Project No./(Center No.) E-24-684 R6324-0A0
Project Director: Dr. T. Govindaraj
Sponsor: Office of Naval Research, Department of the Navy

Agreement No.: SFRC No. N00014-87-K-0482
Award Period: From 6/1/87 To 5/31/90 (Performance) 7/31/90 Reports
Sponsor Amount:
Contract Value: $ 133,284
Funded: $ 111,284 (through 11/30/87)

Cost Sharing No./(Center No.)
Cost Sharing: $ 133,284 (through 11/30/87)

Title: Qualitative Simulation and Intelligent Tutoring Aids for Training in the Operation of Complex Dynamic Systems.

RESTRICTIONS
See Attached Government Supplemental Information Sheet for Additional Requirements.

Travel: Foreign travel must have prior approval — Contact OCA in each case. Domestic travel requires sponsor approval where total will exceed greater of $500 or 125% of approved proposal budget category.

Equipment: Title vests with GIT, provided prior written approval received from ACO, or included in proposal budget.

COMMENTS:

COPIES TO:
GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 04/18/91

Project No. E-24-684
Center No. R6324-0A0
Project Director GOVINDARAJ T
School/Lab ISYE
Sponsor NAVY/OFC OF NAVAL RESEARCH
Contract/Grant No. N00014-87-K-0482
Contract Entity GTRC
Prime Contract No.
Title QUAL SIMULATION & INTELLIGENT TUTORING AIDS TRAINING IN OPERATION OF COMP
Effective Completion Date 901231 (Performance) 910228 (Reports)

Closeout Actions Required:  Y/N Submitted

- Final Invoice or Copy of Final Invoice Y
- Final Report of Inventions and/or Subcontracts Y
- Government Property Inventory & Related Certificate Y
- Classified Material Certificate N
- Release and Assignment Y
- Other N

Comments

Subproject Under Main Project No.
Continues Project No.

Distribution Required:

- Project Director Y
- Administrative Network Representative Y
- GTRI Accounting/Grants and Contracts Y
- Procurement/Supply Services Y
- Research Property Management Y
- Research Security Services N
- Reports Coordinator (OCA) Y
- GTRC Y
- Project File Y
- Other N

NOTE: Final Patent Questionnaire sent to PDPI.
2 January 1989

Dr. Susan E. Chipman
Program Manager, Cognitive Science
Code 1142CS
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000

Dear Susan:

Enclosed please find six copies of the progress report on my ONR Contract (N00014-87-K-0482) for the period September 1988 - November 1988.

Sincerely yours,

T. Govindaraj

Associate Professor of
Industrial and Systems Engineering

Enclosure
Qualitative Simulation and Intelligent Tutoring Aids for Training
in the Operation of Complex Dynamic Systems

T. Govindaraj, Principal Investigator

Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205
E-mail: tg@chmsr.gatech.edu
(404) 894 3873

2 January 1988

Progress Report for September 1988 - November 1988

Submitted to

Dr. Susan E. Chipman
Program Manager, Cognitive Science
Office of Naval Research, Manpower R&D Program

Contract Number: N00014-87-K-0482
Implementation of the marine power plant simulator in Allegro Common Lisp on Apple Macintosh II computers was completed during the sixth quarter. In addition to the simulator development, graphical, menu-based interfaces were refined and representative examples were implemented for the simulator.

As reported in the previous progress reports, the power plant simulator and the interface have been developed using objects defined in Common Lisp Object System (CLOS). While the conceptual development of the direct manipulation interfaces was done in a relatively brief period of time, completion of the interfaces for all the required subsystems and components is taking some time. The simulation involves approximately a hundred components and their interconnections; the interface must make a large number of components accessible via gauges and controls. The object-oriented approach to the design of the simulation and the interface development has enabled us to develop a unified, streamlined approach to the development of the power plant simulator.

We had hoped to complete the implementation of the simulator, including the graphical interfaces, by the end of December 1988. While the interface to the entire system was not completed, the conceptual details have been worked out and appropriate data structures and objects have been designed. The entire interface, and hence the complete simulation, should be ready by the middle of January 1989. In parallel with the simulator implementation, we have been developing a tutor based on our previous research and our most recent work on an expert model. Features of the tutor will be implemented soon. A preliminary version of the tutor is expected to be operational by the end of March 1989.

The complexity of the steam power plant and its size characterized by the large number of interacting components has led to some unanticipated delays in converting the original simulation. This conversion from Interlisp-D on the Xerox Lisp machines to Common Lisp on the Macintosh II was necessary for the development and implementation of an intelligent tutor to provide reasonable response to operator actions. In a recent informal comparison, we found that the simulator is more than an order of magnitude faster on the Macintosh compared to the Xerox 1109. Since the complexity and computational demands of the tutor will continue to increase, the simulator reimplementation using objects in Common Lisp on a readily available personal computer workstation should enable us to concentrate on improving the tutor architectures. With increased availability of accelerator boards and other accessories for the Macintosh II and the
availability of Allegro Common Lisp on more powerful Sun and NeXT workstations, our future research on the development of intelligent tutors should proceed at a faster pace than before. The newer marine power plant simulator should provide a strong foundation for the design of more comprehensive and powerful tutors without the limitations of computational power.
2 October 1989

Dr. Susan E. Chipman  
Program Manager, Cognitive Science  
Office of Naval Research  
800 N. Quincy Street  
Arlington, VA 22217-5000

Dear Susan:

Enclosed please find twenty copies of the progress report on my ONR Contract (N00014-87-K-0482) for the period June 1988 - August 1989.

Sincerely yours,

T. Govindaraj  
Associate Professor of  
Industrial and Systems Engineering

Enclosures
Qualitative Simulation and Intelligent Tutoring Aids for Training
in the Operation of Complex Dynamic Systems

T. Govindaraj, Principal Investigator

Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205
E-mail: tg@chmsr.gatech.edu
(404) 894 3873

2 October 1989


Submitted to

Dr. Susan E. Chipman
Program Manager, Cognitive Science
Office of Naval Research, Manpower R&D Program
Contract Number: N00014-87-K-0482
In this report we describe the progress of our research on the development of an intelligent tutoring system for the marine power plant simulator during the second phase of the three-year contract. Under the current ONR contract, we have three major objectives: 1) to develop an architecture for intelligent tutoring systems for diagnostic problem solving in the supervisory control operation of complex dynamic systems; 2) to implement the tutor for the marine power plant simulator; and 3) to develop and evaluate a training program using the tutor implementation. The research described here concentrates on items (1) and (2), i.e., design, development, and implementation of the ITS. An architecture for the tutor has been developed and a substantial portion of the tutor has been implemented. Progress during the period under review is described below.

At the end of the first year, a small scale preliminary model of expert's fault diagnosis task was developed for incorporation into the tutoring system. This model was based on ideas proposed in an earlier study conducted during our previous ONR contract where data and protocols were collected from experts performing the troubleshooting task on the simulator of the marine power plant. A paper describing the details of this model was presented at the Sixth Symposium on Empirical Foundations of Information and Software Sciences (EFISS) in October 1988. This model evolved into the model of expert described below. Detailed descriptions of the expert model are given in Appendices A and B.

During the second year of the contract, the process of converting the marine power plant simulator into Allegro Common Lisp on Apple Macintosh II computers was completed. The simulator design is based on a qualitative approximation methodology in which moderate amounts of computing power is sufficient to simulate complex dynamic systems. A hierarchical approach is used to represent the system structure, and, the system states are represented qualitatively to provide cognitive compatibility with the operators in training. Common Lisp Object System (CLOS) is used to represent all important entities as objects, including knowledge of the domain and troubleshooting knowledge. A control interface can be added to the simulator to facilitate operator actions such as changing the operating conditions. The rest of this progress report describes the key features and implementation details of the tutor.

A major component of our research effort during this reporting period has been concerned with the design and implementation of the intelligent tutoring systems (ITS). The
complete ITS, implemented in Allegro Common Lisp to run on Apple Macintosh II computers, is comprised of the simulator, the tutor, and simulator and tutorial interfaces. An expert module, a student module, and an instructional module constitute the tutor. Mouse-based direct manipulation graphical interfaces are used for the simulator and tutorial interfaces. These interfaces play a major role in training, together with appropriate means of knowledge organization that support the key elements of the tutor. Important characteristics of the knowledge organization are described in this progress report.

