Center for Human-Machine Systems Research

School of Industrial and Systems Engineering, Georgia Institute of Technology
Atlanta, Georgia 30332-0205

Integration of Interactive Interfaces with Intelligent Tutoring Systems: An Implementation

Vijay Vasandani and T. Govindaraj

Technical Report CHMSR-93-2

September 1993

This research was supported by the Navy Manpower, Personnel, and Training R&D Program of the Office of the Chief of Naval Research under Contract N00014-87-K-0482.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1993 September	Technical

4. TITLE AND SUBTITLE
Integration of Interactive Interfaces with Intelligent Tutoring Systems: An Implementation

5. FUNDING NUMBERS
C: N00014-87-K-0482
PE: 0602233N
PR: RM33M20

6. AUTHOR(S)
Vijay Vasandani and T. Govindaraj

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Georgia Institute of Technology
Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
765 Ferst Drive, Atlanta, GA 30332-0205

8. PERFORMING ORGANIZATION REPORT NUMBER
CHMSR-93-2

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Cognitive Science Program (Code 1142CS)
800 North Quincy Street
Arlington, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Supported by the Office of the Chief of Naval Research Manpower, Personnel, and Training R&D Program.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Turbinia-Vyasa 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. The simulator, **Turbinia**, is based on a hierarchical representation of subsystems, components, and primitives together with necessary physical and logical linkages among them. **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. The interactive interface to the system and the tutor forms an integral and essential component of the instructional system. In this paper, we discuss the interface, including important features and implementation details. Our presentation includes details of the interactions.

14. SUBJECT TERMS
graphical interfaces; human-computer interaction; fault diagnosis; training; maintenance; intelligent tutoring systems; intelligent computer assisted instruction; interactive learning environments; marine power plants; simulation

15. NUMBER OF PAGES
40

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

Integration of Interactive Interfaces with Intelligent Tutoring Systems: An Implementation

Vijay Vasandani and T. Govindaraj

Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205, USA
+1 404 894 3873, tg@chmsr.gatech.edu
(Address all correspondence to TG.)

Abstract

Turbinia-Vyasa 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. The simulator, **Turbinia**, is based on a hierarchical representation of subsystems, components, and primitives together with necessary physical and logical linkages among them. **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. The interactive interface to the system and the tutor forms an integral and essential component of the instructional system. In this paper, we discuss the interface, including important features and implementation details. Our presentation includes details of the interactions.

Introduction

Design of interfaces to complex systems is *still* an art. This is rather unfortunate, because, as systems designers we would prefer a 'science' of interface design that sets forth clear principles and theories for the design of effective interfaces. However, in the absence of a science from which detailed guidelines evolve or can be derived, we must cope with the problem of design by adapting principles and techniques from related disciplines such as human factors engineering, human-computer interaction, and graphic arts. As in fine arts, the science or theories will evolve from practice, rather than theory driving practice (Albers 1975). Based on our extensive experience in interface design, and observation of interfaces to other complex systems, we believe that one of the best ways to develop effective and usable interfaces is by borrowing ideas from many domains and being sensitive to what has been done elsewhere. Such an evolutionary process can help to continually improve interfaces. We describe our research in this context, to illustrate the design features and provide a detailed discussion of the interface we have developed for diagnostic problem solving in a complex system.

Usefulness and effectiveness of interfaces in the operation of complex systems depends on how 'friendly' and accessible the system functions and capabilities are to the operator. Effective interfaces should be transparent and reveal the systems behind them, rather than being intrusive or ob-

trusive. This is especially true for complex systems whose dynamical behavior and system structure are often difficult to comprehend. Systems used by people with different levels of expertise and experience are especially in need of interfaces that are inviting, pleasant to use, and exhibit easy-to-use functionality. We have developed such an interface as an integral component of an intelligent tutoring system (ITS). In this paper, we provide the context, describe the interface, and discuss the important features.

System management via fault detection, diagnosis, and compensation is a key responsibility of human operators of highly automated systems such as aircrafts and steam power plants. While malfunctions seldom occur in such systems, when something does go wrong, prompt corrective response is crucial. During the diagnostic process, operators combine symptom information with mental resources concerning system knowledge and determine the course of action (Govindaraj 1988). Operators employ system knowledge at multiple levels of abstraction and detail for diagnosis (Rasmussen 1985). The efficiency with which operators diagnose problems is related to their level of expertise, which depends on their ability to organize, compile, and integrate appropriate pieces of operational, system state information with system knowledge. Operators acquire expertise via training on actual systems and on simulators. Properly designed ITSs integrated with simulators or actual systems can be very effective in imparting training. We have developed an ITS architecture and implemented it for the oil-fired steam power plant domain. The interactive interface of the system is an important component of the ITS.

Intelligent tutoring systems have traditionally been concerned with modeling the knowledge requirements. Developing interfaces that provide transparent and flexible access to the system has received insufficient attention. Well-designed interfaces can assist in visualizing the system from different perspectives and investigate the components/systems promptly. Such interfaces would be extremely useful during system operation, diagnosis, and maintenance. In this paper, we concentrate on such an interface to an ITS. The interface attempts to offer help in visualizing the system structure, function, behavior, and information to aid diagnosis so that student operators can acquire diagnostic problem solving skills efficiently.

Diagnostic skills are normally acquired via training on-the-job or on simulators (e.g., Johnson 1988; Kearsley 1987; Goldstein 1986), after the operators have learned about the operation of the system. On-the-job training is usually very expensive and the consequences of an error can be catastrophic. Malfunctions occur infrequently and it may be undesirable or impossible to duplicate them during training. A simulator coupled with an intelligent computer-based tutor can provide effective training based on an evaluation of a student's misconceptions from observed actions. Such a combination of simulator and tutor constitutes an intelligent tutoring system.

Since highly automated systems operate reliably and problems seldom occur, operators do not have sufficient opportunity to observe problems in operation. This, combined with the large scale nature of the problem, makes it very difficult for operators to form mental models of the system dynamically when problems do occur. Operators may not be able to selectively combine information to

locate and identify problems.

System complexity often leads to difficulty in forming associations with components and subsystems dynamically, which is essential for diagnostic problem solving. Operators must be able to form approximate system representations or mental models based on the symptoms observed during troubleshooting to evaluate possible responses to hypothesized failures and faults. Training programs that offer help in forming and using appropriate mental models would be helpful. An effective ITS must facilitate the development of accurate mental models of the system so that an operator can integrate information from various subsystems and components in a timely fashion. Interactive interfaces, properly integrated with the ITS, can facilitate this process.

In the next section, we provide a brief review of the general architecture of intelligent instructional systems for diagnostic problem solving. In the section that follows, we describe a methodology for organizing knowledge so that it can be implemented in an instructional system. The student-tutor interface, which forms the core of this paper, is described next. An experimental evaluation of the training system and the results follow. Finally, we conclude with a summary of the results and some observations.

Intelligent tutoring in complex systems

Research in intelligent tutoring and training systems has traditionally focused on relatively simple tasks concerned with imparting basic skills in mathematics, electricity, physics and computer programming (Wenger 1987; Sleeman & Brown 1982). These domains lack the complex interactions between subsystems that are characteristic of most engineering domains. Due to the inability to represent complexity, most ITS design principles have not been successfully extended from simpler, less constrained domains to complex engineering systems (Burns, Parlett & Redfield 1991; Frasson & Gauthier 1990; Psotka, Massey & Mutter 1988). Suitable methodologies for representing the system complexity are generally lacking.

Only a small fraction of the research on ITS deals with engineering domains. SOPHIE, designed to teach troubleshooting in electrical circuits, was perhaps one of the first ITSs in an engineering domain (Brown, Burton & de Kleer 1982). Since SOPHIE, a great deal of progress has been made. The SHERLOCK family of tutors (Lesgold et al.1991), developed for a complex electronic troubleshooting job, has a far richer representation of the work environment than SOPHIE. Intelligent Maintenance Training System (IMTS) and its successors provide interactive environments for constructing domain-specific simulations and training scenarios (Towne & Munro 1988). Finally, in the domain of marine steam power plant, we have developed ITSs (Fath, Mitchell & Govindaraj 1990; Vasandani 1991; Vasandani & Govindaraj 1990, 1991a & 1991b).

In general, all ITSs have a similar architecture. The basic ITS architecture is comprised of an expert module, a student module and an instructional module. In addition, a simulator provides the training environment. The expert module contains the domain expertise which is also the knowledge to be taught to the student. The student module contains a model of the student's current level of com-

petence. The instructional module is designed to sequence instructions and tasks based on the information provided by the expert and student models. Also, the interface used to communicate knowledge to the student can be treated as a separate component of these systems. Together with the simulator and an interactive interface, the three components of the tutor (i.e., the expert, student, and instructional modules) comprise the architecture for the instructional system.

One of the major problems in the design and implementation of ITSs in the past has been the lack of a suitable methodology to represent knowledge. We have developed a methodology that addresses the representation problem by decomposing, organizing and representing domain knowledge of complex dynamic systems for building functional, computer-based intelligent tutors. In our ITS architecture, domain knowledge is separated from pedagogical knowledge. Domain knowledge is decomposed into system and task knowledge. System knowledge is organized using a structure-function-behavior model of the system and its components. Task knowledge is organized in a manner that facilitates evaluation of student misconceptions. Pedagogical knowledge is decomposed into knowledge to plan and execute the pedagogical functions of the tutor that includes evaluation and rectification of misconceptions. A blackboard-like control architecture is used to manage the components of knowledge and to deliver instructions. A brief outline of the knowledge organization and representation follows.

Organization of system and operational knowledge

Operators of complex dynamic systems must be familiar with operational principles of different types of system, e.g., thermodynamics and heat transfer for the fuel system, or electrical characteristics for a turbogenerator. In addition, the operator must know the nominal values of the state variables and parameters. Problem solving and compensation for failures require processing of information from various subsystems using efficient troubleshooting strategies. Therefore, an ITS must be capable of organizing and presenting knowledge about the system and the troubleshooting task at several levels of granularity or detail.

The volume of knowledge to be represented is often overwhelming. Furthermore, this knowledge is interrelated and tightly coupled, and therefore cannot be stored as isolated modules. Knowledge must be integrated into proper contexts that can be easily recognized and comprehended, to assist in the formation of appropriate mental models. A framework that can help integrate the large volume of knowledge associated with the safe operation of real-world systems is needed. We outline a framework here, followed by a detailed discussion of the interface.

Four components of knowledge are necessary in an intelligent tutor for diagnostic problem solving in complex dynamic domains. These components are: (1) a large amount of system knowledge organized to facilitate evolution of system states with time, (2) troubleshooting task knowledge, including knowledge about failures and student actions, (3) knowledge to infer a student's possible misconceptions from observed actions, and (4) pedagogical knowledge to realize the tutoring objectives.

Knowledge about the system and the troubleshooting strategies constitute an expert model of the operator's task. The knowledge must be organized in a manner that is easily accessible and communicable to the student. The instructional module uses this model to train students to use proper diagnostic problem solving strategies. Knowledge of student's actions can help the instructional module to infer possible misconceptions. Finally, knowledge of tutoring goals and how they are to be realized guides the instruction and its communication. Figure 1 summarizes the components of knowledge. The knowledge is implemented in an ITS, together with an interactive interface.

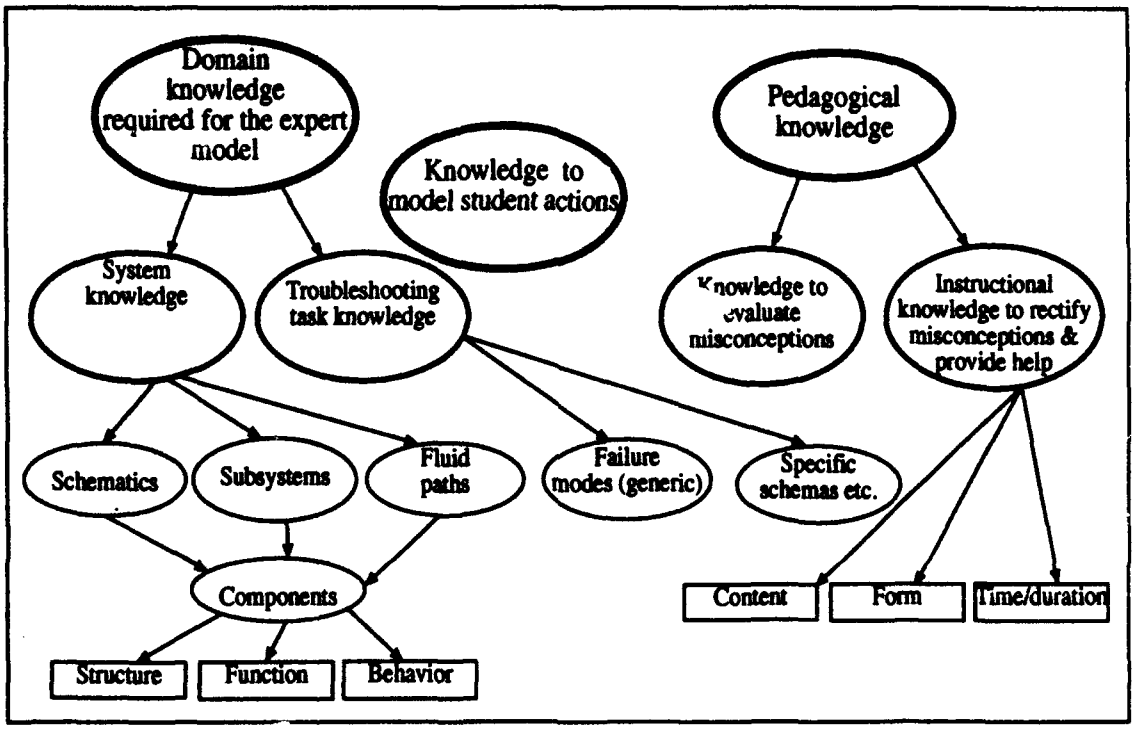


Figure 1. Summary of Knowledge Components

A blackboard-like control architecture coordinates the modules that contain various components of knowledge sources and performs high level control and planning of pedagogical functions. Knowledge organization, including the blackboard control architecture, is described in detail in (Vasandani & Govindaraj 1993 (in press); and 1993a). Complete details of the implementation and a description of the student's interaction with the ITS can be found in (Vasandani 1991).

Knowledge organization captures system structure, function, and behavior under failed and normal states, pedagogical knowledge that concerns evaluation and rectification of misconceptions, instructional knowledge that concerns content, form and time of presentation of instructions, and a control architecture for planning the pedagogical functions of the tutor. However this knowledge alone is insufficient in the absence of properly designed interactive interfaces. We describe the interface and details of interaction between the student and the instructional system next.

Interfaces for student-tutor interaction

General background

Interfaces must communicate appropriately organized knowledge to operators during training and system operation. In an ITS, a good interface can make the knowledge of the tutor transparent to the student and help the student understand the complex structure, function, and behavior of the controlled system. In addition, a well designed interface addresses the external-internal task mapping problem (Moran 1983) and establishes a semantic link between the actions relevant to the task in the domain and the actions to be taken at the interface (Miller 1988).

Traditionally, interfaces in tutoring systems have often been designed as an afterthought. SHERLOCK (Lesgold et al.1991) and IMTS and its successors (Towne & Munro 1990) are exceptions to this trend. In fact, our system has a number of features in common with interfaces to these systems. Our approach is to design the interface as an integral component of the tutoring system. Transparent interfaces should provide direct access to system components. Components of the system should appear as they would in a real system. In addition, the tutorial interface should facilitate the normal performance of an operator's tasks.

The task of the operator, for which the student is being trained, is that of identifying the cause of the problem from the symptoms observed. Typically, an operator in supervisory control sets the system's operating conditions and continuously monitors the system state. When a problem or a malfunction occurs, she formulates hypotheses about what may have caused the observed symptoms, performs a number of tests to refine, modify, and eliminate hypotheses, and finally identifies the plausible cause. The diagnosis may be verified by further tests. The tutoring system must support this process.

