops and cannot get out.

**Approach:**
Replaced hard drive twice. But still looped. Probable network corruption. Reinstalled the system again. Also found that the site has a lot of power outage problems. After reinstallation, the system seemed ok. But the customer anticipated trouble in the near future.

**Solution:**
Frequent power outages are the main reason for the problem. Advised the customer to get some UPS equipment.

Previous   Quit   Next

Scaffolding

**Active Hypotheses**
Hard-drive problem
Frequent power outages

**Conducted Tests**

**Problem Solving Context**
Ok, complaint verified.
Probable Hard-drive or frequent power outages problem
Selecting hard-drive problem

**Troubleshooting Steps**
- View complaint
- Generate hypotheses
- Conduct tests
- Evaluate results

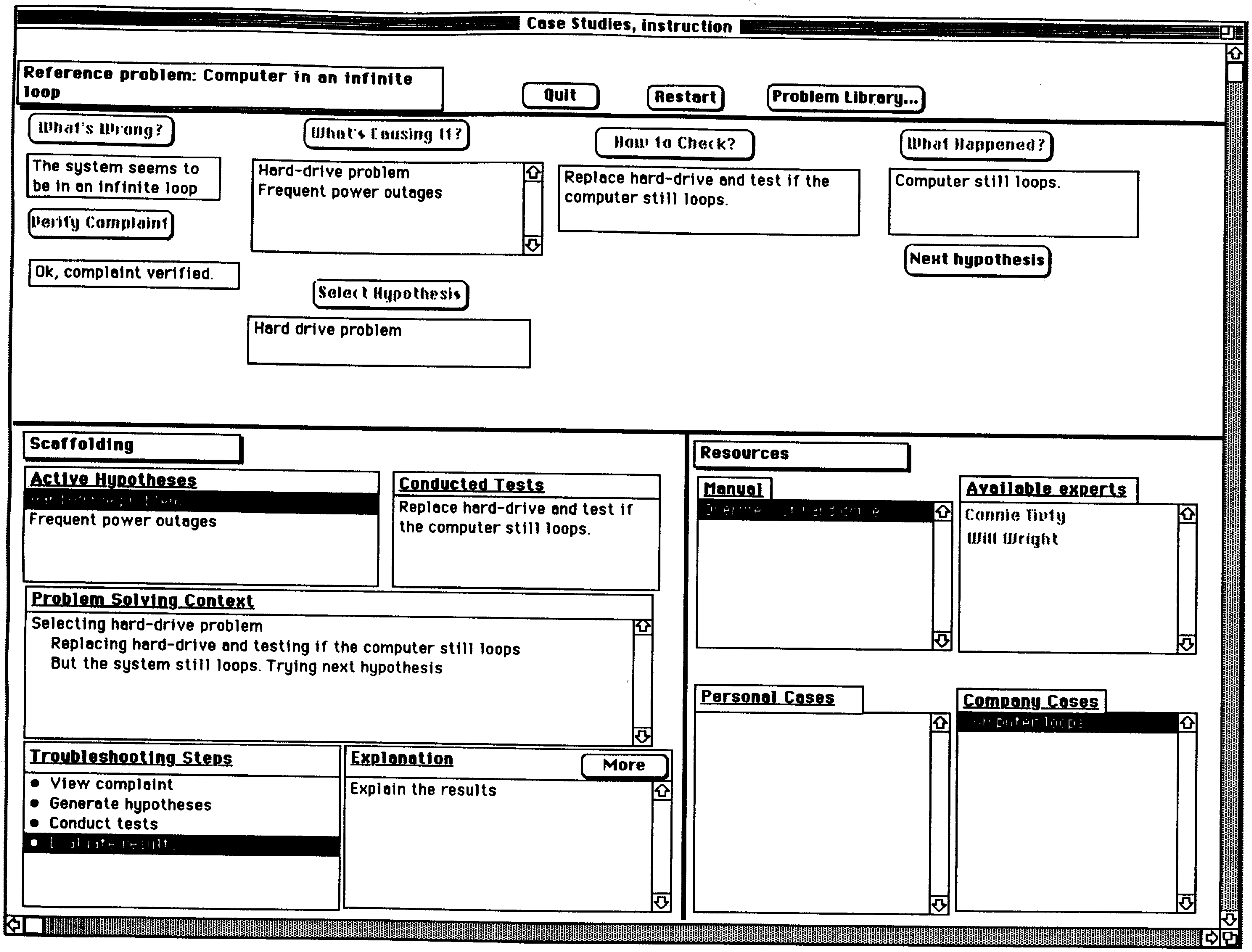
**Explanation**   More
When choosing a possible fault to test, select one that is easy to check

le experts
ifty
ght

**Personal Cases**

**Company Cases**
Computer loops

FIG 5



Case Studies, instruction

Reference problem: Computer in an infinite loop

The system seems to be in an infinite loop

Ok, complaint verified.