Knowledge required for the design and implementation of an intelligent tutoring system for diagnostic problem solving has four components. These components are: (1) appropriately structured domain knowledge, (2) knowledge of troubleshooting strategies, (3) knowledge to infer a student's possible intent and misconceptions from observed actions, and (4) knowledge of tutoring goals and the means to realize the goals. The domain knowledge and troubleshooting strategies constitute an expert model of the operator's task. The instructional module uses this model to train students to use proper diagnostic problem solving strategies. Knowledge of student actions can help the instructional module to infer possible misconceptions that students may have. Finally, knowledge of tutoring goals and how they are realized guides the instruction and its communication.

The domain knowledge is represented in multiple, but complementary, views of the system's structure, function, and behavior. The representations used are the schematics, the functional subsystems, and the fluid paths. The troubleshooting knowledge is a combination of system knowledge and diagnostic strategies, including general knowledge of failures and cause-effect relationships about commonly occurring failures. Results from our own past research as well as those of others are used to develop instructional strategies and means for inferring student intent and misconceptions.

Detailed information on the tutor can be found in the working paper included as Appendix A. Highlights of the tutor are given in Vasandani et al. [1989] (Appendix B).

Even though we had hoped to complete the tutor development and implementation by the end of the second year of the contract, the complexity of the domain and the size of the Lisp code hampered our progress. Also, it was not possible to find additional students knowledgeable in the principles and operation of power plants who are also proficient in
Lisp and knowledge representation methodologies. However, the simulation and the tutor that have been implemented are very robust and modular, making it easy to maintain the code and add functions and features in the future.

In summary, the design of the tutoring system has been completed. A substantial portion of the system has also been implemented, including the graphical interfaces. We expect to complete the implementation and preliminary design of a training program that uses the tutor by the end of December 1989. The training program will be refined by conducting pilot experiments. We plan to start conducting experiments to evaluate the tutor and the training program in the first quarter of 1990.
Appendices A and B

A. Working paper

B. IEEE SMC Conference Proceedings

"An Intelligent Tutor for Diagnostic Problem Solving in Complex Dynamic Systems"
An Intelligent Tutor for Diagnostic Problem Solving in Complex Dynamic Systems

Vijay Vasandani, T. Govindaraj, C.M. Mitchell

Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Diagnostic problem solving is a major activity in the supervisory control of complex dynamic systems. Successful fault diagnosis in such systems depends upon the operator's knowledge of the functional properties of the system and timely compilation, integration and application of this system knowledge. An intelligent tutoring system (ITS) that helps to organize system knowledge and operational information including symptom-cause relationships may enhance competent operator performance. The ITS should incorporate the structure, function and behavior of the controlled system in an appropriate form that achieves cognitive compatibility with the operators. In addition, the ITS must contain a normative task model that provides it with the ability to infer the trainee's misconceptions. This paper proposes a methodology for organizing system and task knowledge in an intelligent tutor to be used for training operators to troubleshoot large dynamic systems.

Introduction

Supervisory control of complex dynamic systems requires monitoring, planning and other problem solving skills (Rasmussen, 1986; Woods, 1986; Wickens, 1984; Rouse, 1982; Sheridan and Johannsen, 1976). Among the various tasks of a supervisory controller, fault diagnosis forms a major part of the operator's problem solving activity.

Fault diagnosis in complex dynamic domains depends upon the operator's use of system knowledge at multiple levels of abstraction and detail (Rasmussen, 1985). Efficiency in diagnostic problem solving is enhanced by timely compilation, integration and organization of appropriate pieces of operational information about the components and the system. Even when operators are familiar with system operation, they are sometimes unable to combine symptom information with mental resources concerning system knowledge during troubleshooting (Govindaraj, 1988). Operators need to be trained to overcome the problems related to the cognitive aspects of diagnostic problem solving. An operator training program that helps organize system knowledge and operational information including symptom-cause relationships is therefore essential to ensure competent performance.

Training for diagnostic problem solving task can either be provided on-the-job or on simulators. On-the-job training has many problems. First, it is usually very expensive. Second, the consequences of an error can be catastrophic. Finally, malfunctions occur infrequently and it may be undesirable or impossible to duplicate them during the training period. Systems that can simulate a wide range of failure conditions offer a good alternative training environment. However, simulators by themselves are
unable to provide appropriate help since they do not have the ability to evaluate a student's misconception from observed actions. A simulator coupled with an intelligent tutor may, however, improve training effectiveness. Such a combination of simulator and tutor constitutes an intelligent tutoring system (ITS). Research on ITS in the past has been directed more towards computer-based training for relatively simple and procedural tasks. Applications in training operators of engineering systems have been scarce. Some tutors that have addressed engineering domains are discussed in the next section.

Following the discussion of existing intelligent tutors for complex systems, this paper proposes an architecture and a methodology for organizing knowledge in such tutors. A framework for evaluating a student's misconceptions based on observed actions is also described. The paper concludes with a description of an application of the proposed architecture.

Intelligent tutoring in complex systems

Although work on intelligent tutoring systems has been in progress for over two decades, computer power and developments in ITS research have not been sufficiently harnessed for application in complex, dynamic engineering domains. Sleeman and Brown (1982), Wenger (1987) and Psotka et al. (1988) provide an extensive survey of existing ITSs but only a few deal with engineering domains. STEAMER (Hollan et al., 1984), IMTS (Munro et al., 1988), AHAB (Fath, 1987; Fath et al., 1989), The Recovery Boiler Tutor (Woolf et al., 1986), SOPHIE (Brown et al., 1982) and SHERLOCK (Lesgold et al., 1988) are some examples that have useful applications in an operator training program. Each of these has a system simulator which provides an environment that lets the student visualize the domain and practice tasks.

A problem with many training environments is that they use steady state values to describe the state of the system. Steady state values provide adequate system information to diagnose faults in many domains. However, in many other domains, such as a power plant, the operator has to begin investigating a failure well before the system attains a steady state, and sometimes there may not even exist a steady state. Thus, the dynamics of a complex system demands that new operator training techniques be explored.

In general, intelligent tutoring systems have an expert module, a student module, and an instructional module (Sleeman and Brown, 1982; Wenger, 1987; and Psotka et al., 1988). In addition, a simulator provides the training environment. The expert module of an ITS 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 competence. 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 the ITS (Wenger, 1987; Psotka et al., 1988).

In the next section, an architecture for intelligent tutoring systems is proposed along with a discussion of the major requirements of instructional systems.

The Architecture

The research described here utilizes artificial intelligence (AI) tools and tutoring methodologies to develop an instructional program for operators of complex dynamic systems. Figure 1 illustrates the major components of such an instructional system. 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
The instructional system has two major requirements: (1) a domain simulator and (2) organization of knowledge that supports the functions of the three major elements of the tutoring system.

Discussion of the two requirements of an instructional system appears next beginning with a brief description of the simulation methodology.

**Simulation Methodology**

Development and successful implementation of effective computer aids depends upon the availability of a simulator constructed from a methodology compatible with human cognition. A simulator designed via qualitative techniques has better cognitive compatibility with operators under training than ones designed from analytical techniques. One such qualitative simulation technique is the "Qualitative Approximation Methodology".

Qualitative approximation (Govindaraj, 1987) provides a convenient, practical means of designing simulators of complex dynamic systems. Qualitative approximation uses a combination of bottom-up and top-down approaches to model system dynamics. In qualitative approximation, system dynamics are represented hierarchically with primitives that approximate the functions of the components at the lowest level. The primitives provide qualitative states that describe the evolution of the system. Using this methodology, large systems can be simulated with a moderate amount of computational power due to reduced computational requirements. QSTEAM, an application of qualitative approximation techniques, simulates the dynamics of a marine power plant under a number of failure situations and forms an integral part of our instructional system.

While the simulation methodology for an instructional system was discussed here, appropriate knowledge organization, the other important issue in the design of such systems, is addressed next.

**Knowledge Organization**

Successful implementation of an intelligent tutor for diagnostic problem solving in complex dynamic domains depends upon the availability of (1) appropriately structured domain knowledge, (2) knowledge of troubleshooting strategies, (3) knowledge to infer a student's possible intent and perhaps misconceptions from observed actions and (4) knowledge of tutoring goals and how they may be realized. The domain knowledge and troubleshooting strategies constitute an expert model of the operator's task. 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 student misconceptions. Finally, knowledge of tutoring goals and how they are realized guides the instruction and its communication.