In a diagnostic problem solving task, a set of schematics often serve as a convenient interface for communication of knowledge. These schematics are designed to minimize the external-internal task mapping problem by having the valves and gauges that are usually under the control of the student operator appear as manipulable objects. Other factors such as grouping of components and graphics also influence the design of these schematics.

Grouping of components into schematics depends upon the degree of logical proximity between components and subsystems; the extent of diagnostic actions necessary to investigate frequent failures; and layouts that ensure smooth transition between schematics. For example, a high degree of interaction between the steam generation and combustion subsystem of a power plant requires that the two subsystems be displayed on the same schematic. Similarly, logically proximate components such as the stack and the burner in a combustion unit must appear together in a schematic. Also, the connections that are discontinued on the left edge of one schematic must continue from the right edge of the connected schematic to maintain 'visual momentum' (Woods 1984).

Graphics and icons in a schematic interface can enhance the performance of instructional systems (Hollan, Hutchins & Weitzman 1984; Towne & Munro 1990; Lesgold et al.1991). Graphical ob-

jects or icons can be effectively used to represent meaningful objects or concepts of the system. For instance, in engineering, it is customary to represent a turbine as a trapezoid with the smaller cross section representing the inlet to the turbine. The trapezoidal shape also reminds the viewer that the steam expands in the turbine as it moves from a smaller cross section inlet to a larger cross section outlet. Similarly, concepts such as blocked-shut valve and functional subsystems of a power plant can also be represented by meaningful icons.

Schematics provide an interface between the student operator and the simulated system as well as between the student and the tutor. The tutor uses the schematics to highlight components that constitute a subsystem or share a common fluid path. There is a three-way interaction between operator, system, and the tutor. Such graphical techniques promote visualization of functional subsystems and their interaction. Schematics can also be used by the tutor to animate fault propagation by highlighting gauges as they turn abnormal under simulated failure conditions.

While schematics along with the simulation provide a practice environment that emulates the real system, they do not cover all aspects of student-tutor interaction. For example, the students require a set of expressive techniques to state their hypotheses. The tutor, apart from observing actions, needs a method of seeking information to evaluate misconceptions. Thus, the tutor interface design also involves developing student- and tutor-initiated channels of communication. Since the ability of the instructional system to answer questions is limited by its knowledge and the way this knowledge is organized, the interface must be designed to control and guide the interaction between the student and the tutor.

In *student-initiated communications*, the interface must assume the responsibility of guiding the user into asking the right type of questions. For instance, when the student seeks information concerning the system's structure, function, or behavior, it is helpful to make the student select appropriate, context-relevant queries from a set of menus. Such menus, when organized hierarchically, can also reflect the inherent hierarchical structure of the complex system and promote a better understanding of the system (Miller 1985). Furthermore, an interface that has the provision to address identical queries via multiple representations of the system helps consolidate knowledge from multiple perspectives.

For *communications initiated by the tutor*, the interface must help the student to understand and correctly respond to the queries. Where a student can respond to a query in multiple ways, the student options must be recognized in advance and the choice restricted to known alternatives. For example, when the student is asked to provide hypotheses concerning the most likely mode of failure, it makes sense to confine the student's response to only those modes of failure that are known to the tutor. Therefore, making the student select from viable alternatives instead of permitting unguided response is a better approach to interface design.

In a typical session, the student is shown the ambient conditions and is presented with the symptoms observed during (abnormal) system operation. She must identify the component responsible for the observed symptoms. Typically, this is done by investigating a number of components and

their associated gauges. The tutorial interface assists the student in interacting with the system.

Design features and basic principles

A small set of principles guide the design of the interfaces. These principles are listed below, followed by details of implementation.

Simulated system should be similar to its real world counterpart.

Access should be transparent.

Maximize the 'information content' in a limited space.

Reduce memory load as much as possible; use recognition rather than recall and show possible options that are relevant for the system state.

Assist by off-loading information and providing external memory; keep track of what has been done and where you have been (navigation).

Use familiar and standard shapes and symbols where possible; develop icons and symbols that are easy to learn, recognize, and interpret.

Provide help to formulate, develop, refine, update, and where necessary eliminate hypotheses concerning problems and failures. Make this process gradual and incremental.

Offer detailed explanation for the evolution of the system behavior and observed symptoms (cause-effect associations to assist in forming patterns etc.).

Sequence instructions and help as it would occur in a real system for a real task.

Reinforce knowledge and build up on existing knowledge (scaffolding).

Concentrate on the diagnostic component by emphasizing on and providing deviations from nominal values, rather than actual values that need further processing.

Provide for redundancy where possible, i.e., multiple forms and/or modalities of displays and input options must be provided to accommodate individual styles and preferences, subject to the space and other relevant constraints.

Implementation particulars

The ITS implementation consists of a domain simulator, **Turbinia**[1], and a computer-based tutor, **Vyasa**[2]. Together, **Turbinia** and **Vyasa** constitute an instructional system that trains operators to troubleshoot oil-fired steam-driven marine power plants. **Turbinia-Vyasa** is implemented in Macintosh Common Lisp with Common Lisp Object System and runs on Apple Macintosh II computers. **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 intelligent tutor that trains operators to troubleshoot **Turbinia**. **Vyasa** operates in two modes: passive and active. In the *passive* mode the student is solely responsible for initiating the communications. When help is needed, the student must interrupt the simulation. She can then

1. Turbines were first applied to marine propulsion by Sir Charles Parsons in 1897. *Turbinia*, an experimental vessel of 100 tons, was fitted with turbines of 2,100 hp driving three propeller shafts. It attained the then record speed of 34.5 knots (A. F. Burstall, 1965, *A history of mechanical engineering*, MIT Press, Cambridge, MA, p.340).

2. Ancient Indian sage, scholar and teacher.

browse through and obtain about detailed knowledge of the system and about generic failures. When the passive tutor is invoked, the simulation is temporarily brought to a halt and the student can access various segments of knowledge in the expert module. In the *active* mode, the tutor takes the initiative to provide instructions when it infers a possible misconception based on the student's actions. In this mode, the tutor monitors the student's actions and offers help when appropriate. The instructions may be provided by the active tutor with or without intervention. The capabilities of active tutor include all the capabilities of the passive tutor as well.

The two manifestations of the tutor, passive and active, offer help when the student wants to investigate the system and when it seems appropriate to intervene. The passive component supports the exploratory phase where the student takes the initiative. The active component attempts to provide 'intelligent' help that lessens the chances for getting stuck.

We have provided the background to tutoring for diagnostic problem solving and an outline of the context. The student-tutor interface of **Turbinia-Vyasa** and the valid forms of interactions in both the passive and active modes are described next.

Interaction with Turbinia-Vyasa

The student-tutor interface of **Turbinia-Vyasa** has been developed on a dual screen Apple Macintosh II workstation. The configuration consists of a 19" color monitor on the left and a 13" color monitor on the right (Figure 2). A single button computer mouse that can point to all locations on both screens is used for input. The mouse is used to interact with the direct manipulation interface to both **Turbinia** and **Vyasa**. All actions at the interface involve moving the mouse cursor to a desired location and clicking *once* on the mouse button. All valid user actions have appropriate response while invalid actions are ignored by the system.

The joint interface to **Turbinia-Vyasa** consists of an interface to the simulator **Turbinia**, and dialogs to interact with **Vyasa**. **Turbinia's** interface consists of seven schematic windows, a schematic menu, a requests menu, a symptom dialog, several error dialogs and a clock. Student interaction with **Vyasa** is accomplished by multiple levels of hierarchically organized passive tutor help dialogs and a hypothesis menu.

At the beginning of every training session, the large screen displays the *schematic menu*, the *requests menu*, the *hypothesis menu* and the clock. A *tutor dialog* is displayed on the bottom edge of the small screen. For sessions where the active mode of the tutor is not invoked, the *hypothesis menu* under the *requests menu* is not displayed.

A student interacts with **Turbinia** through seven schematics. (An example schematic is shown in Figure 3.) These schematics can be accessed by clicking on the icons in the schematic menu described below. When the student clicks on a displayed gauge to probe its reading, an icon appears near the gauge. This icon is a qualitative representation of the current gauge reading, which is either low, slightly low, normal, slightly high or high.

When **Vyasa** operates in the *passive* mode, the student is responsible for initiating communica-

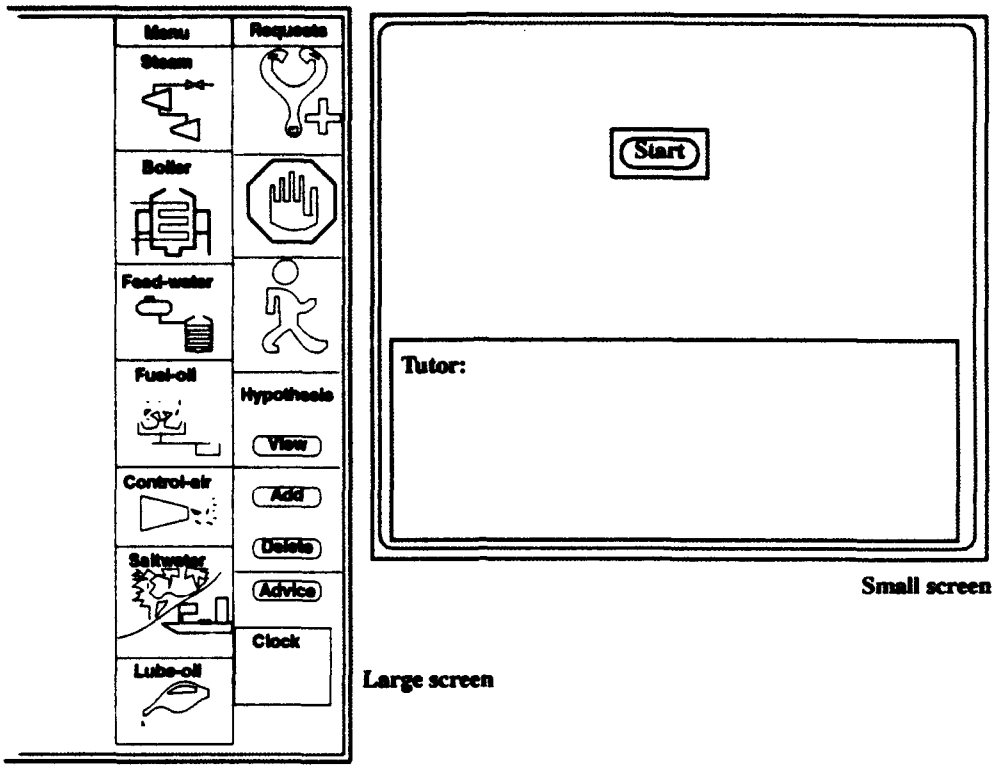


Figure 2. Configuration of screens showing menus and the tutor window

tions with the tutor to learn about the system and the failures. Student-initiated interaction with the tutor is accomplished by clicking on the stop icon in the requests menu. This action halts the simulation temporarily, enabling the student to interact with the tutor while preserving the information concerning system states.

The student uses the *passive tutor help dialogs* to communicate with **Vyasa** and seek information concerning the structure, function and behavior of the subsystems and the components. Students can use these dialogs to explore the tutor's knowledge-base.

In the *active* mode, **Vyasa** often intervenes to communicate with the student. It does this through instructions presented on the tutor dialog, accompanied by a beep. These instructions are delivered following the evaluation of a student's misconception. For instance, if the student investigates schematics, subsystems, or fluid paths unaffected by the failure, the tutor delivers the appropriate instructions to guide the student away from unaffected portions of the power plant.

Schematic menu

The schematic menu on the large screen, shown in the left half of Figure 4, displays seven icons. Each icon represents one of the seven schematics of the simulated power plant. These icons are used to display the schematics on the left screen. The *schematics* display the physical connections between the components of the power plant. They are used to investigate components and probe

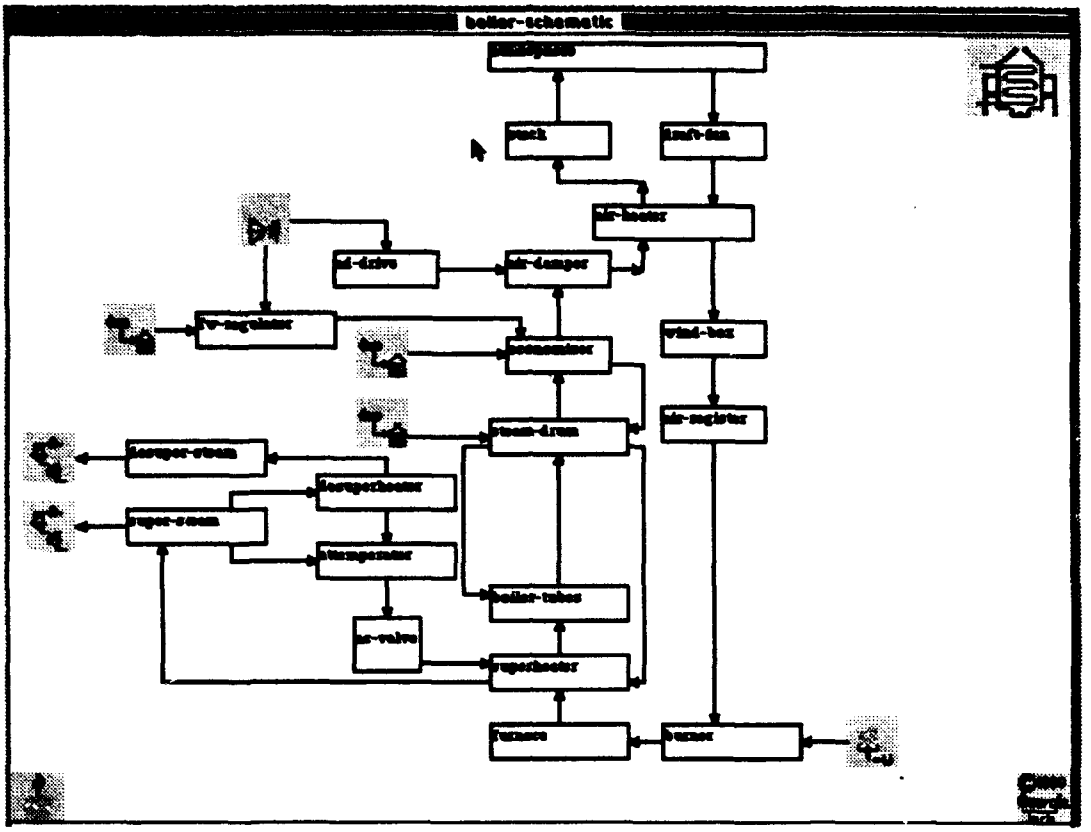


Figure 3. The Boiler Schematic

gauges attached to these components. The names of the schematics are also provided in textual form above each icon. The schematics are the steam, boiler, feed water, fuel oil, control air, salt-water and lube oil. A schematic can be accessed by the student by clicking on the icon representing the schematic.