Hard-drive problem
Frequent power outages

Hard drive problem

Replace hard-drive and test if the computer still loops.

Computer still loops.

Scaffolding

**Active Hypotheses**

[illegible]
Frequent power outages

**Conducted Tests**

Replace hard-drive and test if the computer still loops.

**Problem Solving Context**

Selecting hard-drive problem
  Replacing hard-drive and testing if the computer still loops
  But the system still loops. Trying next hypothesis

**Troubleshooting Steps**

- View complaint
- Generate hypotheses
- Conduct tests
- Evaluate result

**Explanation**

Explain the results

Resources

**Manual**

Intermittent hard drive

**Available experts**

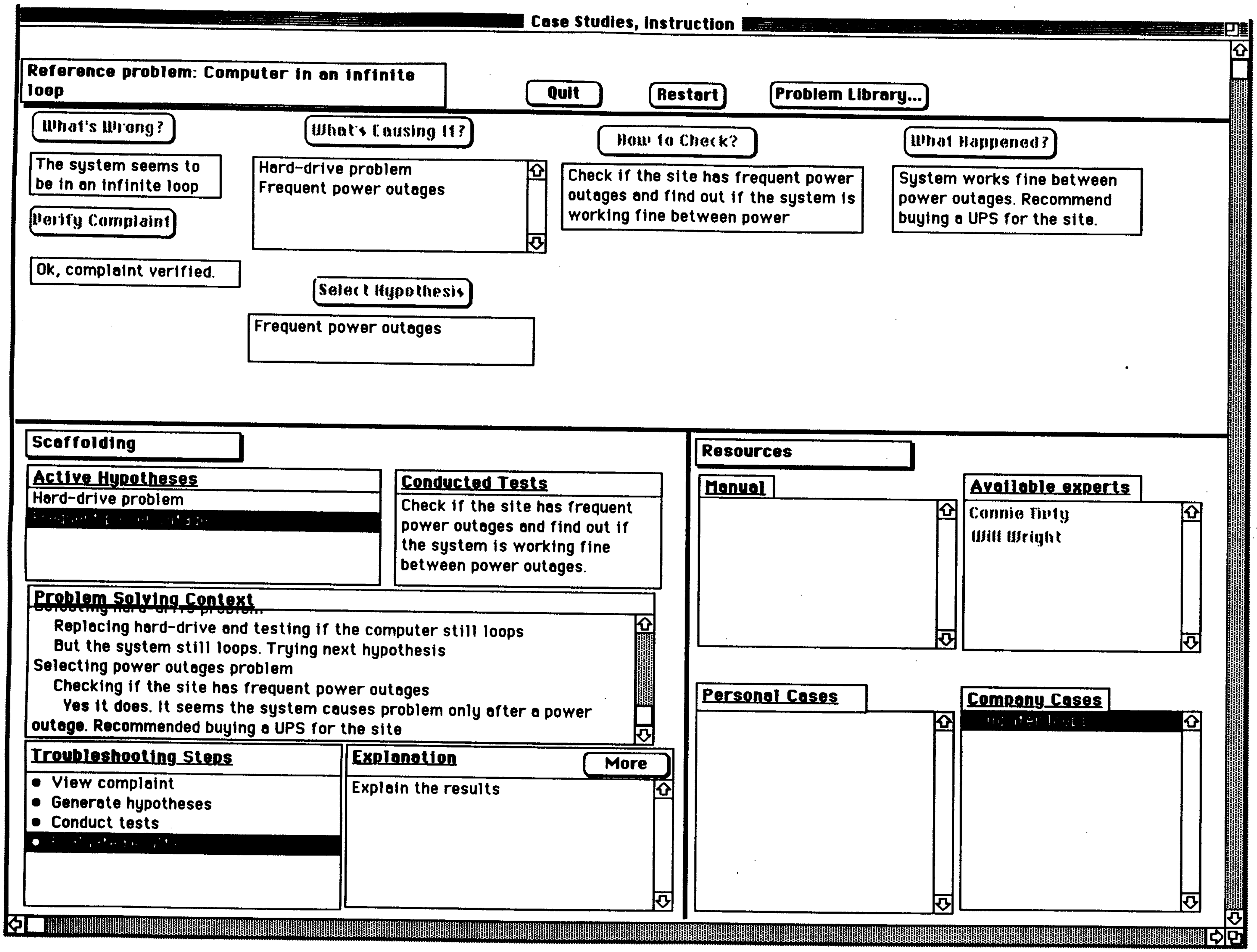
Connie Tirty
Will Wright

**Personal Cases**

**Company Cases**

Computer loops

FIG 6



Case Studies, Instruction

Reference problem: Computer in an infinite loop

The system seems to be in an infinite loop

Ok, complaint verified.