Requirements and features of the tutoring knowledge are described next. The structure of the domain knowledge is discussed first, followed by a discussion of the troubleshooting task knowledge, normative model of the students' troubleshooting actions, and general instructional strategies.

**Domain Knowledge**

Successful fault diagnosis in complex dynamic domains is aided by multiple representations of the system's functional properties (Rasmussen 1983). The expert in a tutor for a diagnostic problem solving task must therefore have access to multiple
representations of the system knowledge. **Schematics, functional subsystems and fluid paths** are the three possible means of representing the system knowledge. A **schematic** is a pictorial representation of the components in the system. Schematics often graphically represent subsystems and fluid paths in a system. A **functional subsystem** is a collection of components responsible for performing a higher level system function. **Fluid paths** are a way of visualizing the system as a collection of different fluids. Thus, the three representations of the system knowledge are complementary rather than mutually exclusive. A detailed description of system knowledge decomposition as suggested above is provided next.

**Schematics:** A schematic presents a view into the structure of the system. Typically, a schematic shows the sequence in which certain components and the gauges appear in a real system. It is also a structure that reveals the logical proximity of two physically unconnected components such as the burner and the stack in a combustion unit. A configuration of all components either responsible for a higher level function or sharing a common fluid is yet another example of a schematic.

In diagnostic problem solving tasks on a simulator, schematics are typically used to view the configuration of components and gauges. Scanning through the various schematics permits an operator to visualize the sequence of system processes as they occur in the system. In a steam power plant, for example, the schematics may display the stages of power generation in a sequence starting with the combustion of fuel, followed by steam generation, steam condensation and preheating of condensed steam for re-use in a closed loop water circuit. A collection of schematics provides a convenient interface between the simulated system and the operator. The operator's interaction with the system during a troubleshooting task involves probing gauge readings in the suspected areas of failure through schematics.

Grouping of components in schematics for a tutoring system depends upon some other factors such as frequency of interaction and level of dependency. There are portions of a system that commonly interact with each other. For instance, in a power plant, the performance of steam generation unit is affected by the performance of the combustion unit. Hence, the steam generation unit and the combustion unit are displayed in a single schematic. There are parts of a system which do not significantly affect other portions of the system and thus are viewed in isolation. For example, problems related to lubrication are usually confined to lube oil path and rarely affect other fluid paths, unless left unattended for a long time. Finally, there are some failures in a system that occur more frequently than others. Components and gauges required for investigating such failures are confined, as far as possible, to a single schematic.

**Functional subsystems:** Functional subsystems are collections of components responsible for achieving specific higher level system functions. There are several higher level system functions that collectively contribute to the system goals. For instance, in a marine power plant, the functions are combustion, steam-generation, power-generation, steam-condensation, feed-water-preheating, auxiliary-steam-use, saltwater-service, lubrication and control-air-distribution. A functional subsystem is described by information related to (1) fluid paths passing through the subsystem; (2) components through which a given fluid flows; (3) the order in which the components and gauges appear in each fluid path; (4) the connected subsystem on either side of the fluid path; and (5) the schematic in which the subsystem may be found.

**Fluid paths:** In decomposing a system by fluid paths, all components on the same fluid path are represented in a group. Additional system knowledge based on fluid paths consists of (1) schematics in which the fluid is found, and (2) the subsystems through which
the fluid flows. Examples of fluid paths in steam power plants are combustion-air, fuel-oil, flue-gas, superheated-steam, desuperheated-steam, feed-water, condensate, main-condenser-hot-fluid, main-condenser-cold-fluid, saltwater, lube-oil and control-air.

Each of the three system representations described above involves mechanical components and gauges. The lowest level of system knowledge description is hence at the component level. System knowledge at the component level has three attributes: structure, function and behavior. A component's structure, for the most part, refers to its connections to other components on the input and output side, the fluids carried by it, the gauges attached to it, and its association to a schematic or a functional subsystem. Structural changes in the components are usually responsible for abnormal behavior of the system. Therefore, the component level structural description for the failed and normal modes of a component are different. Functional knowledge about a component is its intended use in the system and its contribution to the higher level functions of the system. Behavioral knowledge of a component concerns its states. Since the behavior of a component is different under normal and failed modes, the behavioral knowledge, like the structural knowledge, is different for the two modes. Together, the structural, functional and behavioral knowledge of a system and its components form an essential part of the expert's knowledge. Structural, functional and behavioral knowledge are discussed below.

Structural Knowledge: Most of the structural information for components is the same in normal and failed states. The structural information that remains invariant after a failure includes its connectivity relationship to other components, the fluids flowing through it, and its association to a particular subsystem and schematic. When a component fails, some structural information changes. For example, a valve with its control set to the open position but its blade stuck in the closed position represents a structural change for a valve when it is blocked shut. Such structural changes for failed components will be discussed later as a part of "Troubleshooting knowledge".

Functional Knowledge: Functional information defines the purpose or role of a component in the system. Functional knowledge of a component depends upon its structure. For example, a pipe in the system may be modeled as a conduit, where the function of a conduit is to transport moving fluid from one of its ends to another. In an approximate representation, where friction may be ignored, it is reasonable to define the function of the conduit in the manner described above. In general, a large number of primitive function types, like the conduit, can be identified for a system. All the components of the system can be categorized as instances of one of the primitive types. For continuous systems, examples of primitives based on functions include sink, source, source-sink, gain, controller, reactor, transducer, heat-exchanger and phase-changer.

Behavioral Knowledge: Normal and failed modes of a component affect the system differently. The manner in which the system state values are affected by the presence of a component, in both the normal and the failed states constitutes the component's behavioral knowledge.

Normal behavior of components is responsible for normal state values during system operation. For example, normal behavior of the main condenser is responsible for a lower outlet temperature of the hot medium as compared to its inlet temperature. As the hot medium moves from inlet to outlet it undergoes a phase change from gas to liquid. The same normal behavior of the main condenser is also responsible for a corresponding increase in temperature of the cold medium as it flows from its inlet to outlet port. Behavior of all components can be explained by the laws of science, e.g., the law of conservation of energy explains the normal behavior described here.
Abnormal behavior describes the manner in which certain state values are affected by a failure in the component. For tutoring, the behavioral information for a failed component includes contextual information about specific gauges affected by the failure. The explanations for the abnormal gauge readings in terms of cause-effect relationships also form a part of the component’s behavioral knowledge represented in the tutor. Further details of behavioral knowledge of failed components are discussed in the next section.

Domain knowledge, although essential, is not sufficient for the troubleshooting task. Troubleshooting knowledge discussed next includes more than the operational knowledge of the system and its components.

Troubleshooting knowledge

Troubleshooting knowledge combines system knowledge and diagnostic strategies. Troubleshooting knowledge includes general knowledge of the types of failures in the system, detailed information on certain common failures, and cause-effect associations for familiar failures. In this section we discuss the nature of the diagnostic problem solving knowledge.

A mechanical component in a physical system such as a steam power plant can fail in more than one way. There are five common modes of failure in components: (a) blocked-shut, (b) stuck-open, (c) leak-in, (d) leak-out, and (e) reduced-thermal-efficiency (Fath et al., 1989). Faults in components fit one or more of these five mode types. Not all components, however, fail in all five different ways. Some components have multiple faults that fit the same failure mode category. For example, a clogged valve or a valve stuck in closed position are two different ways in which the valve may be blocked-shut. In some other components, the same fault may cause multiple modes of failure. A clogged strainer in a condenser’s cold water path, for example, fits both blocked-shut as well as reduced-thermal-efficiency modes.

Each failure mode exhibits a typical system behavior (Fath, 1987; Fath et al., 1989). The typicality of such behavior provides useful diagnostic information. If the system behavior suggests a particular mode of failure, then the list of suspected components can be reduced to those that fail in that particular mode. The typical system behavior may depend upon the phase of the fluid in the affected path. A blocked-shut mode of failure in a liquid path, for example, causes the liquid level downstream to be lower than normal and the level upstream higher than normal. A similar blocked-shut failure in a gas path, on the other hand, decreases the downstream gas pressure and increases the upstream pressure. In any case, system behavior associated with each mode is manifested in the form of a typical pattern of abnormal state values. Patterns of such abnormal state values can be determined by the application of the laws of physics and thermodynamics, and recognizing these patterns of abnormalities during fault diagnosis often helps to identify the type of failure in the system.