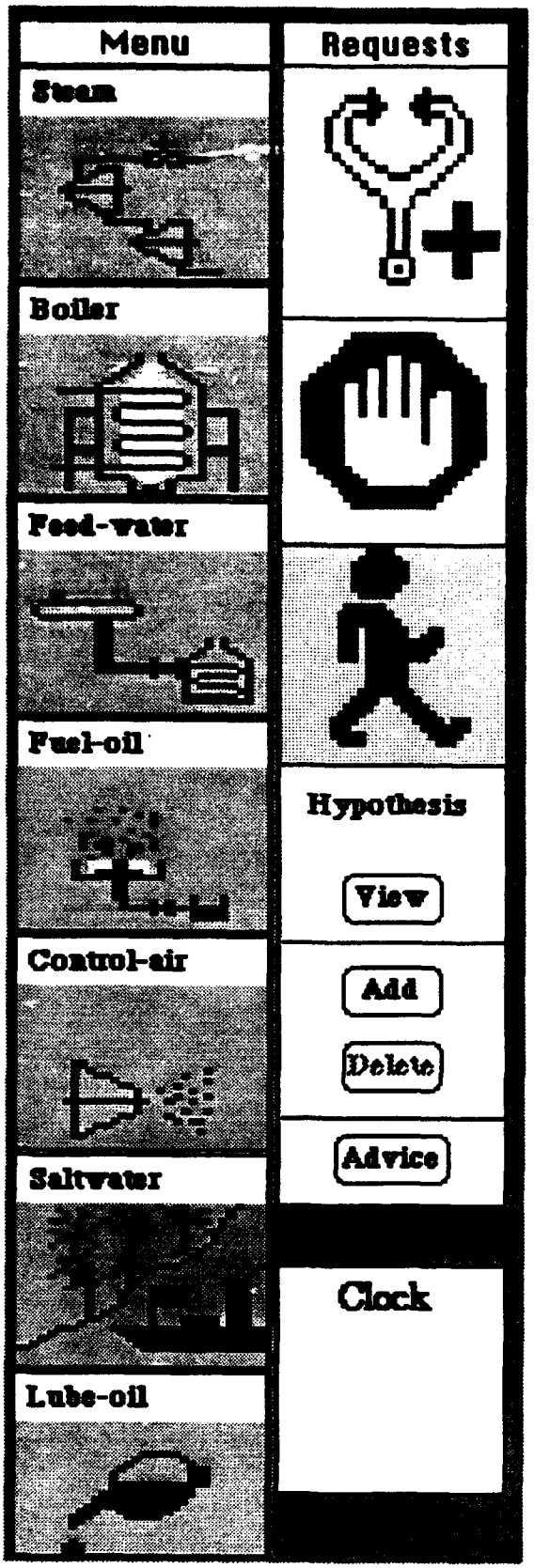
Requests menu

The requests menu to the right of the schematic menu displays three icons (Figure 4). The first icon is used to request for an opportunity to diagnose the fault, the second to temporarily halt the simulation and the third to resume the simulation. When working without **Vyasa**, the stop and resume icons are disabled.

The student must click on the first, the diagnose icon, before identifying the failure. By clicking on the diagnose icon, the student indicates to the instructional system an intention to identify the component responsible for the observed abnormal behavior.

A click on the second, the stop icon, halts the simulation and puts the student in a mode to interact with the passive tutor. In a session without the tutor, a click on the stop icon produces an error message.

The third, the resume icon, is used to restart simulation after it has been halted to communicate with



The *stethoscope (diagnose)* icon is used by the student to indicate the component that may have caused the fault.

The *stop* icon is used to halt the simulation during the tutor modes.

The person walking icon is used to *resume* the simulation.

The buttons in the *hypothesis menu* are used to view the current hypotheses, add or delete to this list, or to request advice from the tutor.

The *clock* displays the remaining time.

Figure 4. The main menus

the passive tutor. In a session without the tutor or when the student is not interacting with the tutor, the resume icon is shown disabled. All disabled icons on the **Turbinia-Vyasa** interface have a yellow-brown colored background as compared to enabled icons that are shown in gray.

Hypotheses menu

Hypotheses menu appears below the requests menu (Figure 4). It has four buttons. Students use this hypothesis menu to communicate information concerning their failure hypotheses to **Vyasa**. The "View" button is used to view the list of hypotheses provided by the student to the tutor. The "Add" button is used when the student wants to specify a new hypothesis. The "Delete" button is used to remove a hypothesis from a list of hypotheses. Hence the delete button gets highlighted only after the student has provided a set of hypotheses. Finally, the "Advice" button is used to seek help concerning a particular hypothesis. Detailed interaction with this menu is described later.

Clock

The time to diagnose a fault is limited. During a session, the clock below the hypothesis menu displays the time left to solve the current problem. This clock is updated every minute. When the clock winds down to 0 and the fault is not yet diagnosed, the simulation stops and the student can no longer continue the diagnosis.

Tutor dialog

The instructional system communicates with the student through textual messages and instructions presented on the tutor dialog. This tutor dialog is displayed on the bottom edge of the small monitor (see Figure 2). All communications through this dialog box are accompanied by a beep.

Symptom dialog

The *symptom dialog* shows the initial symptoms observed at the beginning of the troubleshooting task. The *error dialogs* convey appropriate messages when errors are made. The *tutor dialog* is used by the instructional system to communicate with the student. The display of *symptom dialog*, *error dialogs*, or new text on the *tutor dialog* is accompanied by a beep.

The symptom dialog shows the simulated ship's initial operating condition and the first symptoms that indicate the existence of a problem (Figure 5). Troubleshooting for failure begins after the symptom dialog appears with a beep in the center of the large screen.

The rest of this section is divided into two parts. The first part describes the interaction with **Turbinia** and the second describes the interaction with **Vyasa**. Most of the examples used to describe the interaction are related to a problem solving session that begins with the symptom dialog shown in Figure 5.

Interaction with Turbinia

The student interacts with **Turbinia** through the seven schematics. The student accesses the schematics by clicking on the icons in the schematic menu. The same schematic icon that is used to access the schematic also appears in the top right corner of the schematic it represents, to provide

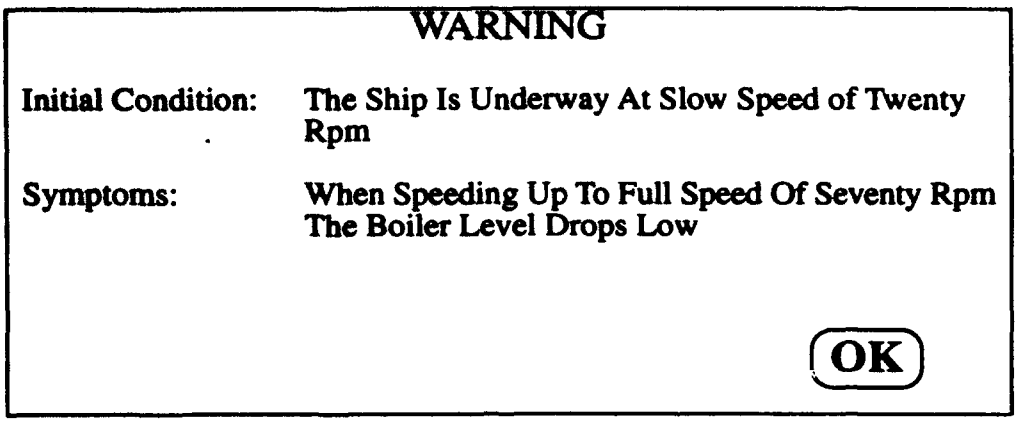


Figure 5. Symptom Dialog

feedback that it was indeed the one that was chosen and for easy visual reference. We discuss the features of the schematic interface of **Turbinia** with reference to boiler schematic shown in Figure 3.

All the components in the schematics are represented by rectangles. (Even though it would have been preferable to use standard symbols (e.g., trapezoids for compressors and turbines), we used rectangles for ease of implementation. Our intent was to replace them later by appropriate icons and symbols. However, pragmatic considerations prevented us from doing this at a later time.) The connections between components are shown by solid lines called connectors. The direction of fluid flow between components is shown by the arrow head on these connectors. For example, the economizer and the steam drum (see Figure 3) have a two way connection. The connection from the economizer to the drum represents the flow of feed water while the connection in the reverse direction represents the flow of flue gases.

Some connectors, like the one connecting the feed water icon to the feed water regulator (represented as f-w-regulator in the boiler schematic), have a component on one end and an icon at the other. Such connectors represent connections between components that are in different schematics. The icon at one end of such connectors represents the schematic in which the connected component can be viewed. In this example, the input connector to the feed water regulator physically originates from the hp heater in the feed water schematic.

If the student clicks on the feed water icon at the end of the input connector of feed water regulator two things happen. First, the display switches to feed water schematic. Second, the boiler icon on the output connector from the hp heater is highlighted with a red band around it. (This provides one-step memory for navigation by highlighting the icon corresponding to the schematic from which we arrived at the current schematic.) The highlighted boiler icon helps establish the physical connection between the hp heater and the feed water regulator. The student can click on the highlighted boiler icon to get back to the boiler schematic. When this is done, the boiler schematic has two feed water icons highlighted, one connected to the feed water regulator and the other to the economizer (see Figure 6). This simply means that the hp heater is connected to both the feed water

regulator and the economizer in the boiler schematic.

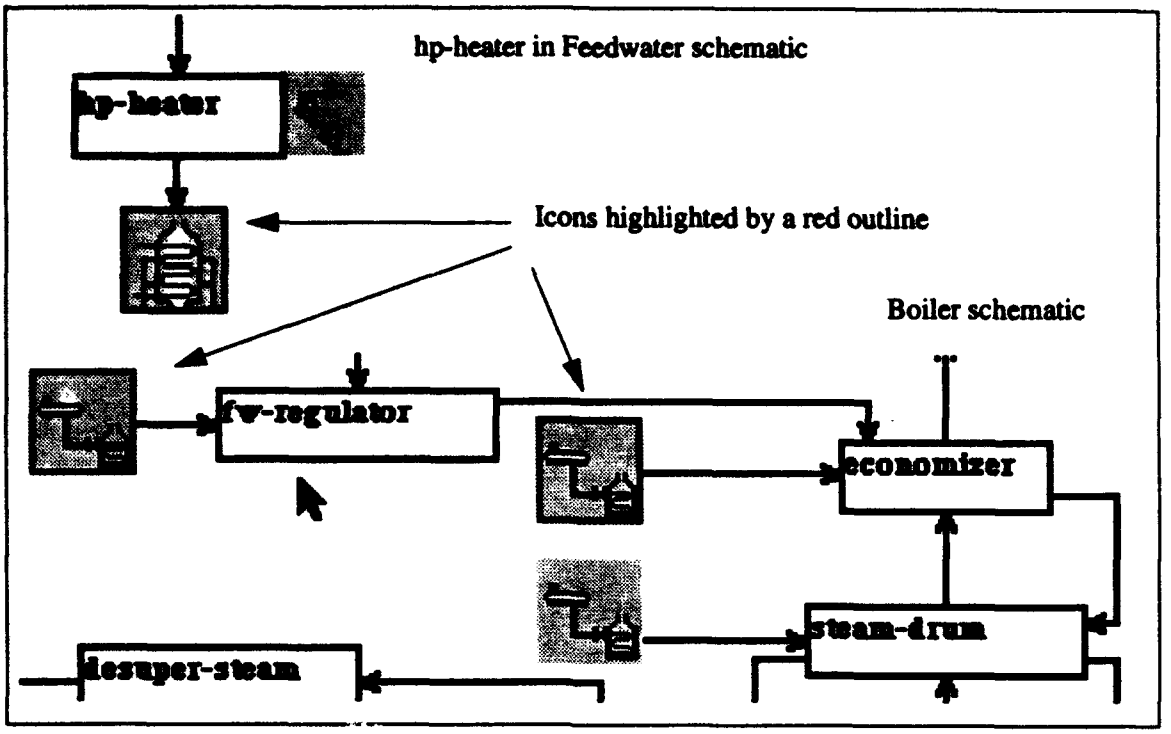


Figure 6. Connections between different schematics

Most components of **Turbinia** are uniquely represented in one of the seven schematics. However, there are a few that have multiple representations. For example, the condenser and the hp heater appear in both the steam and the feed water schematics. Additional occurrences of these components in multiple schematics is indicated by schematic icons attached to these components (see hp-heater in Figure 6). Unlike other schematic icons, these do not have a connector attached to them. If the student clicks on these icons, the display switches to the other schematic that has the second occurrence of the component. The second occurrence of the component in the new schematic is indicated by highlighting in red the schematic icon attached to the component (See Figure 6).

Troubleshooting for failure indicated by the symptoms at the beginning of the session involves gathering information about the system state. The student can collect information concerning the system state using a two-action sequence. The first action of the sequence is an *investigative action*. An investigative action enables the student to display gauges, if any, attached to a component. The second action is an *informative action* that allows the student to access the actual gauge reading. The student takes an investigative action by clicking on a component (see Figure 8) and an informative action by clicking on any gauges displayed by the preceding investigative action (see Figure 10).

The three types of gauges in **Turbinia**, pressure (P), temperature (T), and flow-or-level (L), are represented by icons with letters P, T and L respectively inscribed in them. Although there is no

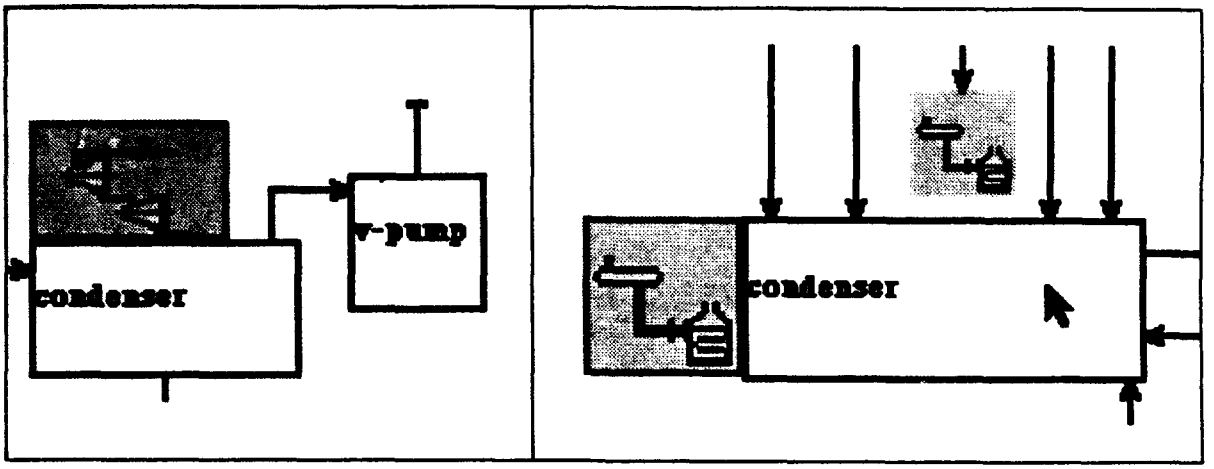


Figure 7. Condenser in feedwater and steam schematics

visible distinction between the level and the flow gauges, the level gauges are mounted on the components while flow gauges appear on the connectors between components. Level gauges in **Turbinia** are attached to tanks (deaerating feed tank, fuel oil settling tank, atmospheric drain tank, distillate tank, hotwell, and drum) and the only gauge that measures flow is located in the fuel oil path across the strainer in fuel oil schematic (Figure 8).

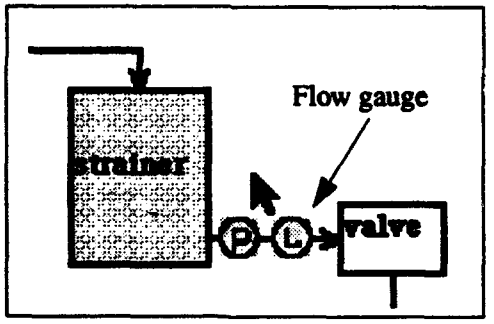


Figure 8. Flow Gauge

When the student clicks on a displayed gauge to probe its reading, an icon appears near the gauge. This icon is a qualitative representation of the current gauge reading. Five different qualitative representations of state values are used in **Turbinia**. These five are low, slightly low, normal, slightly high and high; each is represented by an icon as shown in Figure 9.

Thus, when the student clicks on the drum in the boiler schematic, all four gauges attached to the drum are displayed on the schematic as a result of this investigative action. There are two pressure gauges, one on the flue gas connector to the economizer and the other on the steam drum. There is a temperature gauge on the feed water connector from the economizer and a level gauge on the steam drum (See Figure 10). Now, if the feed water level in the boiler drum is low, the informative action of clicking on the level gauge attached to the drum will result in the appearance of a low level icon below the gauge (also shown in Figure 10). The pressure gauge on the drum shows

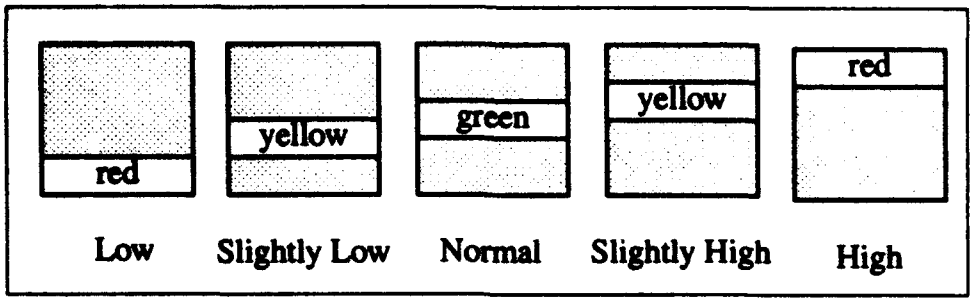


Figure 9. Qualitative State Representation

'slightly low,' whereas the pressure and temperature gauges between the drum and economizer show normal readings.