Hard-drive problem
Frequent power outages

Frequent power outages

Check if the site has frequent power outages and find out if the system is working fine between power

System works fine between power outages. Recommend buying a UPS for the site.

**Scaffolding**

**Active Hypotheses**
Hard-drive problem
[illegible]

**Conducted Tests**
Check if the site has frequent power outages and find out if the system is working fine between power outages.

**Problem Solving Context**
Replacing hard-drive and testing if the computer still loops
But the system still loops. Trying next hypothesis
Selecting power outages problem
Checking if the site has frequent power outages
Yes it does. It seems the system causes problem only after a power outage. Recommended buying a UPS for the site

**Troubleshooting Steps**
- View complaint
- Generate hypotheses
- Conduct tests
- [illegible]

**Explanation**

Explain the results

**Resources**

**Manual**

**Available experts**
Connie Tirty
Will Wright

**Personal Cases**

**Company Cases**
[illegible]

Fig 7

# Final Report
# Psychology Group
# ARI Contract
# December 1994

## OVERVIEW

The psychology group, Lawrence W. Barsalou in collaboration with Christopher Hale, Koen Lamberts, and Brian Ross, concentrated their efforts on two lines of work: mental models of physical systems, and the role of frames in mental models. The work on mental models addressed the acquisition of mental models for physical systems and the use of these models in troubleshooting. The work on frames addressed the nature of frames, empirical evidence for frames in human learning, and a computational model of frame acquisition.

## MENTAL MODELS OF PHYSICAL SYSTEMS

The primary emphasis of our work has been to explore factors that optimize the learning of symptom-fault rules in troubleshooting. For many kinds of devices, the large majority of faults can be encapsulated in a relatively small set of symptom-fault rules. Rather than having to use weak search methods or knowledge-intensive qualitative reasoning to discover faults, troubleshooters can learn symptom-fault rules for the majority of a system's failures and recognize these failures as they occur (see Christopher Hale's dissertation, Hale, 1992, for a comprehensive review of troubleshooting). Our primary hypothesis--to some extent confirmed and to some extent disconfirmed--has been that knowing explanatory knowledge of a system facilitates the acquisition of symptom-fault rules. The following three projects have examined this issue.

### The Role of Mental Models in Constraining Search during Troubleshooting

Our first project examined whether knowing a mental model for a physical system facilitated the subsequent acquisition of symptom-fault rules. Surprisingly, benefits of knowing a mental model where difficult to obtain. In several experiments, we obtained no effect of mental models at all. Instead, the overwhelming factor that determined how well subjects learned a symptom-fault rule was simply the number of times it was studied. The more that subjects studied a symptom-fault rule, the better they learned it. Knowing a mental model--on those occasions when it had a beneficial effect--was much less important to learning a symptom-fault rule than was frequency of study and use. The relative importance of study frequency and the relative unimportance of mental models suggests a simple, inexpensive strategy for training troubleshooters: Don't spend a lot of time and resources teaching mental models to troubleshooters, if the only type of troubleshooting that they will be performing involves symptom-fault rules. Instead, concentrate training efforts on repeated training of the rules so that they become well established in the troubleshooters' memories.

We hasten to add that mental models can be important under circumstances, as we shall see, and that the training of troubleshooting can therefore benefit, on some occasions, from the study of mental models. Our advice is simply that the importance of study frequency should not be ignored, given it's clear importance in learning.

One way in which mental models can benefit the application of symptom-fault rules is through constraining search during troubleshooting. In Barsalou and Hale (1995), we performed a series of experiments, which demonstrate that mental models can constrain search, such that the acquisition of symptom-fault rules is improved. To see how this works, imagines that 20 symptom-fault rules could potentially apply to the repair of a broken device. If all 20 rules must be considered for each broken device, regardless of the symptoms it presents, the troubleshooter must narrow search from 20 potential rules to the 1 correct rule. In contrast, imagine that the troubleshooter could immediately eliminate 16 of the rules as not being relevant, and only consider 4 in trying to converge on the 1 correct rule. In a signal detection context, where the correct rule is the signal and the incorrect rules are noise, it is much easier to detect the signal when the noise is 4 irrelevant rules than when it is 20.

In Barsalou and Hale (1995), we demonstrated that being able to constrain search in this manner, not only facilitates the guessing of rules, when the correct rule was not known, it also speeds the learning of rules, when rules are known. By developing mathematical models that partition guessing and learning, we were able to show that constraining search facilitates both guessing and learning.