System behavior associated with failure mode sometimes deviates from the expected abnormal behavior (Fath, 1987; Fath et al., 1989). The way in which the system components are configured is often responsible for such a deviation. For instance, a source-sink such as a deaerating-feed-tank located downstream in the blocked-shut feedwater-path may prevent further propagation of low feed-water level. The deaerating-feed-tank imposes such a behavior on the system because it is an “infinite” source of feed-water which can at least temporarily compensate for any loss in the water level. The expected abnormal behavior associated with a mode of failure may therefore be confined to the vicinity of the failed component. Furthermore, with the limited availability of gauges around the failed component, the abnormal behavior may not be observable. Knowledge of
such deviations from the norm is essential for correct identification of the type of failure in the system.

Even when the failure mode is recognized from the system behavior, it may not be very useful. An expert needs more than just the knowledge about modes of failure and their associated system behavior. However, when the expert's troubleshooting knowledge also includes information on all possible modes of failure for each component, it can be helpful in at least reducing the list of suspected components. For this reason, it is important to categorize all component faults under the different failure mode types.

Finally, to isolate the failed from the suspected components and to diagnose the fault, additional information such as the gauges affected by the failure and causal relationship between abnormal system states for every fault is required. Knowledge of the affected gauges and the system states for the individual faults can provide the verification of the final diagnosis.

There are other elements of the troubleshooting knowledge, accumulated through experience, that make fault diagnosis in a large complex system time efficient (Govindaraj and Su, 1988). This experiential knowledge, based on prior cases of solved and unsolved problems encountered by the operator, is usually responsible for the formation and rapid refinement of an initial set of hypotheses of either suspected components, subsystems, or fluid paths.

Experiential knowledge is activated by the observation of obvious and non-obvious (i.e. discovered only upon investigation) symptoms. In a complex dynamic system, the size of the system and the effects of fault propagation make it impossible to uniquely associate a symptom to a specific fault. However, in such systems, observable symptoms still help to limit the search for the failed component to a specific location in the system. For example, the symptoms may indicate that a particular higher level function of the system has been affected by the fault. This helps to confine the search for the failed component to components comprising the subsystem responsible for the affected function. Symptoms may further help to categorize the faults, for example, it may separate those related to components with moving parts from those related to speed or load. Such a categorization of failures further reduces the search space for a failed component. For example, a search space generated by a set of all components with moving parts in the combustion system of a power plant is likely to be much smaller than the set of all components in the combustion subsystem.

An operator's fault diagnosis task is also aided by inferences based on failure schemas built through experience. These failure schemas are a part of experiential knowledge. The schemas represent some of the familiar ways in which the system fails. A schema is activated by a symptom and proposes a hypothesis or a partial solution to the diagnostic problem. The partial solution may be a diagnostic test that either provides a conclusive inference or activates another schema. For example, smoke in a boiler may activate a schema that recommends checking for smoke color. Black smoke may then trigger an incomplete-combustion schema while a white smoke may trigger an excessive-air-in-the-burner schema. An abnormal fuel temperature with black smoke in the boiler may prompt the incomplete-combustion schema to specify desuperheated-steam or fuel path as the path suspected of containing the failed component.

Rasmussen (1986) has characterized the application of the troubleshooting task knowledge into two diagnostic strategies: symptomatic and topographic search. Symptomatic search is a simple and economical pattern matching strategy where a successful association between cause and effect is generated based on prior experience. An
unsuccessful attempt with symptomatic search usually leads to topographic search. In topographic search, a hypothesis about the failed component is generated and tested by comparing a model of normal behavior of the suspected component with its behavior in the abnormally functioning system. Neither the symptomatic nor the topographic strategy is adequate in itself; instead, an expert often has to switch between the two strategies many times to complete the task.

The discussion in this section provided an overview of an expert's troubleshooting knowledge and his diagnostic strategies. The system and the troubleshooting task knowledge discussed thus far are also normally the representation of the material to be taught by the tutor. Interestingly however, the knowledge representation that is suitable for expert performance is not necessarily suitable for instruction or for evaluating student's misconceptions (Clancey, 1987). A normative model of the student's actions is an alternative organization of the expert's task knowledge that may help evaluate a student's misconceptions.

**Normative model of the student's actions**

An important feature of an intelligent tutor is its ability to evaluate a student's misconceptions. This capability of the tutor evolves from a normative model of the student's actions described below.

In a normative model of the student's actions, not all actions that occur at the student-tutor interface are valid. Examples of valid actions may range from requests for help to responses to queries and calls for schematics. In addition, in diagnostic problem solving, there may be some other actions performed by the student. These actions may include investigating components for gauges and checking their gauge readings. An action to investigate a component may be called an *investigative action* and a request to display the value of a particular gauge attached to the component an *informative action*. Most of the student's actions, such as the request for help, response to query, call for a change in schematic display and even investigative actions are self explanatory. These actions clearly express the intent of a well-motivated learner interacting with the tutor. However, the informative actions taken during diagnostic problem solving are associated with ambiguity concerning student's intent. We need context-specific knowledge and an understanding of the cognitive aspects of troubleshooting task to resolve these ambiguities.

In a troubleshooting task, the student maintains a set of failure hypotheses that explain the abnormal behavior of the system (Fath 1987; Fath et al., 1989). A set of hypotheses is a list of components suspected to have failed. Each informative action taken by the student is an attempt to reduce the size of the set of failure hypotheses. The manner in which the list of suspected components may be revised depends upon the outcome of the diagnostic test associated with the informative action. The test results have a context-specific significance. For example, in a power plant, if the student has been alerted by a low condensate pressure alarm, it makes sense for him to check the pressure gauge on the condensate pump. If he does check the pressure gauge on the condensate pump, it is reasonable to assume that the condensate pump is probably one of the suspected components. If the pressure gauge shows a low reading, the student has reason to continue suspecting a malfunction in the condensate pump. On the other hand, if the pressure gauge reading is normal, the condensate pump may be omitted from the list of suspected components. However, when the student is alerted to a failure in the system by smoke in the boiler rather than a low condensate pressure alarm, checking for pressure across the condensate pump is inconsistent with the failure data. Thus, the knowledge of what are reasonable actions under various failure situations and how the test results ought to refine the set of failure hypotheses can help in evaluating the student's misconceptions.
A normative model of student’s actions is, therefore, a model that describes the valid actions of a student for each failure condition and can be used to evaluate students' misconceptions. The knowledge required to evaluate misconceptions using the normative model is described next.

**Evaluation of misconceptions**

The normative model describes what a student ought to do under a particular failure situation. When the student’s action does not match actions suggested by the normative model, the reason can be attributed to many causes. Usually the causes are related to lack of knowledge, inappropriate knowledge or deficiencies in knowledge application skills. Evaluating a student’s misconception means determining the probable cause for the deviant behavior. While suggesting remedies may be relatively straightforward when misconceptions are known with certainty, determining the misconception itself is a difficult task. The research reported here proposes an inference framework to evaluate misconceptions in students learning a troubleshooting task.

In order to determine a student’s misconception, the tutor needs to know the types of misconceptions that are associated with incomplete knowledge of the system or the task. Our method of organizing the knowledge in the expert module is also helpful in organizing categories of misconceptions. Misconceptions can be categorized as those related to lack of (1) structural knowledge of the system, (2) functional knowledge of system and components, and (3) knowledge of system behavior resulting from failures. Lack of system structural knowledge makes the student investigate portions of the system unrelated to the failure. Checking components and gauges in the fluid paths unaffected by failure indicates a lack of understanding of different system functions and their inter-relationships. Finally, pursuing a hypothesis that should have been rejected based on evidence gathered, or premature elimination of suspicion from a component due to insufficient evidence, suggests shortcomings in behavioral knowledge related to failures.

After evaluating a student's misconception, an intelligent tutor is also responsible for generating instructions to rectify the misconception and to improve the student's diagnostic problem solving skills. The selection of appropriate sets of instructions and their presentation is guided by strategies some of which are discussed next.

**Instructional strategies**

The instructional module of an ITS contains knowledge that specifies how the tutor should respond to various student actions. Many of the instructional modules rely on a rule-based structure to create instructions (e.g., Burton and Brown, 1982; Clancey, 1987). More recently, Woolf (1984) and Macmillan et al. (1988) have proposed architectures for dynamic instructional planners in adaptive environments. However, in any architecture, the key issues to be addressed are the instructional content, its form and time of presentation.