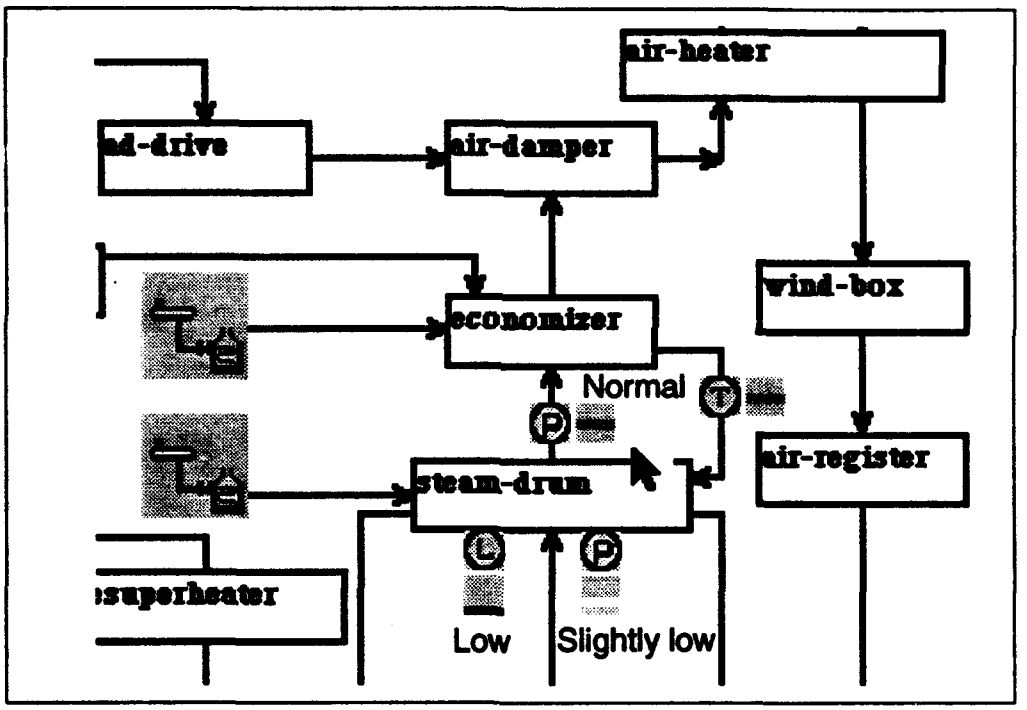


Figure 10. Boiler schematic showing gauges on the steam drum

Gauge readings in **Turbinia** change with time but the displayed gauge readings are not dynamically updated. Therefore, the student must repeat the informative action to view the current gauge reading. This allows the computer-based tutor **Vyasa** to keep track of whether changes in the gauge readings have been observed by the student.

The student can access any displayed gauge or a gauge reading only until a new investigative action is taken. When a new investigative action is taken, the gauges and the gauge readings of the previously investigated component that were visible disappear. This was done to achieve a good mapping between the operator task in the real and simulated domains. In the real domain, components

of power plant are often spread over a large area, or sometimes in many rooms. Thus, multiple gauges cannot be viewed together. Even when the components are located in the same room, often their sizes are huge and it is often not possible to view gauges attached to two components at the same time.

When the student is ready to diagnose the fault, a request for conveying the diagnosis must be submitted. This is done by clicking on the diagnose icon in the requests menu and it puts **Turbinia** in a diagnose mode. After switching to the diagnose mode, **Turbinia** asks the student to select the suspected component. This message is conveyed to the student through text appearing in the tutor dialog at the bottom of the small screen (Figure 11). The student can now use the same action that was earlier used to pick the component for investigation to identify the failed component. If the student's diagnosis is correct, a congratulatory message appears on the tutor dialog (Figure 12). Otherwise, an error dialog accompanied by a beep is displayed over the schematic. This error dialog is shown in Figure 13. The student can close this error dialog by selecting one of the two options available. The student can either revise the diagnosis by choosing "try again" or get back to the troubleshooting mode by choosing "investigate".

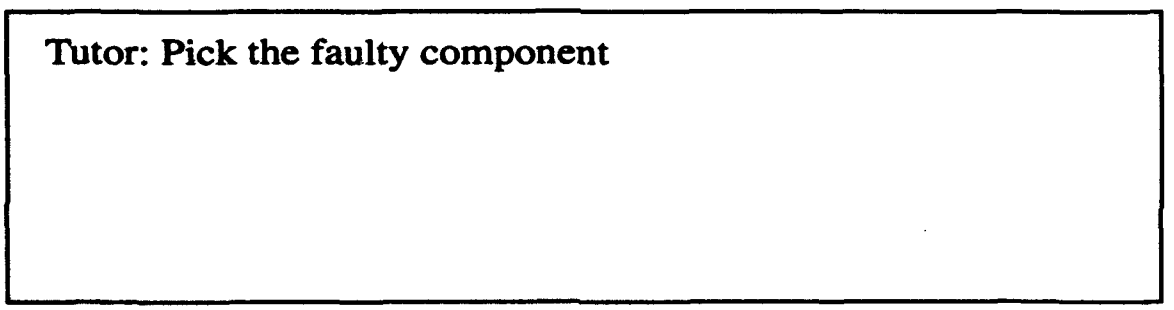


Figure 11. Tutor's Instructions to Select the Suspected Component

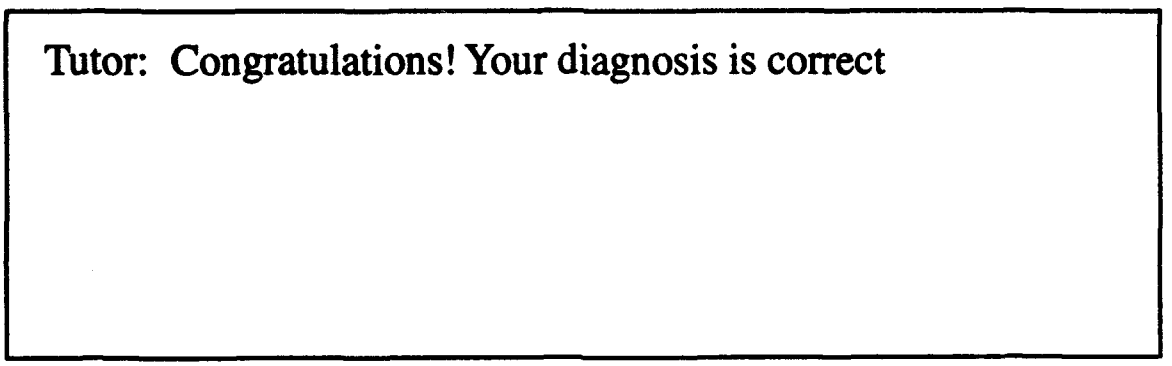


Figure 12. Message of Congratulations on Correct Diagnosis

In addition to the components, connectors and schematic icons, all schematics display two other icons that do not represent a schematic. One icon appears on the bottom left corner and the other on the bottom right corner of all schematics. The icon on the bottom right corner is the Georgia

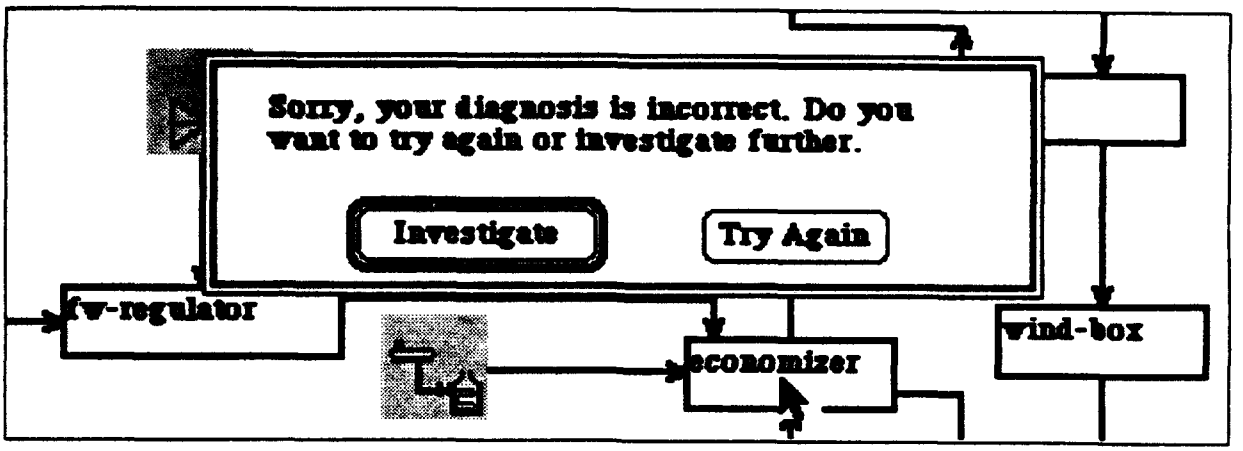


Figure 13. Error Dialog for Incorrect Diagnosis

Tech copyright icon. This icon is disabled and has no response. The icon at bottom left corner is a symptom icon and is used to recall the initial symptoms. Thus, the student can access the ship's initial operating conditions and the initial symptoms at any time.

Interaction with Vyasa

Vyasa uses its knowledge base and the current system state to provide help. The specific nature of help depends on the mode, active or passive. Content and form of this help dialog is described next, followed by a detailed description of the interaction in the passive and active modes.

Content

Information provided to the students comes from units of instructional sets prepared in advance. However, the details to be inserted in the instructional sets is context-driven. For example, when a student inquires about the behavior of a component, the information is always presented in the same format on a dialog box. It consists of relationships between input and output states of the component. Details of the input-output relationships depend upon the functional primitive that represents the component. Thus, although the details of the information presented are context-specific, they are extracted verbatim from the tutor's knowledge base.

The instructional template used for advice generation in response to the hypothesis menu in the active tutor mode is the same on all occasions for the same type of advice. In all there are four such instructional templates used by the tutor (See Figure 14, 14, 15, 16). Each template is used for a different type of advice. One is used for suggesting hypotheses revision, another to indicate lack of adequate evidence to pursue a hypothesis, a third to suggest a diagnostic test to strengthen or weaken a hypothesis; and a fourth to convey the tutor's inability to provide advice under existing conditions.

Form

Like content, the form of instructional presentation is also context-dependent. For example, an-

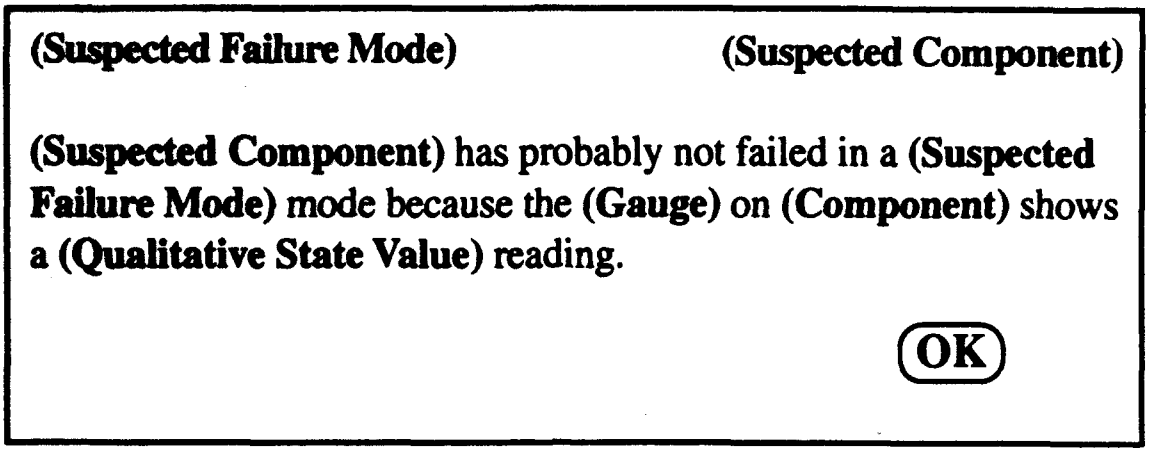


Figure 14. Instructional template "Component not failed in mode"

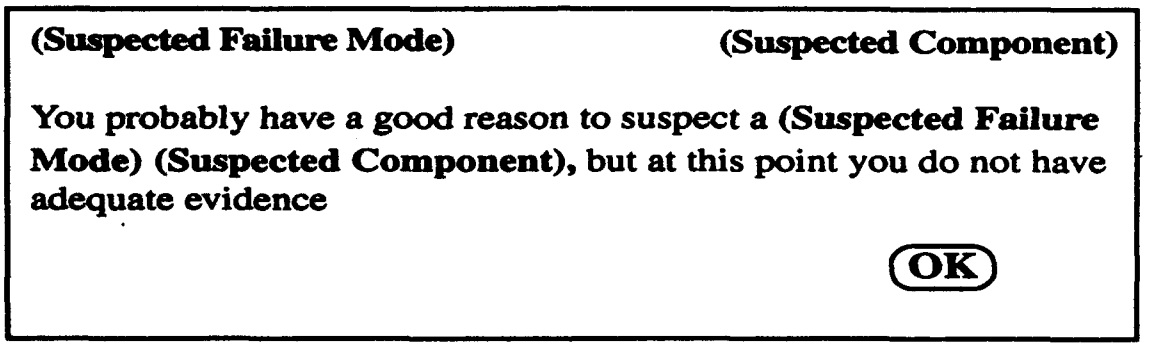


Figure 15. Instructional template "Insufficient evidence"

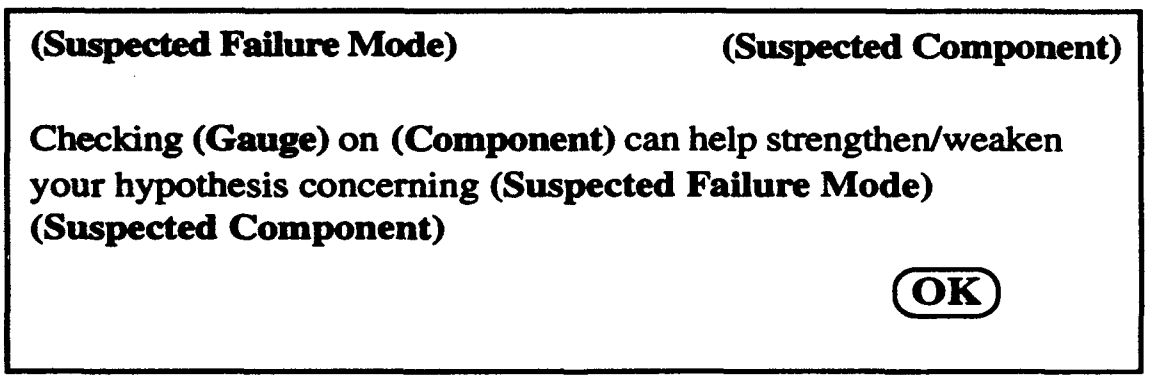


Figure 16. Instructional template "Specific help"

swers to queries related to subsystems and fluid paths that can be visualized on the simulator interface are presented graphically. Where graphics is unlikely to enhance the understanding of the instructions or where graphical aid is computationally expensive the instructions are presented in the form of text. For instance, when the student wants to know the components that constitute a particular subsystem, showing these components by highlighting them in schematics is preferred

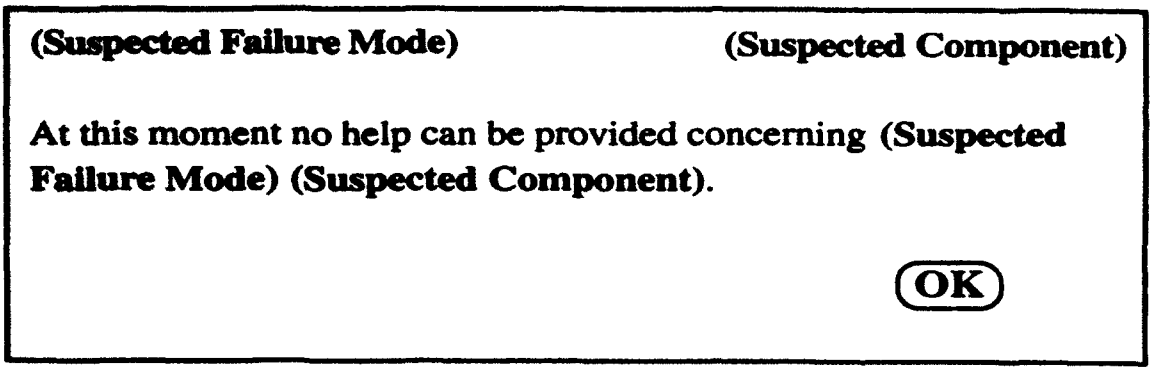


Figure 17. Instructional template "No help"

over listing the names of the components in a dialog box.