To see how the experiments demonstrated this conclusion, imagine a physical system has 20 components, organized into 4 subsystems of 5 components each. In the unconstrained search condition, the symptom and fault always occurred in different subsystems. Thus, given a symptom, subjects could never predict which subsystem the fault was in. In this condition, subjects had to consider 19 potential symptom-fault rules, because they couldn't a priori rule out any subsystems. In contrast, in the constrained search condition, the fault always occurred in the same subsystem as the fault. Thus, subjects learned quickly that they could rule out possible faults in 3 of the 4 potential subsystems (i.e., 15 total faults) and concentrate search on the 4 remaining potential faults in the subsystem containing the symptom. Under these conditions, we observed that learning symptom-fault rules, not just guessing faults, proceeded significantly faster than when search was unconstrained.

These results demonstrate that helping troubleshooters constrain search can improve their ability to learn symptom-fault rules. Again, however, we emphasize the importance of study frequency: The importance of constraining search, although sizable in effect, is smaller than the benefit of increasing the number of study trials.

We obtained one other important set of results from these experiments. When subjects didn't know a fault (i.e., they hadn't learned the corresponding symptom-fault rule), they guessed. Mental models strongly biased subjects' guessing performance, with these biases taking two general forms: direction and distance. For direction, subjects tended to guess faults that lay behind symptoms in the flow of operation (although faults that lay ahead of symptoms were possible and occurred frequently). For distance, subjects tended to guess faults that were near the symptom as opposed to far from it. These biases were ubiquitous in our experiments and typically quite strong, prior to subjects learning the symptom-fault rules. Implications for training are that troubleshooters can be expected to have these biases, and that measures should be taken to anticipate the potential benefits and pitfalls of these biases.

Finally, we note that the learning of symptom-fault rules will not enable troubleshooters to discover new or particularly difficult faults, which may typically require deeper explanatory knowledge of the system or weak search methods. Again, these more 'exploratory' types of troubleshooting lie beyond the scope of symptom-fault rules. Again, the rationale for focusing on symptom-fault rules was our belief that they are used in the very large majority of troubleshooting cases, and are therefore critical to understand scientifically. See Hale (1992) for further discussion.

## The Nature of Human Explanations that Underlie Mental Models

In most of our research on mental models, we focused on performance measures of learning and troubleshooting, such as percent correct and reaction time. In this project, we focused on the explanations that people construct while learning mental models and using them in troubleshooting. This work is currently under review at the *Journal of the Learning Sciences* (Hale & Barsalou, 1994).

To explore the content and construction of human explanations, we performed detailed protocol analyses of the explanations that subjects produced while learning about a fictional system (the system learning phase) and later while troubleshooting it (the troubleshooting phase). During the system learning phase, subjects received a series of diagrams for a system and explained how it worked. During the troubleshooting phase, in the absence of the diagrams, subjects explained relations between problematic symptoms and their faults. To examine the evolution of explanations with learning, subjects explained the same system diagrams and the same symptom-fault relations multiple times.

The explanations that subjects constructed varied considerably between system learning and troubleshooting, indicating that different goals produced different explanations. During system learning, subjects focused almost exclusively on functional explanations, exhibiting no trend toward deeper mechanistic explanations over time. To construct these functional understandings, subjects primarily instantiated existing knowledge in memory, specializing it to explain the purposes of individual components. During troubleshooting, subjects still focused on function, but included a much wider variety of functional information in their explanations, as well as higher proportions of topographical and mechanistic information. Rather than focusing on individual components, subjects focused on qualitative relations between components, as they attempted to understand forward causal relationships from faults to symptoms. Also unlike system learning, explanations during troubleshooting evolved significantly over time, shifting from qualitative reasoning to reminding.

The topography and coherence of the diagrams studied in system learning affected subjects' explanations substantially: During system learning, explanations were more efficient and mechanistic when topography and coherence were present than when they were absent. During troubleshooting, these factors led to faster understanding of symptom-fault relations, as well as to higher rates of remindings, perhaps through improved indexing.

Our findings raise a variety of issues for further study. Perhaps most significantly, it is not clear how the instantiation process works during system learning. What is the nature of the existing knowledge that people bring to bear on instantiation? How does information in the current situation specify what relevant background knowledge in memory to utilize? How is relevant background knowledge configured and transformed to construct specific explanations? Does this process resemble the proof of explanation-based learning, or the schema-fitting of explanation patterns?