Instructional content depends upon the instructional objectives. Several units of instructions may be available that satisfy these objectives. Selection of a particular unit of instruction is governed by instructional strategies chosen for the tutor. Such strategies may, under different situations, include preference for hints or discussion of generalities as opposed to solutions or discussion of specifics, preference for graphical instead of textual mode of presentation, and preference for presenting material with or without intervention. Instructions without intervention can only be provided at the end of a training session. While non-intervention has some advantages because it does not disturb the student's
thought process, intervention at critical stages of diagnostic activity may be an effective way of emphasizing a point. With respect to tutorial intervention, both the model tracing approach (Anderson et al., 1985) which calls for intervention as soon as the student's observed actions stray from the normative actions and the issue-based tutoring (Burton and Brown, 1982) which encourages intervention at particular occasions can be usefully implemented. Furthermore, having a control structure for instruction selection that is flexible enough to be manipulated for different contexts is desirable.

The tutoring system architecture and the knowledge organization proposed in this paper have been integrated into a tutoring system. Implementation details of this system are described next.

Implementation

To evaluate the proposed architecture and the knowledge organization methodology, a tutoring system is being implemented on an Apple Macintosh II computer in Allegro Common Lisp with object-oriented extensions. This section provides an introduction to the training environment and discusses representation of system and troubleshooting task knowledge, diagnostic strategies, normative model of student's actions, and instructional strategies.

The training environment

The simulator, QSTEAM, is an application of qualitative approximation techniques which simulates the dynamics of a marine power plant under a number of failure situations. QSTEAM also provides an environment for training operators to troubleshoot the marine power plant. Its interface with the trainee has a fixed menu displaying icons representing the various schematics in QSTEAM. A schematic can be called by clicking on the icon associated with the schematic. There are seven schematics: steam, boiler, feed-water, fuel-oil, control-air, saltwater and lube-oil. Together, the seven schematics provide a comprehensive, yet economical, layout of the marine power plant (Figure 2).

In the schematics of QSTEAM, all the major stages of power generation and fluid paths can be distinctly identified. Configuration of components responsible for a higher level system function is mostly confined to a single schematic. Where a functional subsystem does span over multiple schematics, related schematics can be accessed from each other. To move between these related schematics, one clicks on an icon representing the related schematic. Such icons are found at one or more locations where the subsystem has its connections broken. In moving from one schematic to another, visual momentum (Woods, 1984) is essential and is provided by (a) high lighting an icon in the new schematic to establish orientation with respect to the old schematic; (b) making it possible to switch back to the old schematic by clicking on the high lighted icon; and (c) providing a smooth visual transition between schematics, as if the schematics are portions of one big schematic and moving from one schematic to another is equivalent to scrolling. For example, broken connections which are on the right edge of a schematic always join the connected schematics from the left side and vice versa. In addition, representation of a component in multiple schematics is indicated on the schematic by icons adjacent to such components. These icons mark regions from which the schematics they represent can be accessed.

Since the student's primary interaction with the system is via probing gauge readings in the suspected areas of failure, the student is provided the facility to investigate gauges attached to any of the components displayed in the current schematic. A mouse
click on a component displays all the gauges attached to it, including their types (e.g., pressure, temperature or flow) and locations along the fluid paths. Clicking on any of the displayed gauges shows the gauge reading. At any instant, only the most recently investigated component has its gauges and the observed gauge readings displayed in a schematic. However, if the student wishes to keep a gauge and its reading locked on a schematic for future reference, it is possible to do so.

Dialog boxes and menus are used to facilitate other valid interactions with the system. Student's interactions with the system are monitored to evaluate misconceptions and provide instructions. Details of the process of evaluating misconceptions and presenting the instructions are discussed later. The next section describes how the knowledge about the system and the troubleshooting task are represented using the method of knowledge organization proposed in the paper.

**Knowledge representation**

The method of knowledge representation in the tutor uses objects to represent knowledge structures. This approach is supported by an object-oriented environment that facilitates the creation and manipulation of abstract data types without replication of code. For example, components in power plant that are instances of the same functional primitive have similar object representations and share the same methods that create and manipulate them. Examples of objects used for representing knowledge are described below.

**System knowledge**

(a) **Subsystems:** Table 1 shows how knowledge about functional subsystems in the marine power plant has been represented. The example shows an object belonging to a class subsystem, which stores information such as fluid paths in the subsystem, components within each of these fluid paths and schematics in which the subsystem is found.

(b) **Fluid paths:** Information related to fluid paths is stored in objects shown in Table 2. This information includes the schematics in which this fluid path may be viewed, the subsystems through which this fluid flows and the components that lie along its path.

(c) **Components:** A component is an instance of one of the simple or composite primitives. There are nine simple primitives and one composite primitive. Slots in the primitives contain the component's structural and functional details. Table 3 shows knowledge representation for a component which is an instance of a simple primitive.

**Troubleshooting task knowledge**

(a) **Symptomatic maps:** Symptomatic maps establish a mapping between the visual and audio alarms first observed after a failure and the category of failures. Failures in the system are categorized according to higher level functions of the system.

An object representation of symptomatic map for failures in a combustion subsystem, for example, stores all obvious symptoms that indicate a fault in a component belonging to the combustion subsystem. Smoke from the boiler is an example of a symptom that activates the symptomatic map for failures in combustion subsystem. Table 4 shows the structure of a symptomatic map.
The subcategories of failures also have symptomatic maps associated with them. These maps when activated, reduce the list of suspected components within the affected subsystem. Noise and vibration along with smoke from the boiler is an example of a symptom that activates a symptomatic map to reduce the list of suspected components in the combustion subsystem to only those which have moving parts.

(b) General failure schemas: A class of objects called schema is used to store information about familiar failures. Every instance of a schema proposes a diagnostic test, and based on the results of the test, a conclusion or the next step. The conclusion, if any, is the determination of the fluid path affected by the failure. If the test results do not support any hypothesis concerning a fluid path, they activate another schema instead. Many schemas are thus linked in a tree-like manner. Such a resulting tree of schemas provides an inference mechanism to identify the fluid path affected by the failure based on symptoms observed during the troubleshooting process. Table 5 gives a brief description of the variables in the object schema.

An example of an inference tree resulting from failure schemas related to combustion is shown in Figure 3. A possible way in which the schema represented at the top of the tree is activated is when smoke is noticed in the boiler. The diagnostic test suggested by the schema is to check for the color of the smoke. Black smoke triggers an incomplete-combustion schema and white smoke triggers an excessive-air-in-the-burner schema. Chaining forward or backward through the tree of schemas with each diagnostic test eventually leads to a set of suspected fluid paths.

(c) Component failure modes: The typical system behaviors associated with component failure modes are contained in the instances of failure-mode class of objects. There are five failure-mode objects, one for each of the five failure types—blocked-shut, stuck-open, leak-in, leak-out, and reduced-thermal-efficiency. Knowledge about a failure mode includes information such as the upstream and downstream system behavior along the affected fluid path and any deviation from the abnormal behavior due to any typical system characteristics. Table 6 shows how system behavior associated with failure modes is represented.

(d) Component faults: Knowledge of the individual faults in the component which includes the causes and symptoms associated with these faults, the failure mode, the upstream and downstream behavior, the gauges affected, and the cause-effect explanations for the observed system state abnormalities are stored in objects of class component-fault. A description of the slots in the object component-fault is provided in Table 7.

The next section describes how the normative model of student's behavior is represented and used to determine possible misconceptions in a student's knowledge.