Form of presentation include more than just the graphics and/or text. There are certain instructions that convey an error message or require immediate attention. When these instructions are displayed, they are accompanied by a beep. Instructions of this type must be acknowledged before the student is permitted to proceed further. Such instructions disable the mouse until the student performs a specified action.

Passive Mode

When **Vyasa** operates in the passive mode, the student is responsible for initiating communications with the tutor to learn about the system and the failures. Student-initiated interaction with the tutor is accomplished by clicking on the stop icon in the requests menu. This action halts the simulation temporarily, enabling the student to interact with the tutor while preserving the information concerning system states.

Whenever the passive tutor is invoked, the stop and the resume icons change their background colors. The stop icon background changes to yellow-brown indicating that it has been disabled. At the same time, the background of resume icon turns gray indicating that it has been enabled. The cursor too changes shape and turns into a "?". All these changes indicate that the student is not in the troubleshooting mode and hence cannot investigate components and view gauge readings.

When the passive tutor is invoked using the stop icon, a *help-levels* dialog appears in the top left corner of the large monitor. This dialog box is shown in Figure 18. This dialog has seven buttons of which two are enabled. The two highlighted buttons indicate the levels of help that the tutor can provide in the passive mode. These two levels are for the failure and system knowledge. Using these buttons the student can access knowledge concerning specific modes of failure, components, subsystems, and fluid paths. The information accessed by the student is presented in textual and/or graphical form as appropriate.

To access knowledge about the system, the student must choose the "system" button in the help-levels dialog. Following the selection of the "system" button, the "components," "subsystems," and

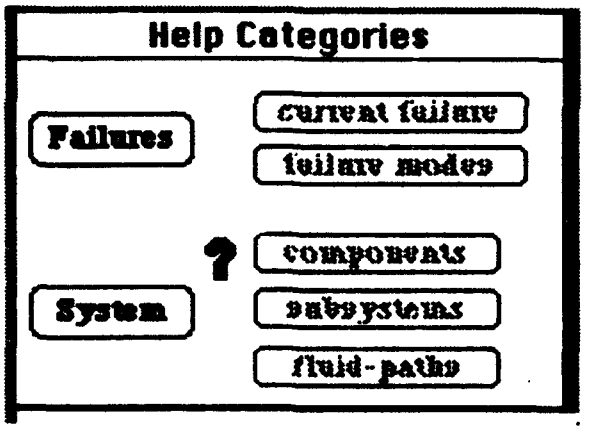


Figure 18. Help-Levels Dialog

"fluid-paths" buttons are enabled (Figure 19). These three buttons provide the student with further options to select the type of system knowledge. When the student selects any one of these three buttons, a new passive tutor help dialog associated with the selected button appears next to the help-levels dialog. This new dialog also contains several selectable items. The student can, by selecting items in the passive tutor help dialogs, explore the entire knowledge-base of the tutor at the component, subsystem and fluid path levels.

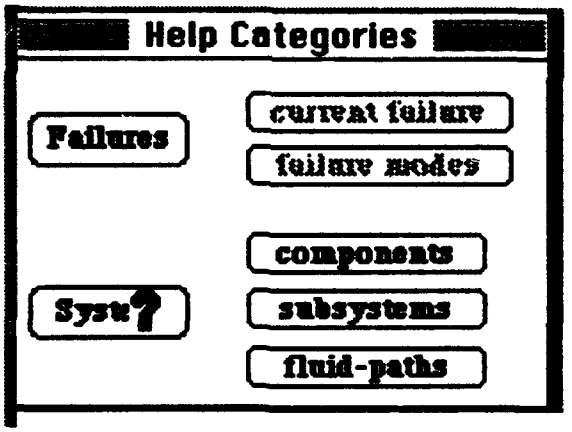


Figure 19. Help-levels dialog with "System" buttons enabled

For instance, if the student selects the "subsystems" button in the help-levels dialog, another dialog that lists all the subsystems in the power plant appears next to the help levels dialog (Figure 20). The student can select a subsystem from the list of subsystems to further explore the tutor's knowledge of the selected subsystem. The selection is made by clicking on the subsystem name listed in the dialog. After the student selects a subsystem, a dialog box listing the selected subsystem and the types of information concerning the selected subsystem that can be accessed by the student are displayed.

Figure 21 is an example of such a dialog box which is displayed when the student selects the steam

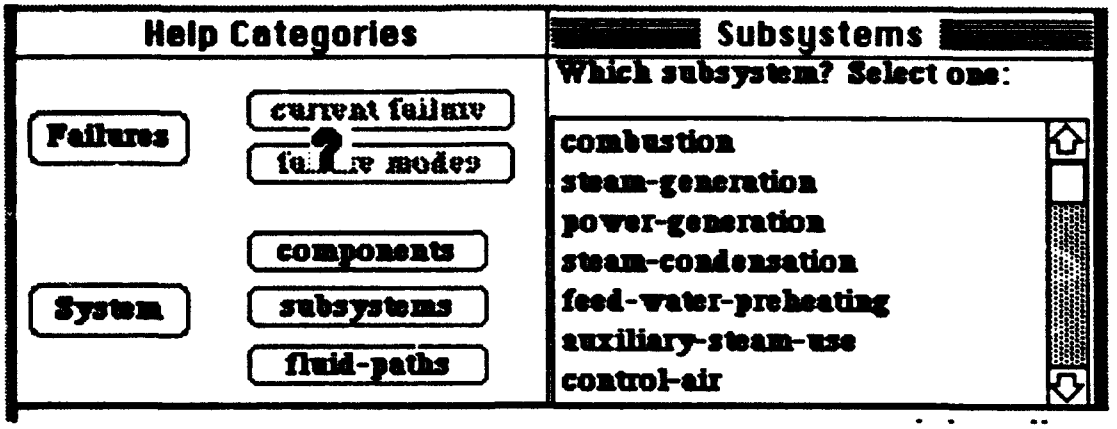


Figure 20. Passive tutor dialog to select subsystem

generation subsystem. This dialog provides the student with three options to further explore the steam generation subsystem: (1) view the components that make up the subsystem, (2) view the fluid paths that pass through, or (3) ask for the description of the function performed.

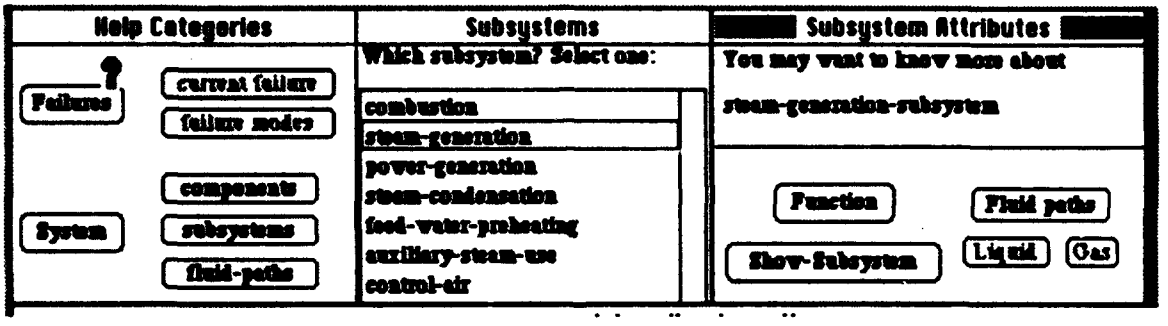


Figure 21. Passive tutor dialog to access subsystem related knowledge

If the student decides to view the components that constitute the steam generation subsystem, the button marked "show subsystem" must be selected. A click on "show subsystem" results in the appearance of a dialog box containing the seven schematic icons (Figure 22). Those icons that represent schematics in which the steam generation subsystem can be found are highlighted in this dialog box. Since the steam generation subsystem can be found in the boiler and fuel oil schematics, the boiler and the fuel oil schematic icons are shown highlighted (see Figure 22). A click on either of these icons displays the schematic represented by the icon and highlights in red all components that constitute the steam generation subsystem in that schematic.

The set of hierarchically organized passive tutor help dialog menus that guides the student's exploration of system knowledge also maintains *context-sensitivity* and offers substantial flexibility to the student. For example, when the student inquires about fluid paths in Figure 21, the tutor offers a choice of selecting a liquid or gas path from only among those that can be found in the inquired subsystem. Thus, knowledge about subsystems and fluid paths does not have to be obtained inde-

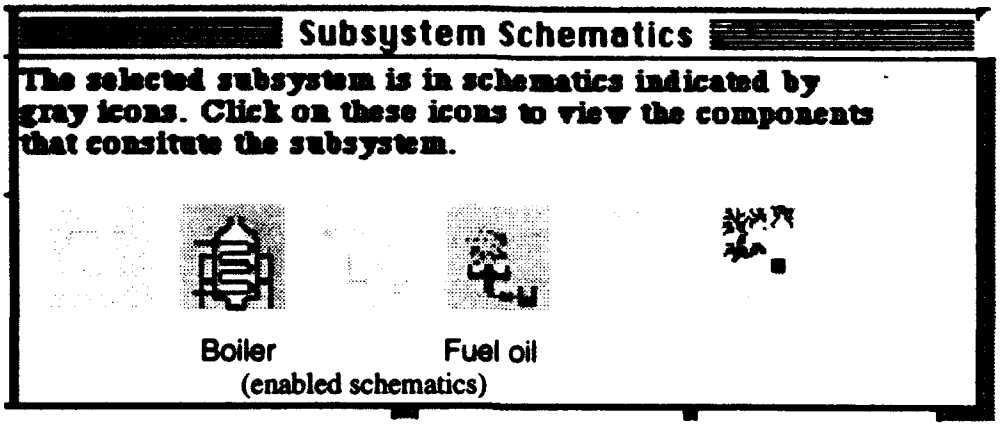


Figure 22. Passive tutor dialog to display selected subsystem

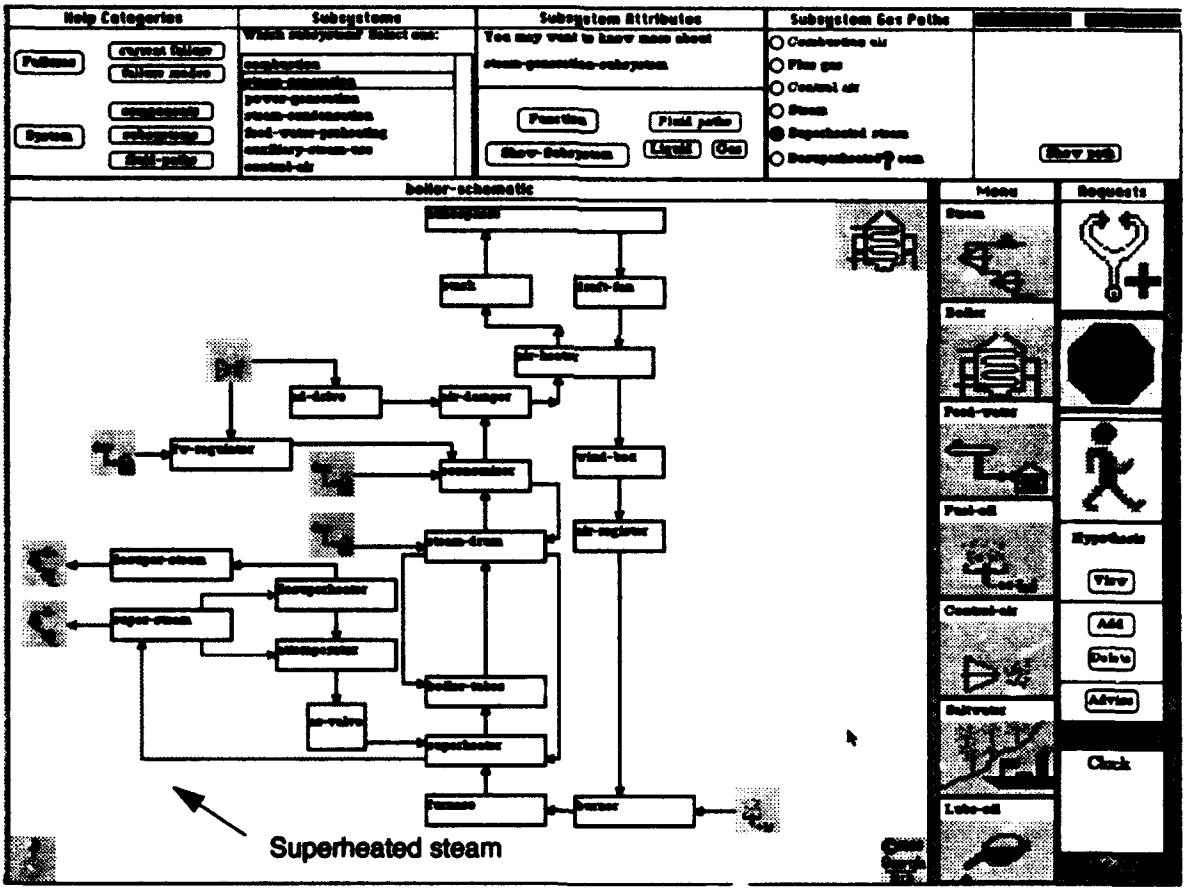


Figure 23. Superheated steam within steam subsystem highlighted in boiler schematic

pendently to deduce which fluid paths lie in which subsystems. For instance, Figure 23 shows the superheated steam path in the steam subsystem.

In general, **Vyasa** responds to an interaction in passive tutor help dialogs by either highlighting the lowlighted buttons in the currently active dialog box or displaying a new dialog box with certain

items highlighted. The student can select an enabled button or any highlighted item in the displayed dialogs to make the queries more specific. Sometimes, it may be necessary for the student to select an item in the schematic or use the keyboard to make the query more specific. For example, when inquiring about a component, the student must select the component by either clicking on the component in the schematic or typing its name using the keyboard. In any event, when a query is specific enough for **Vyasa** to comprehend, it responds with an answer. Otherwise, an error is indicated. The answer is presented as text in a dialog box or as graphics in the schematics.

The flexibility in interaction with **Vyasa** also enables the student to alter the query at any time. This can be done by merely discontinuing the sequence of actions necessary to express the current query and switching to a new query by clicking on a highlighted item in any of the displayed dialog boxes. In such cases, only the dialog boxes relevant to the new query are kept open by the tutor while all others are closed. For instance, if a student wishes to learn more about a component, she can click the "components" button in Figure 21; the component menu will replace the subsystems submenu and the submenus to the right will disappear.

Thus, the student can access tutor's knowledge of the system by following a sequence of straightforward interactions with passive tutor help dialogs. The tutor assumes the responsibility of guiding the student's interaction and provides lot of flexibility to the student to express and alter queries. The upper portion of Figure 24 provides a summary of interactions with the passive tutor help dialogs to access the different components of system knowledge.