Second, what is the nature of the distributed understanding process that occurs during system learning? Although people did not talk much about relational information while understanding a system, they nevertheless appeared to use it extensively. In trying to explain the purposes of individual components, people appeared to rely heavily on their understandings of adjoining components. What information about adjoining components is critical to this process? How is information from adjoining components integrated with information about the focal component? What explanatory criteria are people trying to satisfy as they construct these distributed explanations?

Third, our findings raise additional issues that concern troubleshooting: On receiving symptom information, to what extent do troubleshooters rely on topographical relations, functional similarity, and so forth to select possible faults? Once people have identified candidate faults, what explanatory criteria do they use to evaluate the candidates and select one? What specific forms do functional and mechanistic reasoning take in this process, and what roles do they play? How do various types of knowledge acquired in system learning facilitate search and decision processes in troubleshooting?

Fourth, our data point to some interesting issues concerning the role of remindings in troubleshooting. What sorts of topographical indexes do people establish initially, and how do these indexes operate later during search? Also, how do good explanations enhance the use of topography in remindings? How does the quality of knowledge established during system learning affect the quality of remindings later during troubleshooting?

In summary, this project provides a rich source of preliminary information on human explanations about physical systems. However, it is important to realize that this project constitutes only one of many possible situations in which people construct explanations. Although our findings probably capture important general properties of explanations, it is obviously essential to explore explanations in other situations. Doing so will establish general properties with more certainty, as well as identify conditions that produce departures from them.

## The Role of Explanations in Learning Mental Models

Three general issues were of interest in this project, which became Christopher Hale's dissertation (Hale, 1992): First, do explanations provided to people while they are learning a system's structure facilitate the later learning of symptom-fault rules? Second, do explanations provided to people while they are learning symptom-fault rules (after they have learned the system's structure) facilitate learning? Third, are functional explanations maximally effective in facilitating learning, or does added causal information improve learning further?

In initial experiments, we observed little if any beneficial effect of explanations at any point in learning (in comparison to control groups who received no explanations). Consequently, we stopped performing these experiments and tried to understand why the explanations weren't working. To understand the hypothesis that we eventually developed, consider a few details of how these experiments work. During the initial learning of a system, subjects study diagrams of the system's structure, with each diagram representing a set of physical components connected by their input-output relations. For example, one of the diagrams for a satellite refueler might represent several chemical tanks, which are connected to a mixer that blends the chemicals to form fuel. In this diagram, subjects would learn that chemicals flow from the chemical tanks to the mixer, where they are mixed. Later, during symptom-fault learning, subjects might learn that when the mixer malfunctions, it is typically because a chemical tank is broken. Consequently, the symptom is a malfunctioning mixer, and the fault is a broken chemical tank. In the experiments, subjects receive "mixer malfunctioning" as the symptom, and "chemical tanks broken" as the fault. The way to think about this kind of troubleshooting is as a real world situation where the troubleshooter is dealing with a modular system, for which various monitoring and diagnosis systems isolate malfunctioning and broken components. For example, a monitoring system might initially warn the operator that the mixer is malfunctioning. The operator might then run various diagnosis procedures, which specify that a chemical tank is broken, causing the mixer to malfunction. At that point, a repair person might swap out the malfunctioning tank with a replacement.

Note that in the above malfunction and diagnosis, no information is provided about the behavior of the malfunction (i.e., how the mixer is malfunctioning), or about what is wrong with the chemical tanks. Instead, the operator simply receives information that the mixer is malfunctioning, and that a chemical tank is broken. Our hypothesis about why explanations weren't helping subjects learn symptom-fault rules was as follows: Perhaps subjects couldn't access the explanations from the symptoms, because the symptoms were too "barren," not containing any behavioral information that would be specific enough to retrieve the explanations. For example, imagine that the symptom is "malfunctioning mixer," and that the explanation for why the a broken chemical tank caused this malfunction is that "the mixer is malfunctioning because, a chemical tank is failing to send a necessary chemical, thereby causing the mixer to produce incorrect fuel mixture." Because the symptom ("malfunctioning mixer") fails to specify its behavioral problem ("producing incorrect fuel mixture"), the learner has difficulty accessing the explanation that contains this information. In contrast, when the symptom is elaborated as "mixer that's producing an incorrect fuel mixture," the match is higher, and the explanation is more likely to be retrieved, thereby providing access to the probable fault. To summarize, perhaps subjects only take advantage of explanations when the cueing information in the symptom activates the explanations stored with the fault in memory. In contrast, perhaps we weren't seeing explanation effects, either at learning or test, because, the symptoms weren't specific enough to retrieve the explanations.