Representation of the normative model and rules for evaluating misconceptions

The normative model of student behavior is also represented by objects. For each failure condition, the model has a list of affected components, fluid paths, gauges, and the schematics in which these affected system entities may be found. All student actions are recorded and compared with actions specified in the normative model for the failure situation being investigated. This comparison is made by the instructional module of the tutor. If differences appear, the tutor has a rule-based framework to classify the student deficiencies into three types of misconceptions resulting from lack of (1) structural knowledge of the system, (2) functional knowledge of the system and components, and (3) knowledge of system behavior associated with failure. Although it is impossible to completely understand a student's behavior, a context-specific guess is made using a set of
rules for evaluating the misconception. For instance, if a student's investigative actions
do not involve any of the schematics specified for the failure, the tutor interprets them as a
lack of knowledge of system structure. If, on the other hand, the student investigates
components, fluid paths or gauges that do not contribute to the process of problem solving,
the tutor deduces a problem related to lack of knowledge of the system functions and their
interactions. Sometimes, to determine the type of misconception, the tutor relies on
feedback from the student. For example, consider the case where redundant investigations
of a component are made even when conclusive evidence has already been gathered that
eliminates any possibility of failure in the component. Such a case indicates a deficiency
in the student's ability to relate suspected failures with system behaviors. However, before
taking any remedial measures, the tutor queries the student to ascertain the hypothesis
suggested by his actions.

An instructional system's tutoring goals can only be achieved if there are effective
interactions between the tutor and the student. Although effective interactions depend upon
the accuracy with which misconceptions are evaluated they also involve proper selection of
instructions, their form and time of presentation. The tutor's method of selecting
instructions and presenting them is described next.

Instructions

The instructional module of the tutor monitors and evaluates every action of the
student. If the student's action matches one of those specified by the normative model, the
tutor may compliment the student for his efforts. If not, the tutor invokes the rules for
evaluating misconceptions to determine the cause of the mismatch. If the cause cannot be
traced to any of the three pre-conceived types of misconceptions, no intervention by the tutor
is considered necessary. On the other hand, if the type of misconception is identified,
another rule-based structure containing instruction selection rules is activated to suggest a
general remedy that can perhaps rectify the misconception. General remedial procedures
for each type of misconception are encoded as consequents of the instruction selection
rules. Example of such a procedure may be show-fluid-paths. The instructional module
then manipulates the general remedies to prepare an instructional unit suitable for the
current context. For example, the general procedure show-fluid-paths may only be applied
to fluid paths in the context of the current failure. Finally, a presentation rule decides the
mode of presentation of instructions.

In addition, the instructional module provides the students with the facility to
request for help or hints and explore the expert's knowledge base through an instructional
menu. Using this menu, a student can check for the different modes of failure in a
component, its behavioral characteristics during failed states, the composition of
functional subsystems, feasible failure hypotheses at any time and the next best diagnostic
test to refine the set of hypotheses. Aid provided to the student in this manner is "student-
initiated" as opposed to "tutor-initiated".

Summary

Training for diagnostic problem solving is necessary for supervisory control of
complex dynamic domains. Difficulties in fault diagnosis are often due to the operator's
inability to properly organize, compile, integrate or apply system knowledge. Programs
that help organize system knowledge and diagnostic strategies can enhance operator
performance. ITSs having the ability to infer a trainee's misconceptions can provide better
help. A methodology for organizing system and task knowledge in intelligent tutors for
such training programs was proposed in this paper. A framework was presented for
developing a normative model of the student behavior that can be used for evaluating a
student's misconceptions. Finally, essential details for implementing the tutor were described.

Acknowledgements

This research is supported by contract N00014-87-K-0482 from the Manpower R&D Program, Office of Naval Research (Dr. Susan E. Chipman, contract monitor).

References


**Figure 1.** Instructional System Architecture
Figure 3. Combustion related tree of failure schemas
Object: subsystem

<table>
<thead>
<tr>
<th>variables</th>
<th>type</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>list</td>
<td>symbolic description of the function performed by the subsystem</td>
</tr>
<tr>
<td>in-schematics</td>
<td>list</td>
<td>schematics in which the subsystem is found</td>
</tr>
<tr>
<td>fluid-paths</td>
<td>list</td>
<td>fluids flowing through the subsystem</td>
</tr>
<tr>
<td>components</td>
<td>list</td>
<td>components in the subsystem</td>
</tr>
</tbody>
</table>

Table 1. Description of subsystem

Object: fluid-path

<table>
<thead>
<tr>
<th>variables</th>
<th>type</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-schematics</td>
<td>list</td>
<td>schematics in which the fluid is found</td>
</tr>
<tr>
<td>in-subsystems</td>
<td>list</td>
<td>subsystems through which the fluid flows</td>
</tr>
<tr>
<td>components</td>
<td>list</td>
<td>components in the fluid-path</td>
</tr>
</tbody>
</table>

Table 2. Description of fluid-path

Object: simple-primitive

<table>
<thead>
<tr>
<th>variables</th>
<th>type</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>symbol</td>
<td>name of the component</td>
</tr>
<tr>
<td>in-subsystems</td>
<td>list</td>
<td>subsystems to which the component belongs</td>
</tr>
<tr>
<td>in-schematics</td>
<td>list</td>
<td>schematics in which the component is seen</td>
</tr>
<tr>
<td>linked-graphic-objects</td>
<td>graphic-object-vector</td>
<td>an object that stores information on graphic representations of the component</td>
</tr>
<tr>
<td>input-gauge-list</td>
<td>list</td>
<td>gauges on the input side</td>
</tr>
<tr>
<td>output-gauge-list</td>
<td>list</td>
<td>gauges on the output side</td>
</tr>
<tr>
<td>input-components</td>
<td>list</td>
<td>names of components on the input side</td>
</tr>
<tr>
<td>output-components</td>
<td>list</td>
<td>names of components on the output side</td>
</tr>
<tr>
<td>input-from</td>
<td>state-vector</td>
<td>an object that describes each input connection and state values along that connection</td>
</tr>
<tr>
<td>output-to</td>
<td>state-vector</td>
<td>an object that describes each output connection and state values along that connection</td>
</tr>
<tr>
<td>fluid</td>
<td>list</td>
<td>fluids carried by the component</td>
</tr>
</tbody>
</table>

Table 3. Description of simple-primitive
Object: symptomatic-map

<table>
<thead>
<tr>
<th>variables</th>
<th>type</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>symptoms</td>
<td>list</td>
<td>initially observed symptoms</td>
</tr>
<tr>
<td>failure-category-type</td>
<td>symbol</td>
<td>the failure category</td>
</tr>
<tr>
<td>suspected-subsystem</td>
<td>symbol</td>
<td>name of suspected subsystem</td>
</tr>
<tr>
<td>suspected-components</td>
<td>list</td>
<td>name of all suspected components</td>
</tr>
<tr>
<td>failure-subcategories</td>
<td>list</td>
<td>symptomatic maps related to failure subcategories</td>
</tr>
</tbody>
</table>

Table 4. Description of symptomatic-map

Object: schema

<table>
<thead>
<tr>
<th>variables</th>
<th>type</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>diagnostic-test</td>
<td>list</td>
<td>fluid-path and state-variable to check in that path</td>
</tr>
<tr>
<td>conclusive-test-result</td>
<td>symbol</td>
<td>the qualitative state value reading, if any, for the diagnostic test</td>
</tr>
<tr>
<td>suspected-fluid-path</td>
<td>symbol</td>
<td>an inference based on test-result</td>
</tr>
<tr>
<td>next</td>
<td>symbol</td>
<td>name of next schema to activate</td>
</tr>
</tbody>
</table>

Table 5. Description of schema

Object: failure-mode

<table>
<thead>
<tr>
<th>variables</th>
<th>type</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>gas</td>
<td>behavior</td>
<td>for the failure-mode, an object of class behavior describes the system behavior in the gas path</td>
</tr>
<tr>
<td>liquid</td>
<td>behavior</td>
<td>for the failure-mode, an object of class behavior describes the system behavior in the liquid path</td>
</tr>
</tbody>
</table>

where, behavior class of objects have the following variables

<table>
<thead>
<tr>
<th>variables</th>
<th>type</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>upstream-behavior</td>
<td>list</td>
<td>system behavior upstream from the location of the failure</td>
</tr>
<tr>
<td>downstream-behavior</td>
<td>list</td>
<td>system behavior downstream from the location of the failure</td>
</tr>
<tr>
<td>behavior-limiting -components</td>
<td>list</td>
<td>for each fluid-path, gas or liquid, the names of components that curtail further propagation of system behavior</td>
</tr>
</tbody>
</table>