The help levels dialog can also be used to obtain information concerning failures. To do so, the student must choose the "failures" button in the help-levels dialog shown earlier (Figure 18). Following the selection of the "failures" button, all buttons related to the "system" button are disabled and all visible dialogs concerning any earlier query related to system knowledge are closed. At the same time, the "current failures" and "failure modes" buttons are now enabled. Using the "failure-modes" button the student can access information concerning typical system behavior associated with each mode of failure in the liquid and gas paths (Figure 25).

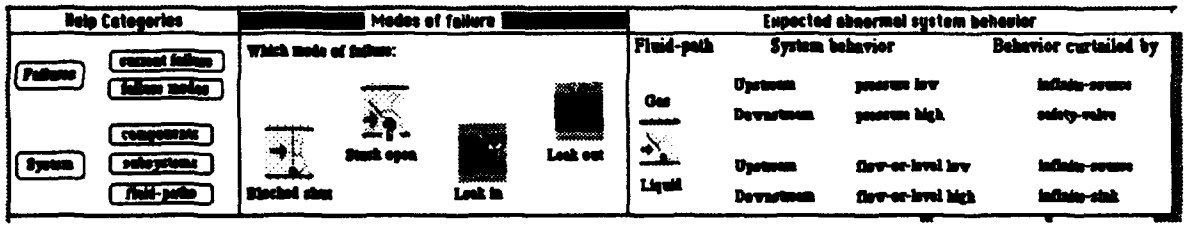


Figure 25. Passive tutor dialog to display failure mode knowledge

Using the "current failure" button the student can bring up a clipboard that extends to the smaller screen on the right. The lower portion of Figure 24 provides a summary of interactions with the passive tutor help dialogs to access tutor's failure knowledge.

The clipboard presents a summary of observed results from the student's diagnostic actions. Based on the observed gauge readings, the clipboard displays the schematics, subsystems and fluid paths

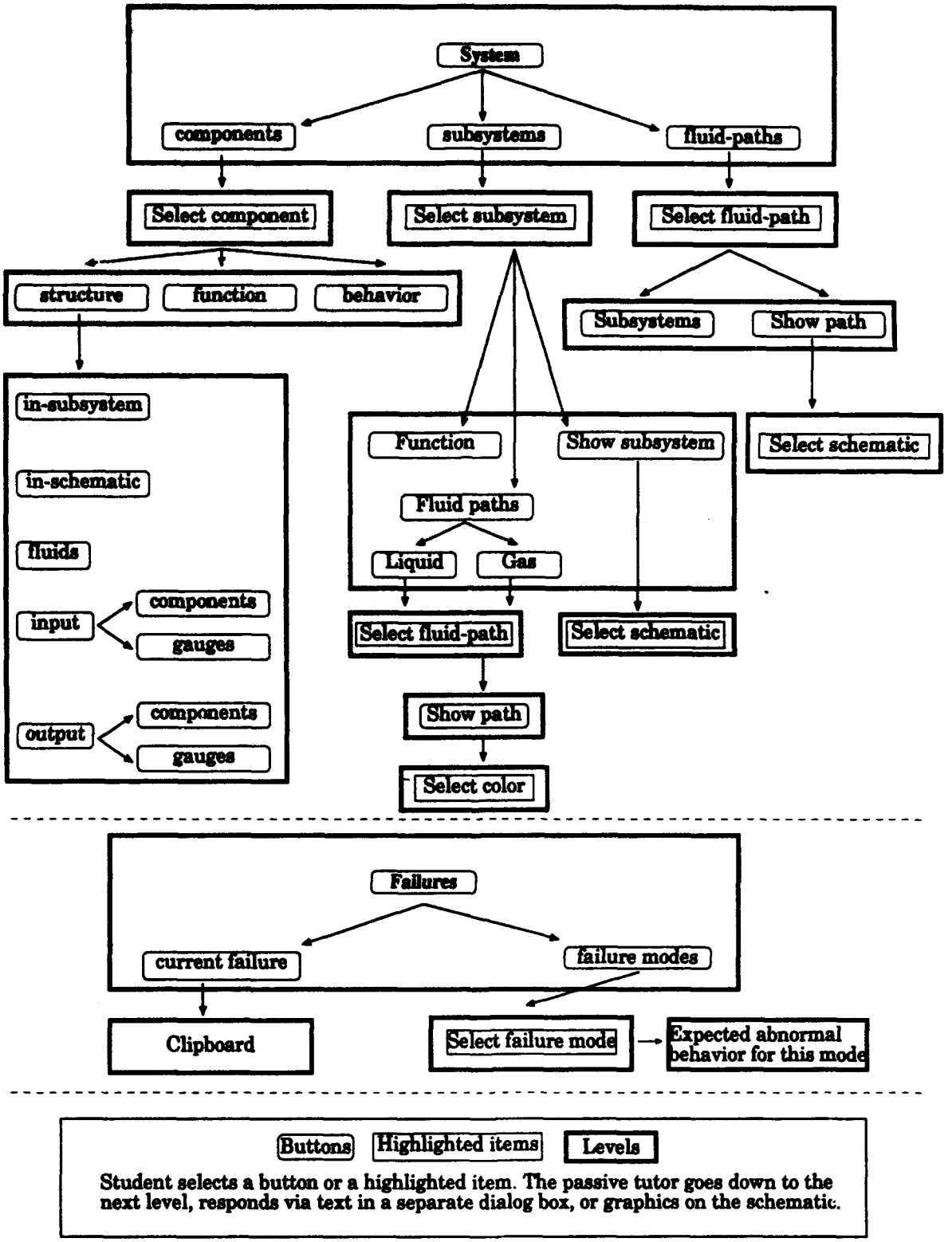


Figure 24. Summary of interactions with passive tutor dialogs to access system and failure knowledge

that contain the affected gauges. The extended portion of the clipboard on the smaller screen displays the gauges probed along with their gauge readings. For example, if the student has only investigated the level gauge on the steam drum since the start of the troubleshooting session and found it to be low, the clipboard will show the boiler schematic, steam generation subsystem and feed water, steam, and superheated steam paths as the affected schematic, subsystem and fluid paths respectively (Figure 26). In addition, the extended portion of the clipboard displays the drum's level gauge investigated by the student (Figure 27). By displaying this information, the clipboard eliminates a student's need to use paper and pencil to keep a record of the diagnostic information gathered during troubleshooting.

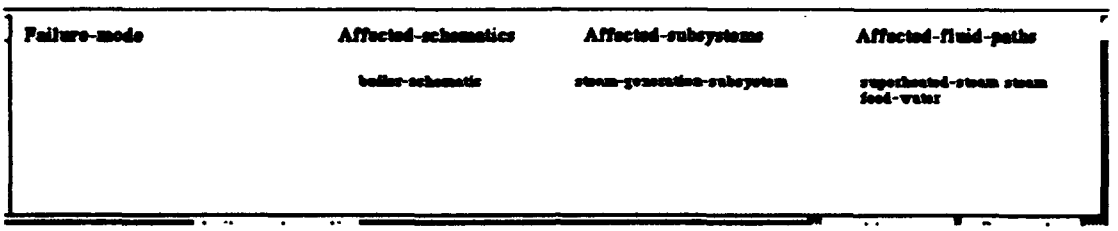


Figure 26. Clipboard (left screen)

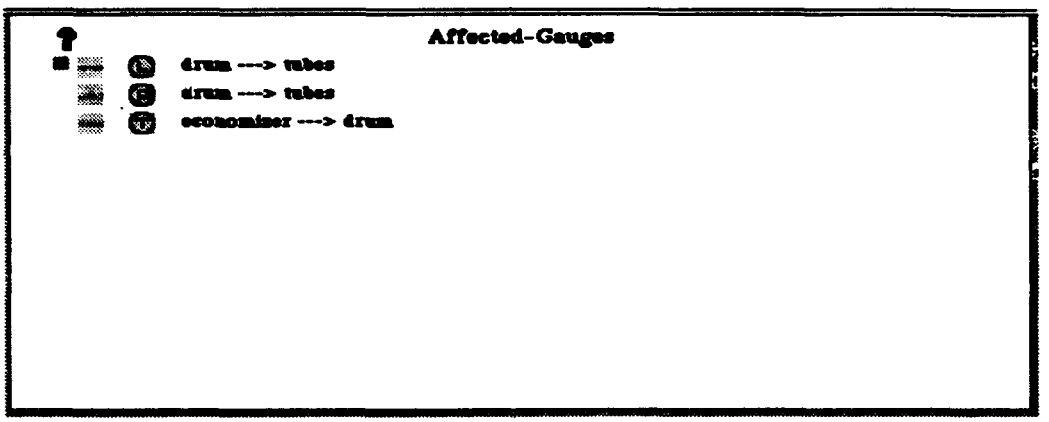


Figure 27. Extended portion of the clipboard (right screen)

The clipboard helps the student in more than one way. Apart from book-keeping, the clipboard informs the student if any gauge reading has changed since it was last viewed. It does so by displaying a blue colored marker next to the gauge reading (see Figure 27, below the '?' mark cursor). Once the student re-investigates the gauge, the marker disappears and the clipboard is updated to contain the latest information. If a gauge reading is not stable, the blue marker next to the gauge may appear and disappear on its own. This is an indication for the student that the gauge reading is perhaps oscillating.

The clipboard also informs the student about the most likely mode of current failure, if and when it can be inferred from the tests conducted. For instance, while investigating the cause for low feed water level in the drum, if the student finds the feed water level to be high in the deaerating feed

tank, "blocked-shut" is posted as the most likely mode of failure (Figure 28). This mode of failure is inferred from the two gauge readings observed by the student: a high feed water level in the de-aerating-feed-tank and a low feed water level in the steam drum.

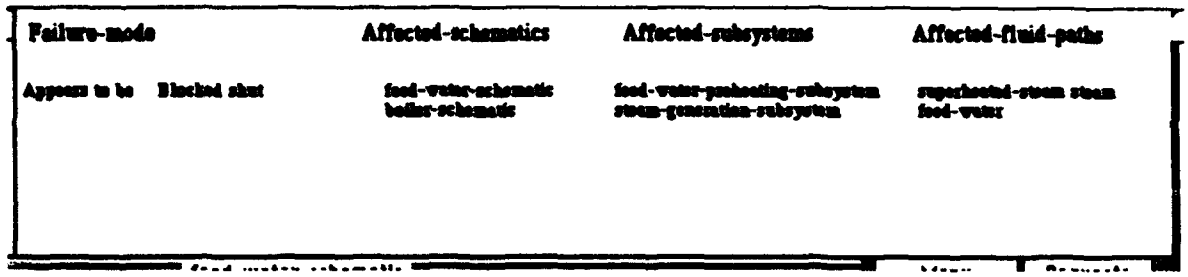


Figure 28. Clipboard indicating blocked-shut mode of failure

The student can get back to the troubleshooting mode by clicking on the resume icon in the requests menu. A switch back to the troubleshooting mode is indicated by the changes in the background colors of resume and stop icons. These icons revert to their original colors and the cursor changes back to the shape of an arrow. However, all the passive tutor help dialogs last displayed remain visible on the screens. Thus, even in the troubleshooting mode, the student can have the clipboard visible on the screen. She can observe the changes that are posted on the clipboard as a consequence of further diagnostic actions.

If the passive tutor is to be invoked again after it has been invoked at least once during a session, the student may do so by clicking on any selectable item in any of the displayed passive tutor help dialogs. This action has the same effect on the cursor shape and the background colors of stop and resume icons as when the tutor is invoked via the stop icon.

Active Mode

In the active mode, **Vyasa** often intervenes to communicate with the student. It does this through instructions presented on the tutor dialog, accompanied by a beep. These instructions are delivered following the evaluation of a student's misconception by the tutor. Instructions are generated by filling in context-specific information into the templates stored in the tutor's knowledge base.

The tutor monitors the student's actions continuously and offers help when it infers a misconception. For instance if the student investigates schematics, subsystems or fluid paths unaffected by the failure, the tutor delivers the appropriate instructions to guide the student away from unaffected portions of the power plant (Figure 29). Such instructions are usually displayed for a fixed, but short, period of time since the instructions lose their context due to system dynamics and student actions.

In addition to guiding the students with instructions, the tutor in the active mode is capable of helping the students with their hypotheses. Student's hypotheses concerning failures are either solicited by the tutor or voluntarily disclosed by the student. In either case, the manner of communicating

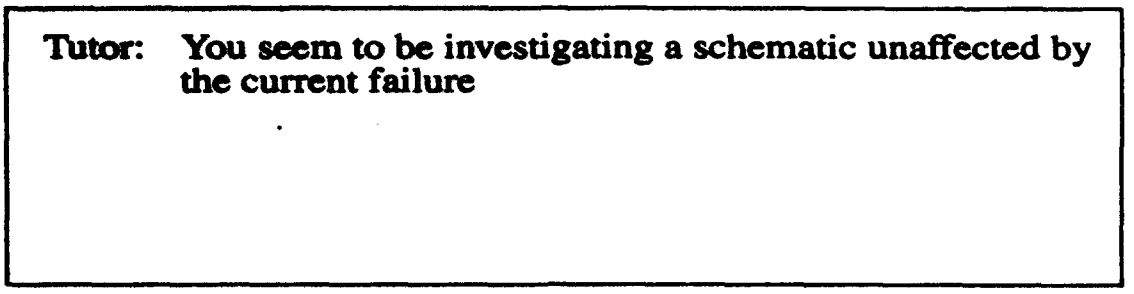


Figure 29. Examples of Instructions from Vyasa

the failure hypotheses to the tutor is identical. If the student has not provided any failure hypotheses to the tutor during the first five minutes, the tutor asks the student to select suspected component(s) responsible for the current abnormal system behavior (Figure 30). Once the selection is made, the student is prompted to indicate the failure mode. The student indicates the mode by picking the appropriate icon in the displayed dialog box (See Figure 31; economizer in the selected component in this example).

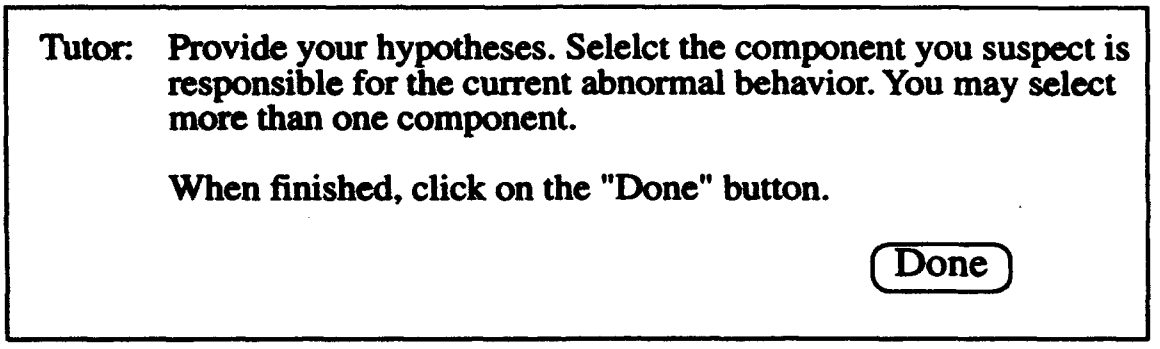


Figure 30. Tutor Soliciting Hypotheses from the Student

When adding hypothesis, either on request from the tutor or otherwise, the action of selecting the suspected component is identical to the investigative action in the troubleshooting mode. Also, the student can add multiple hypotheses at the same time. To add multiple hypotheses, the sequence used to add the first hypothesis is repeated. In fact, the student remains in the mode of adding hypotheses until the "Done" button in the tutor dialog is pressed. After clicking on the "Done" button the student returns to the troubleshooting mode. It is, however, considered an error to click on the "Done" button without providing a single hypothesis when the tutor requests for it.

Vyasa provides help with the hypothesis in two ways: with and without intervention. When the tutor notices that the student is pursuing a hypothesis that should have been rejected based on evidence already gathered, the tutor intervenes to provide evidence against the hypothesis. For example, if the student suspects the condensate pump to be blocked shut in spite of having observed a normal reading on the pressure gauge attached to the cpd valve, the tutor attempts to rectify the student's misconception by providing instructions in the manner shown in Figure 32.