To test this, we ran a few pilot subjects for whom we provided behavioral information about the malfunctioning component in the symptom, and we finally found explanation effects. As a result, we redesigned our original experiments to incorporate this critical variable, namely, whether or not the symptom is elaborated behaviorally or not. In these experiments, this factor is crossed fully with two additional factors: (1) whether subjects receive explanations at learning, and (2) whether subjects receive explanations at test. Note that explanations at learning describe how the system works as subjects study and memorize the diagrams for the system, whereas explanations at test describe how the symptom and fault are related as subjects receive feedback about the fault they generated for the symptom.

The implications of this finding appear important for designing troubleshooting applications in the real world. Essentially, it suggests that whatever symptom information subjects receive should be specific and should map closely onto the explanation that links it to the associated fault. For example, many modular systems nowadays simply instruct the operator that a particular component is malfunctioning, without specifying the malfunctioning behavior. If the operator knows an explanation linking the symptom to the fault, it is likely that this barren error message will fail to activate the explanation and thereby fail to take advantage of the explanation's ability to generate the fault. In contrast, if the error message states the behavioral properties of the symptom in a way that maps closely onto the explanation, the explanation is likely to become active and bring the fault to the operator's attention.

Thus, the primary finding from these experiments is that the largest explanation effect occurs for elaborated symptoms: When symptoms are barren, the presence or absence of explanations at either learning or test has little effect. Explanations primarily appear to facilitate learning when symptoms are elaborated. Moreover, there is a main effect of elaborated versus barren symptoms: Even when subjects receive no explanations, elaborated symptoms seem to promote better learning than barren symptoms. However, the best performance appears to occur when subjects receive both explanations and elaborated symptoms, suggesting that an explanatory understanding of the system, coupled with elaborated symptoms capable of accessing relevant parts of this explanatory structure, produce optimal troubleshooting performance.

Hale also ran one further experiment that assessed whether causal explanations produce any better learning than simple functional explanations. For example, a functional explanation might be, "the mixer is malfunctioning because, a chemical tank is failing to provide a necessary chemical, thereby causing the mixer to produce incorrect fuel mixture." In contrast, a causal explanation might be, "the chemical tank cracked and drained, such that its chemical did not flow to the mixer, thereby causing the mixer to produce incorrect fuel mixture." The latter explanation is causal, because it describes a physical mechanism (the crack) that caused the malfunction, rather than simply describing the functional problem as in the former explanation (i.e., a necessary chemical failed to arrive at the mixer). Of interest is whether the additional causal information facilitates learning, even though it doesn't increase the match between an elaborated symptom and the explanation. Our hypothesis was that the causal information strengthens the explanation, thereby increasing the chances of generating the fault, once the symptom accesses the explanation. The results weakly supported the causal hypothesis. However, it is critical to explore the role of causal information further before drawing strong conclusions. We doubt if the role of causal knowledge, if indeed important, plays as substantial a role in learning as does presentation frequency, constrained search, symptom elaboration, and the presence of explanations during system learning and troubleshooting.

Finally, it is essential to note that Chris Hale continues to carry out this line of work at Armstrong Laboratories (Brooks AFB). One of the primary accomplishments of this funding was the training and development of Hale as a research scientist. Not only did Hale contribute to much of the research in this report, he continues to contribute significantly to this area, and he will certainly contribute for many more years to come. For the past two years, Hale has been continuing the research in his dissertation, along with other related projects on the acquisition and troubleshooting of physical systems. One of Hale's primary projects has been to examine the long-term effects of explanations on the acquisition of symptom-fault rules. In all of our previous work, subjects were tested immediately on the application of these rules, thereby not enabling an assessment of what happens after longer delays. Of particular interest is whether explanations have larger effects than those observed thus far when there are long delays between rule acquisition and rule application, as often occurs in the real world. Hale's preliminary findings indeed suggest that explanations become increasingly important with delay. Because most real-world troubleshooting involves delay, the outcome of Hale's current studies has important implications in applied settings.

Once Hale has completed these studies, he plans to write a long report or monograph containing both these studies and those from his dissertation.

## FRAMES IN MENTAL MODELS

In performing our research on mental models, it became clear that mental models are a type of frame or schema. For this reason, we undertook a theoretical analysis to assess the nature of frames. Without such an analysis, we didn't believe that we could provide a clear and compelling account of the mental models that underlie the acquisition and troubleshooting of physical systems. Thus, we performed three projects in this area: one that attempted to establish the nature of frames, a second that attempted to acquire preliminary evidence for our particular formulation of frames, and a third that attempted to implement a simulation of this theory.

## The Nature of Frames

Several initial papers examined the structure of frames: Barsalou (1991), Barsalou (1992), and Barsalou and Hale (1993). Barsalou's (1992) text on cognitive psychology also develops the construct of frame as central theoretical entity and illustrates its role throughout the cognitive system. These accounts contrast frames with feature lists, the latter being a derivative of propositional logic, and the former a derivative of predicate calculus. Frames are much more powerful expressively than feature lists, representing several types of structure impossible to represent in feature lists. Most importantly, it is just this type of structure that is central to mental models.