Table 6. Description of failure-mode
Object: component-faults

<table>
<thead>
<tr>
<th>variables</th>
<th>type</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocked-shut</td>
<td>list</td>
<td>list of class of objects called fault-description that describe the mode of failure</td>
</tr>
<tr>
<td>stuck-open</td>
<td>list</td>
<td></td>
</tr>
<tr>
<td>leak-in</td>
<td>list</td>
<td></td>
</tr>
<tr>
<td>leak-out</td>
<td>list</td>
<td></td>
</tr>
<tr>
<td>reduced-thermal-efficiency</td>
<td>list</td>
<td></td>
</tr>
</tbody>
</table>

where, fault-description class of objects have the following variables

<table>
<thead>
<tr>
<th>variables</th>
<th>type</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>symptoms</td>
<td>list</td>
<td>symptoms associated with this mode of failure</td>
</tr>
<tr>
<td>cause</td>
<td>string</td>
<td>the cause of the failure</td>
</tr>
<tr>
<td>upstream-behavior</td>
<td>list</td>
<td>system behavior upstream from the location of the failure</td>
</tr>
<tr>
<td>downstream-behavior</td>
<td>list</td>
<td>system behavior downstream from the location of the failure</td>
</tr>
<tr>
<td>gauges-affected</td>
<td>list</td>
<td>gauges with abnormal readings</td>
</tr>
<tr>
<td>cause-effect-association</td>
<td>list</td>
<td>explanations for fault propagation</td>
</tr>
</tbody>
</table>

*Table 7. Description of objects component-faults and fault-description*
An Intelligent Tutor for Diagnostic Problem Solving in Complex Dynamic Systems

Vijay Vasandani, T. Govindaraj, C.M. Mitchell

Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Diagnostic problem solving is a major activity in the supervisory control of complex dynamic systems. Successful fault diagnosis in such systems depends upon the operator's knowledge of the functional properties of the system and timely compilation, integration and application of this system knowledge. An intelligent tutoring system (ITS) that helps to organize system knowledge and operational information including symptom-cause relationships may enhance operator performance. The ITS should incorporate the structure, function and behavior of the controlled system in an appropriate form that achieves cognitive compatibility with the operators. In addition, the ITS must contain a normative task model that provides it with the ability to infer the trainee's misconceptions. This paper proposes a methodology for organizing system and task knowledge in an intelligent tutor to be used for training operators to troubleshoot large dynamic systems.

Introduction

Fault diagnosis in complex dynamic domains forms a major part of the operator's problem solving activity, and its success depends upon the operator's use of system knowledge at multiple levels of abstraction and detail (Rasmussen, 1985). Efficiency in diagnostic problem solving is enhanced by timely compilation, integration and organization of appropriate pieces of operational information about the components and the system (Govindaraj, 1988). An operator training program that helps organize system knowledge and operational information including symptom-cause relationships is therefore essential to ensure competent performance.

Training for diagnostic problem solving tasks can either be provided on-the-job or on simulators. On-the-job training has many problems. First, it is usually very expensive. Second, the consequences of an error can be catastrophic. Finally, malfunctions occur infrequently and it may be undesirable or impossible to duplicate them during the training period. Simulators too, by themselves, are unable to provide appropriate help since they do not have the ability to evaluate a student's misconceptions from observed actions. However, a simulator coupled with an intelligent tutor may improve training effectiveness. Such a combination of simulator and tutor constitutes an intelligent tutoring system (ITS). In the past, research on computer-based training has focused on relatively simple and procedural tasks. Applications of ITS in training operators of engineering systems have been scarce. STEAMER (Hollan et al., 1984), IMTS (Towne et al., 1988), AHAIB (Fath, 1987; Fath et al., 1989), RIT (Woelf et al., 1986), SOPHIE (Brown et al., 1982) and SHERLOCK (Lesgold et al., 1989) are a few examples of ITSs that have addressed engineering domains.

This paper proposes an architecture and a methodology for organizing knowledge in tutors for training operators of large dynamic systems. For such tutors, a framework for evaluating a student's misconceptions based on observed actions is also described. The paper concludes with a description of an application of the proposed architecture.

The Architecture

Figure 1 illustrates the major components of a tutoring system for training operators of complex dynamic systems. The three generally accepted modules of a tutor (an expert, a student and an instructional module) along with a simulator and an interactive interface comprise the architecture for this instructional system. The instructional system has two major requirements: (1) a domain simulator and (2) organization of knowledge that supports the functions of the major elements of the tutoring system.

Simulation Methodology

Development and successful implementation of effective computer aids depends upon the availability of a simulator compatible with human cognition. A simulator designed via qualitative techniques has better cognitive compatibility with operators under training than ones designed from analytical techniques. One qualitative simulation technique is the "Qualitative Approximation Methodology" (Govindaraj, 1987). Using this methodology, large systems can be simulated with a moderate amount of computational power due to reduced computational requirements. QSTEAM, an application of qualitative approximation techniques, simulates the dynamics of a marine power plant under a number of failure situations and forms an integral part of our instructional system.

Figure 1. Instructional System Architecture
Knowledge Organization

Successful implementation of an intelligent tutor for diagnostic problem solving in complex dynamic domains also depends upon the availability of (1) appropriately structured domain knowledge, (2) knowledge of troubleshooting strategies, (3) knowledge to infer a student’s possible intent and, perhaps, misconceptions from observed actions and (4) knowledge of tutoring goals and how they may be realized. The domain knowledge and troubleshooting strategies constitute an expert model of the operator’s task. 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 student misconceptions. Finally, knowledge of tutoring goals and how they are realized guides the instruction and its communication. Requirements and features of the various components of the tutoring knowledge are described next.

Domain Knowledge

Successful fault diagnosis in complex domains is aided by multiple representations of the system’s functional properties (Rasmussen 1986). The expert in a tutor for a diagnostic problem solving task must therefore have access to multiple representations of the system knowledge. Schematics, functional subsystems and fluid paths are three possible means of representing the system knowledge. A schematic is a pictorial representation of the system components. A functional subsystem is a collection of components responsible for performing a higher level system function. Fluid paths are a way of visualizing the system as a collection of different fluids. The three representations of the system knowledge are complementary rather than mutually exclusive and are described in detail below.

Schematics A schematic presents a view into the structure of the system. Typically, a schematic shows the sequence in which certain components and the gauges appear in a real system. It is also a structure that reveals the logical proximity of two physically unconnected components such as the burner and the stack in a combustion unit. A configuration of all components either responsible for a higher level function or sharing a common fluid is yet another example of a schematic.

In diagnostic problem solving tasks on a simulator, schematics are typically used to view the configuration of components and gauges. They provide a convenient interface between the simulated system and the operator. The operator’s interaction with the system involves probing gauge readings in the suspected areas of failure through the schematics.

Grouping of components in schematics for a tutoring system depends upon factors such as frequency of interaction and level of dependency. There are portions of a system that commonly interact with each other. For instance, in a power plant, the performance of steam generation unit is affected by the performance of the combustion unit. Hence, the steam generation unit and the combustion unit are displayed in a single schematic. There are parts of a system which do not significantly affect other portions of the system and thus are viewed in isolation. For example, problems related to lubrication are usually confined to lube oil path and rarely affect other fluid paths, unless left unattended for a long time. Finally, there are some failures in a system that occur more frequently than others. Components and gauges required for investigating such failures are confined, as far as possible, to a single schematic.

Functional subsystems Functional subsystems are collections of components responsible for achieving specific higher level system functions. There are several higher level system functions that collectively contribute to the task goal. For instance, in a marine power plant the functions are: combustion, steam-generation, power-generation, steam-condensation, feed-water-preheating, auxiliary-steam-use, saltwater-service, lubrication and control-air-distribution. A functional subsystem is described by information related to fluid paths passing through the subsystem; (2) components through which a given fluid flows; (3) the order in which the components and gauges appear in each fluid path; (4) the connected subsystem on either side of the fluid path; and (5) the schematic in which the subsystem may be found.

Fluid paths In decomposing a system by fluid path, all components on the same fluid path are represented in a group. Additional system knowledge based on fluid paths consists of (1) schematics in which the fluid is found, and (2) the subsystems through which the fluid flows. Examples of fluid paths in steam power plants are combustion-air, fuel-oil, flue-gas, superheated-steam, desuperheated-steam, feed-water-condensate, main-condenser-hot-fluid, main-condenser-cold-fluid, saltwater, lube-oil and control-air.