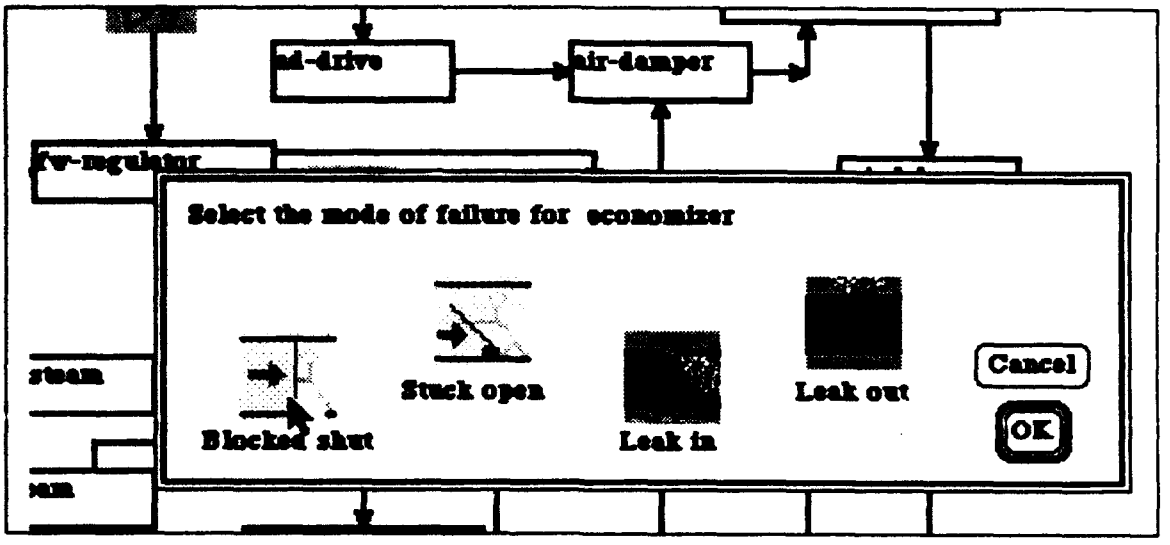


Figure 31. Student Communicating the Suspected Mode of Failure

Tutor: Gauge reading normal for the pressure gauge on the output of cpd-valve is evidence that the

condensate pump has not failed in the Blocked-Shut mode

Figure 32. Example of Hypothesis Aiding with Intervention

Vyasa also provides help with the failure hypothesis without intervention. This help is provided on request. All communications with **Vyasa** concerning failure hypotheses, including a request for help, are carried out via the hypothesis menu shown earlier (see Figure 4). Interaction with the hypothesis menu is described next.

Help for hypothesis

To view, delete, or seek advice on a failure hypothesis, the student must use the "View", "Delete" and "Advice" buttons respectively. The tutor displays the hypotheses provided by the student in a dialog box on the small screen for review. The interaction with the tutor to delete a hypothesis or to seek advice is also straightforward. The tutor prompts the student for every action through the tutor dialog. When deleting a hypothesis, the student first selects the hypothesis in the dialog box shown in Figure 33 and then clicks on the highlighted "Delete" button in the same dialog box. When seeking advice on a hypothesis, the student first selects the hypothesis in the dialog box shown on the small screen and then clicks on the highlighted "Help" button. Any advice from the tutor is displayed in a separate dialog box over the "Help" button (Figure 34).

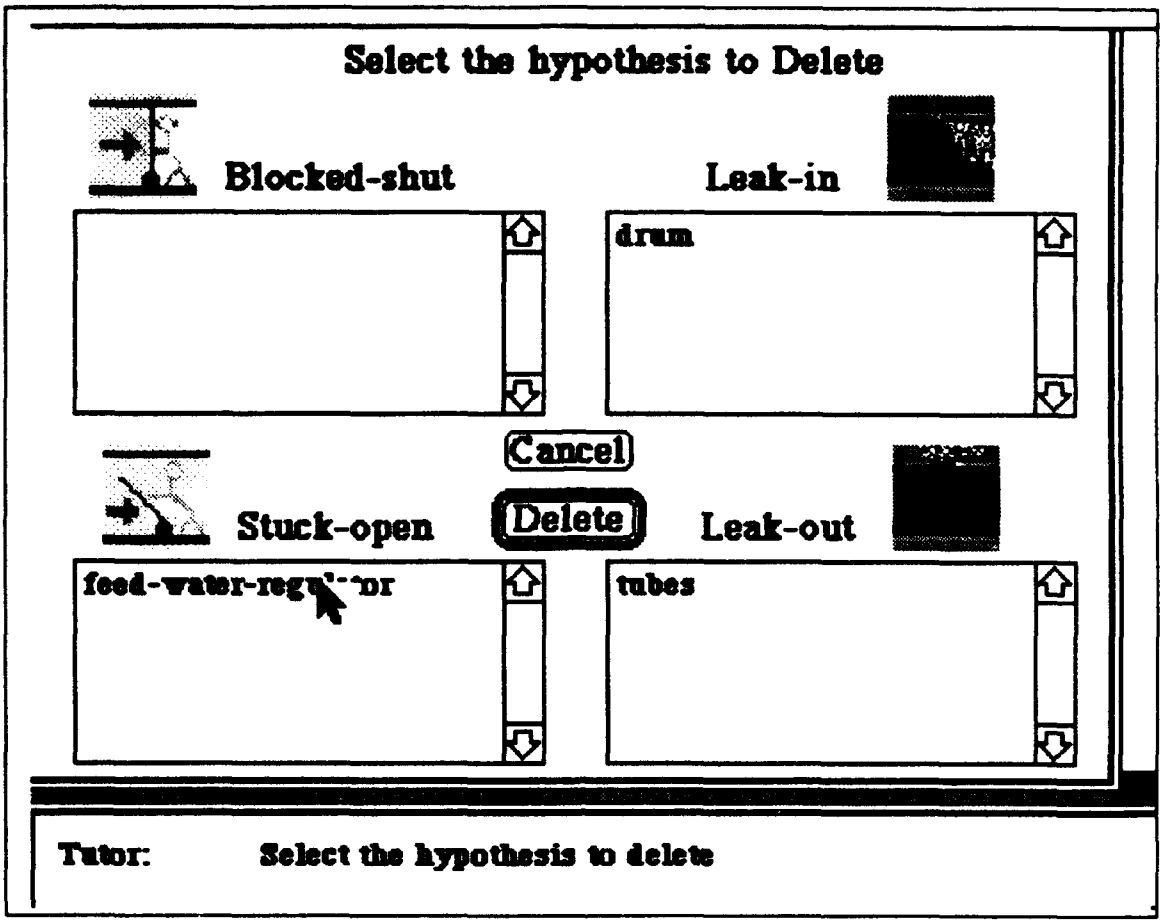


Figure 33. Delete hypotheses dialog

In addition to the interaction described thus far, the student interacts with the instructional system at the end of every problem solving session to view the solution to the problem last presented. The solution and the interaction with the instructional system depends upon whether the student was using only the simulator or was also aided by the tutor. In either case, the student is presented with a dialog box shown in Figure 35 at the end of each problem. While students using just the simulator see only the solution as shown in Figure 36, the students aided by the tutor have the option to view the explanation for each observed abnormal behavior (Figure 37). If the student decides to click on the "Explain" button in the dialog box shown in Figure 37, explanations are provided for all observed abnormal system behavior. These explanations containing causal reasons for each abnormal gauge reading are presented, one at a time, and in the order in which the gauges are affected by the failure. In presenting the reasons for each abnormal gauge reading, first the schematic which contains the affected gauge is displayed, then the affected gauge along with its gauge reading are made visible and finally an explanation for the abnormal reading is displayed in a dialog box on the small screen (Figure 38). Once the student has read the explanation and clicked on the "OK" button, the tutor proceeds to provide a similar explanation for the next affected gauge. This process continues until the tutor completes providing an explanation for each abnormal gauge reading caused by the

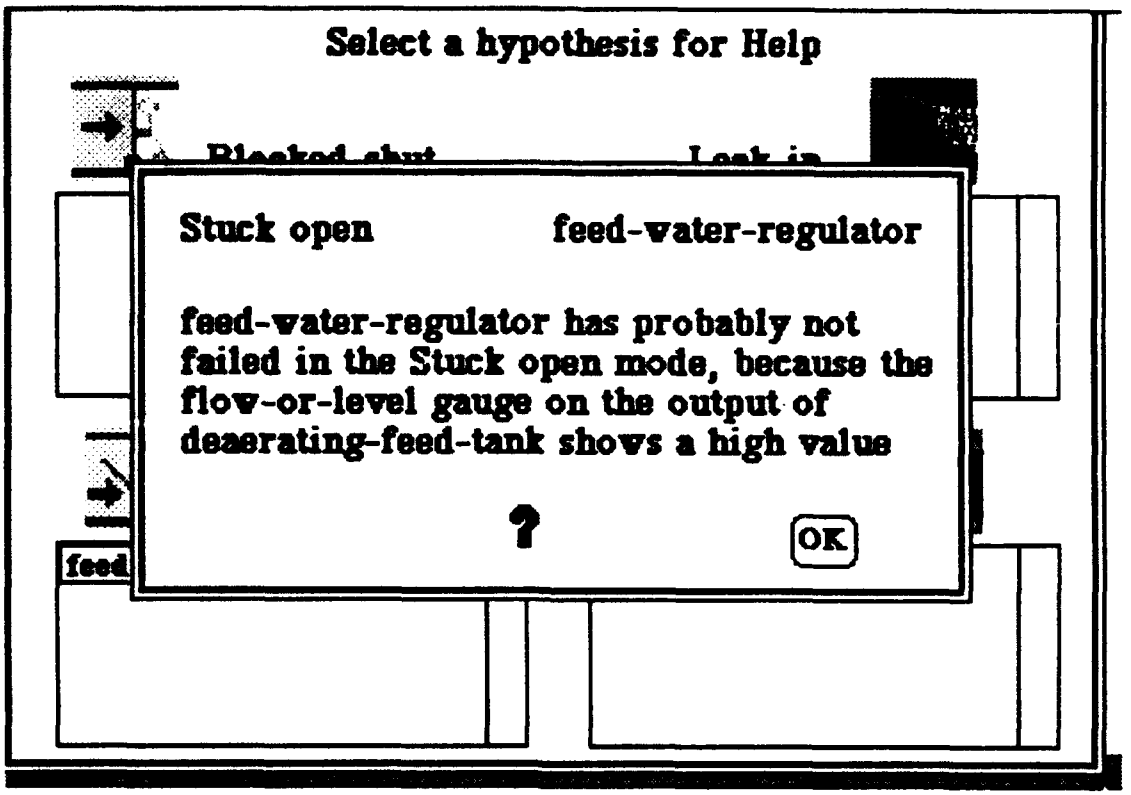


Figure 34. Example of hypotheses aiding without intervention

failure.

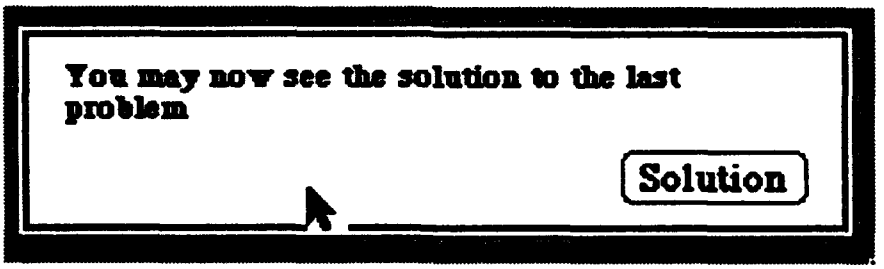


Figure 35. Dialog to request solution

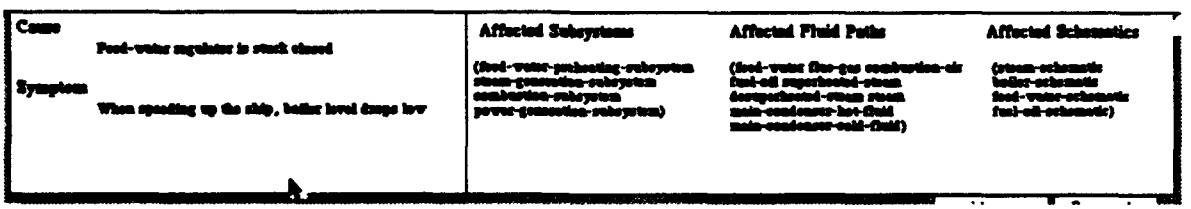


Figure 36. Solution for students trained on simulator

We have described the student-tutor interface of **Turbinia-Vyasa** and the valid forms of operator interactions at this interface. In the next section, we describe briefly an experimental study to evaluate the ITS architecture implemented in **Turbinia-Vyasa**.

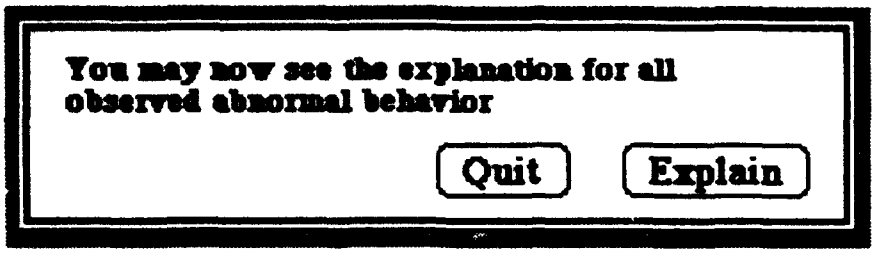


Figure 37. Solution for students aided by the tutor

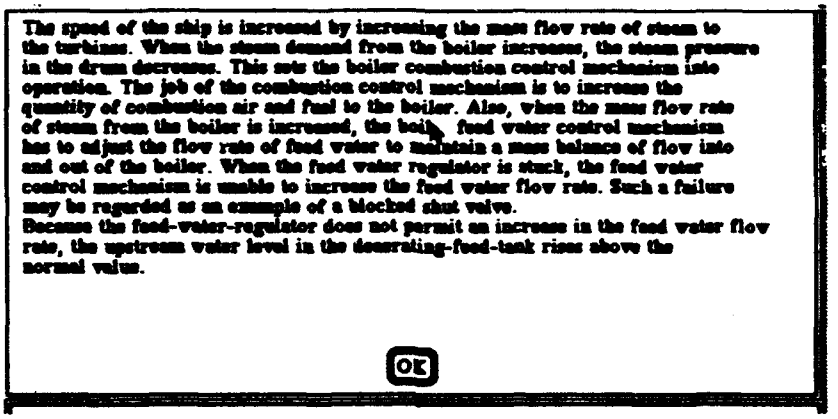


Figure 38. Explanations for abnormal system behavior

Experiment

In the experiment, we compared the performance of subjects trained with and without the tutor. We should point our that the purpose of the experiment was not to evaluate the interface. Instead, we evaluated the interface implicitly in the context of the tutoring system.

Even though there was no explicit evaluation of the interface itself, the evaluation does point to the desirability of the interface as an integral component of the ITS. In our experiment, we sought to answers three questions. First was the feasibility of building an effective computer-based tutor by implementing the ITS architecture. The next question was whether training by computer-based tutor was better than training provided by the simulator alone. Finally, we explored the effect of the level of aiding during training on performance.

Thirty paid volunteers, who were students at Georgia Institute of Technology and cadets with the Naval Reserve Officers Training Corps unit, participated as subjects. All except one subject were male. Subjects had a basic understanding of the theory of marine power plants. Among those selected for the experiment were twenty-four sophomores and six juniors. A few of the subjects had additional exposure to thermodynamics through course-work or had limited experience operating the marine power plants.

For every session completed in both the training and the testing phase each subject was paid $6 per session. In addition, an award of $25 was promised for the best troubleshooter in each of the three

groups based on performance in the two data collection sessions. All subjects were told about the performance measures prior to the experiment. They were also informed that for the purpose of determining the award, the number of problems correctly diagnosed in minimum time was the only measure to be considered.