In these papers, we established four critical relations that constitute frame structure. Our strong conjecture is that these four relations are necessary and jointly sufficient for the existence of any frame. These relations are: (1) attribute-value relations (i.e., slots bound to instantiations), (2) structural invariants (i.e., predicates that link attributes conceptually), (3) constraints (i.e., correlations between the values of two or more attributes, and (4) recursion (i.e., the potential decomposition of any attribute, value, structural invariant, or constraint into further frames). We know of no aspect of conceptual structure that cannot be accounted for by these four relations. Most importantly, it is exactly this sort of structure that is central to mental models of physical systems. Models of physical systems have attributes that take values (e.g., the current states of components), structural invariants (e.g., the connectivity between components), constraints (e.g., the dependence of one component's state on another component's state), and recursion (e.g., the hierarchical decomposition of a component).

In two later papers, we explored the thesis that frames have a perceptual basis (Barsalou, 1993; Barsalou, Yeh, Olseth, Luka, Mix, & Wu, 1993). These papers demonstrate how the four basic frame relations just described exist naturally and intrinsically in perceptual representations. Of future interest is exploring the roles that such perceptual representations play in the acquisition and troubleshooting of physical systems

## Empirical Evidence for Frames

Two preliminary projects attempted to obtain evidence for our account of frames. The first, performed in collaboration with Brian Ross and Koen Lamberts, examined the acquisition of frames for municipal water systems. Subjects studied a total of 12 water plants. One group of subjects--the frame condition--was asked to learn about water systems in general, whereas the other group--the exemplar condition--was asked to learn the individual water systems. The 12 water systems had a variety of attributes and structural invariants in common, while varying on the values of the attributes. Of interest was the degree to which subjects learned the frame structure, especially the attributes and their distributions of values. We predicted that the frame condition would learn the frame for the water systems, whereas the exemplar group would not. Results from the one experiment performed so far support this prediction. Model subjects were able to produce frame-structured information much more extensively than exemplar subjects. Interestingly, model subjects also appeared better able to remember information about exemplars, suggesting that acquiring the frame for water systems in general facilitated the learning of individual water systems. Further studies in this project are currently being planned.

The second project similarly sought evidence for frames. This project, performed in collaboration with Janellen Huttenlocher and Koen Lamberts, obtained compelling evidence that people integrate information across multiple presentations of the same individual into a frame for that individual. In these experiments, we demonstrated that people make an important decision, on encountering new information in the environment, about whether to create a new frame for storing the information, or to add the information to an existing frame. Essentially, this sort of processing conforms to the *one-entity one-frame-principle* (Barsalou et al., 1993). According to this principle, information about an individual extracted from multiple processing events is integrated into a single frame, assuming that the individual's identity is established correctly on each occasion. On encountering a familiar individual, its frame is retrieved to guide processing in two ways: First, the frame provides top-down guidance in interpreting the individual, utilizing information from previous experiences to understand and predict the individual's current behavior. Second, the frame absorbs information about the individual from the current processing episode, adding new information to the frame and strengthening repeated information, perhaps reinterpreting it in the process. The result is an integrated memory structure for the individual that contains both generic and episodic information. A paper reporting this work is currently under review (Barsalou, Huttenlocher, & Lamberts, 1994).

Thus, our preliminary attempts to obtain evidence for frames have been highly successful. We plan to continue gathering empirical evidence for frames, especially the role of frames in structuring the mental models that underlie physical systems.

## A Computational Model of Frames

During the funding period, we began building a general computational system--Image Frame Nets (IFN)--that reflects three themes: (1) The representation language should be inherently perceptual, not propositional. (2) Predicates, argument binding, and recursion should structure the system, enabling productive, symbolic computation. (3) The processing environment should be inherently statistical, much like a connectionist system, exhibiting generalization, pattern completion, and adaptive learning.

The IFN representation language is grounded in a general perceptual syntax that can apply to any modality of experience, including the five sensory modalities, proprioception, and introspection. Units in the system represent 'regions' of an experiential image that are attended to by selective attention. Connections between units represent attentional shifts from one region to another. Various relations between images, represented implicitly, underlie argument binding, predicates, and recursion (Barsalou, 1992). For example, an argument is a region of an image that becomes specialized with other images (e.g., the face on the schematic image for *person* is an argument, because images of many particular faces can specialize it). Similarly, recursion results from specializing the region of a region (e.g., specializing *eye* in *face* for *person*). Thus, specific patterns of attentional shifts between images and their regions implicitly define a productive syntax that can produce infinite symbolic structures from finite elements.