Subjects were randomly assigned to the three experimental groups. The experiment consisted of two phases: training and data collection. In the training phase, subjects were exposed to one of the three instructional methods: (a) training on simulator alone (S); (b) training with the aid of a passive tutor (P); and (c) training with the aid of an active tutor (A). During data collection, trained subjects from all three conditions attempted to solve the same set of problems unaided by the tutor. The effect of training was then evaluated in the data collection phase where all subjects were exposed to identical problems on **Turbinia** without the aid of the tutor.

There were ten training sessions, each lasting no more than forty-five minutes. The sessions were run on consecutive days with typically one session per day. Occasionally, when a subject missed a day, the lost session was made up by extending the training period by a day. Under no circumstances was a subject permitted multiple sessions in a day.

The first training session for each group introduced the system using a single problem. Audio taped instructions, different for each group, were used during this session. These instructions introduced the subjects to the interface and valid forms of interactions.

After the first session, subsequent training sessions had three problems each. A subject had thirteen minutes to solve each problem. If the subject solved the problem in less than the allotted time, the next problem was immediately presented. Thus, if the subject solved one or more of the three problems in a session within the allotted time for each problem, the session could potentially be completed in less than forty-five minutes.

At the end of each problem the subject was provided the solution. While solutions presented to subjects with the tutor were accompanied by an explanation, no such explanation was provided to subjects using the simulator alone.

The data collection phase consisted of two sessions. These sessions were run on consecutive days immediately following the completion of training. During these sessions, the subjects interacted with the simulator only, unaided by any tutor, irrespective of their training condition. Thus, even the subjects in Groups P and A who were earlier aided by the tutor were unaided during the data collection sessions.

Each data collection session was approximately fifty minutes long and consisted of five problems. If the subject solved the problem within the ten minute time period allocated for each problem, the next problem was immediately presented. However, unlike the training sessions, no solution was provided to the student at the end of the problem. At the end of the data collection sessions, all subjects completed an exit questionnaire.

Experimental Results

The data were analyzed using the SAS General Linear Model and Type III sum of squares. The effect of training condition on the performance of the subjects is summarized Table 1. A brief discussion follows.

Table 1. Summary of Training Condition Effect

Performance Measures	Training Condition			Performance Comparison (α=0.05) * Significant at α=0.1
	Simulator (S)	Passive Tutor (P)	Active Tutor (A)	
Product Measures				
Percentage of problems solved	93.00	95.00	88.00	Not significant
Troubleshooting time (minutes)	2.62	3.43	3.69	Not significant*
Process Measures				
Number of informative actions per problem	10.72	8.18	8.83	Not significant*
Percentage of relevant informative actions	59.70	72.50	71.50	(S) < (P), (A)
Percentage of guesses	71.40	35.23	29.50	(S) > (P), (A)
Investigations (per problem) in unaffected				
Schematics	0.36	0.81	1.81	Not significant*
Subsystem	0.12	0.40	1.00	(S) > (P), (A)
Fluid-paths	0.23	0.35	0.98	(S) > (P), (A)
Nature of diagnosis (% of solved problems)				
Premature	26.80	53.70	19.35	(S) > (P), (A)
Timely	14.73	81.00	4.20	(S) < (P), (A)
Overdue	9.00	85.22	5.60	(S) > (P), (A)

There was no significant difference in the *number of problems solved* across the three training conditions. The relatively poor performance by subjects in Group A can be attributed to three factors. First, a single subject was responsible for five of the unsolved problems. Second, subjects in Group A were more inclined to leave a problem unsolved because they were reluctant to guess the failures. Third, the subjects in this group became somewhat dependent on the tutor to solve the problems and when the tutor was withheld from them, during the test sessions, their performance deteriorated.

The *troubleshooting time* was also not significantly different. This result is not at all surprising considering that the unaided group did not have a guided strategy to solve the problems and relied heavily on guessing. Guessing as opposed to abstract reasoning takes less time. However, at α level of 0.1, the effect of training condition on troubleshooting time *was* significant.

Even though the *number of informative actions* was statistically not significant, the data indicate that the subjects in Group S, in comparison to the subjects in other two groups, needed more diagnostic tests to solve the problems. In other words, subjects in Groups P and A utilized the diagnostic information more effectively and required smaller number of diagnostic tests to solve the problems.

The *percentage of relevant informative actions* taken by the subjects in the two aided groups was

significantly higher, implying that those trained by the tutor were better able to identify the diagnostic tests that were useful for solving a problem.

The effect of training condition was significant with higher *percentage of guesses* for the unaided group in comparison to the two aided groups. Evidence of guessing strategy was noticed in 60% of the problems for Group S and only 39% and 30% of the problems for Groups P and A respectively. Also, the data indicate that the subjects in Group S often used guessing as a primary strategy whereas the subjects in Groups P and A started guessing only when they were running out of time.

Detailed analysis of the *number of unaffected schematics/subsystems/fluid-paths investigated* showed that subjects in the two aided groups performed significantly fewer investigations in unaffected subsystems and fluid paths. In other words, subjects in the two aided groups were able to better identify the location of the fault and investigate the relevant portions of the power plant.

The usual SAS analysis of variance was not possible for *the nature of diagnosis*, which consisted of comparing the three mutually exclusive categories of correct diagnoses (premature, timely and overdue) from each training group. Therefore, pair-wise comparisons were performed to detect significant differences across the three training conditions. Of the problems solved, subjects in Group S performed more premature diagnoses as compared to subjects in Groups P and A. Since the subjects in Group S relied rather heavily on guessing, it is not surprising that they got lucky more often. The results suggest that the subjects in the two aided groups either formed a better understanding of cause-effect associations or utilized it more effectively to diagnose faults. Also, for subjects in Group S, more diagnoses were overdue as compared to the two aided groups. This shows that the subjects in Group S were not as good at integrating diagnostic information as the subjects in the two aided groups.

From the results presented above and additional data analysis (Vasandani 1991), it was apparent that the tutor in both the passive and the active modes helped the students to develop useful troubleshooting strategies. Those trained by the tutor formed plausible failure hypotheses based on observed symptoms and systematically eliminated them by conducting appropriate diagnostic tests. In comparison, those trained without the tutor did not develop good troubleshooting strategies. They relied rather heavily on guessing the solution. Furthermore, the tutor helped the students to recognize and integrate crucial diagnostic information in a timely manner that the students without the tutor were unable to do. Students trained by the tutor were better-prepared for unfamiliar situations than those trained on the simulator.

The data also indicated that the effectiveness of a tutoring strategy depended upon the individual student. For example, the strategy of providing explanations for all observed symptoms for each problem was intended to help the students develop a proper causal model of fault propagation. Some students who learned to map salient symptoms to causes from these explanations became overly conservative. During troubleshooting they spent a lot of time eliminating all probable hypotheses linked to an observed symptom even when sufficient evidence in support of a highly probable hypothesis had been collected. Another tutoring strategy adopted by the active tutor was to

provide help in building, refining, and eliminating failure hypotheses. In this capacity the active tutor came to be perceived by some students as an on-line associate. These students often took the help of the active tutor to refine their failure hypotheses and thus became dependent on the tutor to solve problems. Performance of these students deteriorated when the active tutor was withdrawn.

Experimental results show that a simulator alone is inadequate for training purposes. However, a simulator in conjunction with an effective computer-based tutor can help develop efficient troubleshooting skills. Such a tutor must teach operators to identify useful diagnostic tests, use the results of these tests to formulate plausible hypotheses concerning failure, and systematically refine the hypotheses based on new diagnostic data until the cause of failure is identified. Operators trained by such a tutor are likely to rely less on guessing and more on abstract reasoning. Consequently, these operators are likely to provide incorrect diagnoses less often.

In real-world, where there is a cost associated with each incorrect diagnosis, less incorrect diagnoses can save valuable time and reduce troubleshooting costs. However, since not all students are equally receptive to every tutoring strategy, provisions must be made in training programs for individual preferences and differences in abilities and styles. Otherwise, student's may become overly conservative or too dependent on the tutor for help. While conservative behavior may not necessarily be bad, too much dependence on the tutor is undesirable. Therefore, assistance provided by the tutor that directly helps students in solving the problems must be avoided to control the students' dependence on the tutor.

Also, use of certain features of the tutor, like hypothesis aiding and the interactive interfaces, may be useful in other applications such as an on-line operator's associate. Performance of students who exploited the hypothesis aiding feature of the active tutor to successfully solve problems during training suggests the possibility of using this feature in an on-line operator's associate. Students frequently provided the tutor with failure hypotheses and sought advice on each one of them. As a part of its counseling task, the tutor would check if evidence had already been gathered to reject the hypothesis, and if so, the student would be told about it. Thus, the students in fact assigned the tutor the task of filtering out the less likely alternatives and based on the tutor's advice refined their hypotheses until they could identify the failed component with reasonable amount of certainty.

Conclusions

In spite of the progress made in the field of intelligent tutoring, not many ideas of existing ITSs have been successfully extended to real-world applications. One of the reasons for the limited success was attributed to the simplicity of the domains and tasks considered by most existing ITSs. Real-world engineering systems, in contrast, are more complex due to their size, interaction between subsystems, and dynamics. Well-designed interfaces play a major role in making the system knowledge readily accessible in such systems.

In this paper, we described a prototype instructional system, **Turbinia-Vyasa**, with an integrated, interactive, interface designed to enhance the ITS. Our focus was on the details of interaction be-

tween the student and the instructional system. We conducted an experimental study to evaluate the architecture of the instructional system.

The results of the experiment established the viability of designing and implementing an effective tutoring system. The results also demonstrated that instructional systems that integrate intelligent tutors with a simulator and provide access to multiple, complementary, system representations via direct manipulation graphical interfaces can contribute greatly to an effective training program.

In our current research, we are studying diagnostic problem solving to develop models employed by operators with different levels of expertise. We are investigating several domains such as manufacturing systems and computer networks. We hope to enhance the architecture of instructional systems, and contribute to improving and enhancing interactive interfaces. The research results will be used to implement computer-based, "intelligent," operator associates and supervisory control systems that assist people with different skill levels.

Acknowledgments

The research reported here has its roots in work sponsored by a previous grant from the Office of Naval Research (ONR). Drs. Marshall Farr and Henry Halff at ONR realized the need to study troubleshooting in complex, real-world domains, and provided financial and moral support for our work. Later, Drs. Michael Shafto and Susan Chipman saw the need to continue this support via contract N00014-87-K-0482 from Manpower, Personnel, and Training R & D Program to the Georgia Tech Research Corporation. Drs. Farr, Halff, Shafto, and Chipman were cheerleaders and promoters, and performed many other supporting roles that one does not normally associate with faceless government bureaucrats. Dr. Susan Chipman, the most recent Contract Monitor, helped with critical comments and suggestions on methodological issues relevant to cognitive science and training throughout the duration of this project and made efforts to publicize our research. We (especially TG) are grateful for all that they have done. We wish to thank the staff and cadets of the Georgia Tech Naval ROTC unit for their cooperation and help. We especially appreciate the help from Lt. William A. Marriot.

References

1. Albers, J. (1975). *Interaction of color.* Revised edition. Yale University Press, New Haven, CT.
2. Brown, J. S., Burton, R. R., and de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in Sophie I, II and III. In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London.
3. Burns, H., Parlett, J. W., and Redfield, C. L. (Eds.) (1991). *Intelligent tutoring systems: Evolution in design.* Lawrence Erlbaum Associates, Hillsdale, NJ.
4. Fath, J. L., Mitchell, C. M., and Govindaraj, T. (1990). An ICAI architecture for troubleshooting in complex, dynamic systems. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-20 no. 3, pp. 537-558.

5. Frasson, C. and Gauthier, G. (Eds.) (1990). *Intelligent Tutoring Systems: At the crossroad of artificial intelligence and education*. Ablex Publishing Corp., Norwood, NJ.

6. Goldstein, I. L. (1986). *Training in Organizations: Needs Assessment, Development, and Evaluation*. Brooks/Cole Publishing Co., Pacific Grove, CA.

7. Govindaraj, T. (1988). Intelligent computer aids for fault diagnosis training of expert operators of large complex systems. In J. Psotka, L.D. Massey and S.A. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned*, Lawrence Erlbaum Associates, Hillsdale, NJ.

8. Hollan, H. D., Hutchins, E. L., and Weitzman, L. (1984). STEAMER: an interactive inspectable simulation-based training system. *AI Magazine, 5(2)*, pp 15-27.

9. Johnson, W. B. (1988). Pragmatic considerations in research, development, and implementation of intelligent tutoring systems. In Polson, M. C. and Richardson, J. J. (Eds.), *Foundations of intelligent tutoring systems*. Lawrence Erlbaum Associates, Hillsdale, NJ.

10. Kearsley, G. Overview. In Kearsley, G. (Ed.) (1987). *Artificial Intelligence and Instructions: Applications and Methods*. Addison- Wesley, Reading, MA.

11. Lesgold, A., Lajoie, S. P., Bunzo, M., and Eggan, G. (1991). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin, R. Chabay, and C. Scheftic (Eds.), *Computer assisted instruction and tutoring systems: Establishing communication and collaboration*, Lawrence Erlbaum Associates, Hillsdale, NJ.

12. Miller, J. R. (1988). Human-Computer interaction and intelligent tutoring systems. In *Foundations of ITSs*, Polson and Richardson, Eds. Lawrence Erlbaum Associates, Hillsdale, NJ.

13. Miller, R. A. (1985). A systems approach to modeling discrete control performance. In W. B. Rouse (Ed.), *Advances in Man-Machine Systems Research* Vol. II. JAI Press Inc., Greenwich, CT.

14. Moran, T. P. (1983). Getting into a system: external-internal task mapping analysis. *Proceedings of the ACM - CHI Conference on Human Factors in Computing Systems*. Boston, MA, pp 45-49, 1983.

15. Psotka, J., Massey, L. D., and Mutter, S. A. (Eds.) (1988). *Intelligent Tutoring Systems: Lessons Learned*, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

16. Rasmussen, J. (1985). The role of hierarchical knowledge representation in decision making and system management. *IEEE Transactions on System, Man, and Cybernetics*, vol. SMC-15(2), pp. 234-243.

17. Sleeman, D., and Brown, J. S., (Eds.) (1982). *Intelligent tutoring systems*, Academic Press, Orlando, FL.

18. Towne, D. M., and Munro, A. (1988). Intelligent maintenance training system. In J. Psotka, L. D. Massey and S. A. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned*, Lawrence Erlbaum Associates, Hillsdale, NJ.

19. Towne, D. M., and Munro, A. (1990). Model-building tools for simulation based training. In *Interactive learning environments*, 1, pp. 33-50.

20. Vasandani, V., and Govindaraj, T. (1990). Knowledge Representation and Human-Computer Interaction in an Intelligent Tutor for Diagnostic Problem Solving. Proceedings of *The 1990 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 665-667.

21. Vasandani, V., and Govindaraj, T. (1991a). Experimental Evaluation of an Intelligent Tutor for Diagnostic Problem Solving. Proceedings of *The 1991 International Conference on the Learning Sciences*, pp.414-421.

22. Vasandani, V., and Govindaraj, T. (1991b). Intelligent Diagnostic Problem Solving Tutor: An Experimental Evaluation. Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics, Charlottesville, VA.

23. Vasandani, V. (1991). *Intelligent Tutoring for Diagnostic Problem Solving in Complex Dynamic Systems*. Doctoral dissertation, Center for Human- Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

24. Vasandani, V., and Govindaraj, T. (1993, in press). Knowledge structures for a computer-based training aid for troubleshooting a complex system. In D. M. Towne, T. de Jong, and H. Spada (Eds.) *The Use of Computer Models for Explication, Analysis and Experiential Learning*. NATO ASI Series F, Programme AET, Springer-Verlag, Heidelberg.

25. Vasandani, V., and Govindaraj, T. (1993a). Knowledge organization in intelligent tutoring systems for diagnostic problem solving in complex dynamic domains. Submitted for publication.

26. Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*, Morgan Kaufmann Publishers, Los Altos, CA.

27. Woods, D.D. (1984). Visual momentum: a concept to improve the cognitive coupling of person and computer. *International Journal of Man-Machine Studies, vol. 21*, pp. 229-244.