Based on its learning history, the IFN system adapts itself to its informational environment. In some ways, learning proceeds as in standard connectionism. For example, regions and attentional shifts develop strengths that reflect processing frequency. To the extent that some regions of an experiential image are processed more than others, the units that represent them become more responsive to input. Similarly, to the extent that some attentional shifts between regions occur more than others, the connections that represent them convey activation more rapidly. In other ways, however, the IFN system departs significantly from standard connectionism. For example, the system constantly grows new units as new images and regions are processed. Similarly, the system constantly grows new connections as new attentional shifts are performed. Unlike the standard connectionist system, which has a fixed set of units and a homogenous, fixed set of connections, our architecture is not 'closed' structurally. Thus, our system is fairly 'localist' in nature, although an entity's representation is always distributed over many units. Our system further differs from standard connectionism in how it controls activation. Rather than controlling activation through negative connections, the IFN system controls activation through negative baselines toward which units constantly decay. Thus, most connections are positive, reflecting a Hebbian sort of learning. Negative connections do arise when strategic attention deliberately attempts to suppress a unit, but this is far from the standard practice in which roughly half of a system's connections have negative weights. Finally, learning is much more local than in standard connectionism. Rather than distributing error across all connections, the IFN system only adapts those units and connections attended to explicitly, thereby avoiding catastrophic interference.

Initially, the IFN system is being developed as a general computational architecture. It will exhibit a broadly applicable form, as well as a straightforward, user-friendly interface that will provide considerable ease and flexibility to non-programmers. Currently, the representation language has been completely implemented, together with a classic Macintosh interface that allows users to construct a very large space of possible networks. Also completed are the mathematics that underlie activation and learning. Two final components of the system remain to be written: (1) The user interface for defining the exemplars that train the system during learning. (2) The tracking system for capturing aspects of the system's performance for later study.

Once finished, the IFN system could be used to represent and examine static nets, similar to the interactive activation nets in the word recognition literature. Alternatively, the IFN system could be used to build adaptive systems that learn during categorization and memory experiments. Once the IFN system reaches this stage of development, the next stage will be to build mechanisms that underlie problem solving and reasoning, so that we can model the types of results on system learning and troubleshooting reported earlier.

## REFERENCES

All of the works below were supported by ARI contract MDA 903-90-K-0012 and cite support from this funding source.

Barsalou, L.W. (1991). Deriving categories to achieve goals. In G.H. Bower (Ed.), *The psychology of learning and motivation: Advances in research and theory* (Vol. 27, pp. 1-64). San Diego, CA: Academic Press.

Barsalou, L.W. (1992). Frames, concepts, and conceptual fields. In E. Kittay & A. Lehrer (Eds.), *Frames, fields, and contrasts: New essays in semantic and lexical organization* (21-74). Hillsdale, NJ: Lawrence Erlbaum Associates.

Barsalou, L.W. (1992). *Cognitive psychology: An overview for cognitive scientists*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Barsalou, L.W. (1993). Structure, flexibility, and linguistic vagary in concepts: Manifestations of a compositional system of perceptual symbols. In A.C. Collins, S.E. Gathercole, & M.A. Conway (Eds.), *Theories of memory*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Barsalou, L.W., & Hale, C.R. (1992). Components of conceptual representation: From feature lists to recursive frames. In I. Van Mechelen, J. Hampton, R. Michalski, & P. Theuns (Eds.), *Categories and concepts: Theoretical views and inductive data analysis*. San Diego, CA: Academic Press.

Barsalou, L.W., & Hale, C.R. (1995). The role of mental models in guiding search during symptom-fault learning. Manuscript in preparation for submission to a journal.

Barsalou, L.W., Huttenlocher, J., & Lamberts, K. (1994). Processing individuals during categorization. Under review, *Cognitive Psychology*.

Barsalou, L.W., Yeh, W., Luka, B.J., Olseth, K.L., Mix, K.S., & Wu, L. (1993). Concepts and meaning. In In K. Beals, G. Cooke, D. Kathman, K.E. McCullough, S. Kita, & D. Testen (Eds.), *Chicago Linguistics Society 29: Papers from the parasession on conceptual representations* (pp. 23-61). University of Chicago: Chicago Linguistics Society.

Hale, C.R. (1992). *Effects of background knowledge on associative earning in causal domains*. Doctoral dissertation, Georgia Institute of Technology. Manuscript in preparation for submission to a journal.

Hale, C.R., & Barsalou, L.W. (1994). Explanation content and construction during system learning and troubleshooting. Under review, *Journal of the Learning Sciences*.