Each of these three system representations is an aggregation of mechanical components. Therefore, the lowest level of system description is at the component level. System knowledge at the component level concerns a component’s structure, function and behavior. A component’s structure, for the most part, refers to its connections to other components on the input and output sides, the fluids carried by it, the gauges attached to it, and its association to a schematic or a functional subsystem. Since the structural changes in the components are usually responsible for abnormal behavior of the system, the component level structural description for the failed and normal modes of a component are different. A component’s function depends upon its structure and defines the purpose of the component in the system and the contribution it makes to the higher level functions of the system. Finally, the behavior of a component concerns its states. The manner in which the component state values are affected by the presence of a component in both the normal and failed states constitutes behavioral knowledge.

Since the behavior of a component is different under normal and failed modes, the behavioral knowledge, like the structural knowledge, is different for the two modes. Together, the structural, functional and behavioral knowledge of components form an essential part of the expert’s knowledge.

Domain knowledge, although essential, is not sufficient for the troubleshooting task. Troubleshooting knowledge includes more than the operational knowledge of the system and its components.

Troubleshooting knowledge

Troubleshooting knowledge combines system knowledge and diagnostic strategies. Troubleshooting knowledge includes general knowledge of the types of failures in the system, detailed information on certain common failures, and cause-effect associations for familiar failures. This section discusses the nature of the diagnostic problem solving knowledge.

A mechanical component in a physical system can fail in more than one way. There are five common modes of failure in components: (a) blocked-shut, (b) stuck-open, (c) leak-in, (d) leak-out, and (e) reduced-thermal-efficiency (Fath, 1987; Fath et al., 1989). Faults in components fit one or more of these five
mode types. Each failure mode is responsible for a system behavior that manifests in the form of a typical pattern of abnormal state values (Fath, 1987; Fath et al., 1989). The typical system behavior associated with fault depends upon the phase of the fluid in the affected path. A blocked-shut mode of failure in a liquid path, for example, causes the liquid level downstream to be lower than normal and the level upstream higher than normal. A similar blocked-shut failure in a gas path, on the other hand, decreases the downstream gas pressure and increases the upstream pressure. Such patterns, when detected, are helpful in reducing the list of suspected components to those that fail in that particular mode.

Typical system behavior associated with a failure mode sometimes deviates from the expected abnormal behavior (Fath, 1987; Fath et al., 1989). The way in which the system components are configured is often responsible for such a deviation. For instance, a source-sink such as a deaerating-feed-tank located downstream in the blocked-shut feed-water-path may prevent further propagation of low feed-water level. The deaerating-feed-tank imposes such a behavior on the system because it is an "infinite" source of feed-water which can at least temporarily compensate for any loss in the water level. Knowledge of such deviations from the norm is also essential for correct identification of the type of failure in the system.

Finally, to isolate the failed from the suspected components and to diagnose the fault, additional information such as the gauges affected by the failure and causal relationship between abnormal system states for every fault is required. Knowledge of the affected gauges and the system states for individual faults can help verify the final diagnosis.

There are other elements of the troubleshooting knowledge, accumulated through experience, that make fault diagnosis in a large complex system time efficient (Govindaraj and Sa, 1988). This experiential knowledge, based on prior cases of solved and unsolved problems encountered by the operator, is usually responsible for the formation and rapid refinement of an initial set of hypotheses of either suspected components, subsystems, or fluid paths.

Experiential knowledge is activated by the observation of obvious and non-obvious (i.e., discovered only upon investigation) symptoms. In a complex dynamic system, the size of the system and the effects of fault propagation make it impossible to uniquely associate a symptom with a specific fault. However, in such systems, observable symptoms help to limit the search for the failed component to a specific location in the system. For example, the symptoms may indicate that a particular higher level function of the system has been affected by the fault. This helps to confine the search for the failed component to components comprising the subsystem responsible for the affected function. Symptoms may further help to categorize the faults, for example, it may separate those related to components with moving parts from those related to speed or load. Such a categorization of failures further reduces the search space for a failed component. For example, a search space generated by a set of all components with moving parts in the combustion system of a power plant is likely to be much smaller than the set of all components in the combustion subsystem.

One form of experiential knowledge is a collection of failure schemas. The schemas represent some of the familiar ways in which the system fails. A schema is activated by a symptom and proposes a hypothesis or a partial solution to the diagnostic problem. The partial solution may be a diagnostic test that either provides a conclusive inference or activates another schema. For example, smoke in a boiler may activate a schema that recommends checking for smoke color. Black smoke may then trigger an incomplete-combustion schema while a white smoke may trigger an excessive-air-in-the-burner schema. An abnormal fuel temperature with black smoke in the boiler may prompt the incomplete-combustion schema to specify desuperheated-steam or fuel path as the path suspected of containing the failed component.

Rasmussen (1986) has characterized the application of the troubleshooting task knowledge into two diagnostic strategies: symptomatic search and topographic search. Symptomatic search is a simple and economical pattern matching strategy where a successful association between cause and effect is generated based on prior experience. An unsuccessful attempt with symptomatic search usually leads to topographic search. In topographic search, an hypothesis about the failed component is generated and tested by comparing a model of normal behavior of the suspected component with its behavior in the abnormally functioning system. Neither the symptomatic nor the topographic strategy is adequate in itself; instead, an expert often has to switch between the two strategies many times to complete the task.

Interestingly, knowledge organization that is suitable for expert performance is not necessarily suitable for instruction or for evaluating student's misconceptions (Clancey, 1987). A normative model of the student's actions is an alternative organization of the expert's task knowledge that may help evaluate a student's misconceptions.

Normative model of the student's actions

An important feature of an intelligent tutor is its ability to evaluate a student's misconceptions. This capability of the tutor depends on a normative model of the student's actions. Valid student actions at the student-tutor interface range from requests for help to responses to queries, and calls for new schematics. In addition, in diagnostic problem solving, there may be some other actions performed by the student. These actions include investigating components for gauges and checking the gauge readings. Actions to display gauge readings are the informative actions taken by the student in the process of troubleshooting. Most of the student's actions are self-explanatory. The informative actions are, however, associated with ambiguity concerning the student's intent. Context-specific knowledge and an understanding of the cognitive aspects of troubleshooting task are needed to resolve these ambiguities.

In a troubleshooting task, the student maintains a set of failure hypotheses that explain the abnormal behavior of the system (Fath, 1987; Fath et al., 1989). A set of hypotheses is a list of components suspected to have failed. Each informative action taken by the student is an attempt to reduce the size of the set of failure hypotheses. The manner in which the list of suspected components may be revised depends upon the outcome of the diagnostic test associated with the informative action. The test results have a context-specific significance. For example, in a power plant, if the student has been alerted by a low condensate-pressure alarm, it makes sense to check the pressure gauge on the condensate pump. If the student checks the pressure gauge on the condensate pump, it is reasonable to assume that the condensate pump is probably one of the suspected components. If the pressure gauge shows a low reading, the student has reason to continue suspecting a malfunction in the condensate pump. On the other hand, if the pressure gauge reading is normal, the condensate pump may be omitted from the list of suspected components. However, if the student is alerted to a failure by smoke in the boiler rather than a low condensate pressure alarm, checking for pressure across the condensate pump is
inconsistent with the failure data. Thus, the knowledge of reasonable actions under various failure situations and how test results refine the set of failure hypotheses constitutes a normative model that can help in evaluating the student's misconceptions.

Evaluation of misconceptions

The normative model describes what a student ought to do under a particular failure situation. When the student's action does not match actions suggested by the normative model, the reason can be attributed to many causes. Usually the causes are related to lack of knowledge, inappropriate knowledge or deficiencies in knowledge application skills. Evaluating a student's misconception means determining the probable cause for the deviant behavior.

In order to determine a student's misconception, the tutor needs to know the types of misconceptions that are associated with incomplete knowledge of the system or the task. Our method of organizing the knowledge in the expert module is also helpful in organizing categories of misconceptions. Misconceptions can be categorized as those related to lack of (1) structural knowledge of the system, (2) functional knowledge of system and components, and (3) knowledge of system behavior resulting from failures. Lack of system structural knowledge makes the student investigate portions of the system unrelated to the failure. Checking components and gauges in the fluid paths unaffected by failure indicates a lack of understanding of different system functions and their inter-relationships. Finally, pursuing an hypothesis that should have been rejected based on evidence gathered, or premature elimination of suspicion from a component due to insufficient evidence, suggests shortcomings in behavioral knowledge related to failures.

After evaluating a student's misconception, an intelligent tutor provides instructions to rectify the misconception and to improve the student's diagnostic problem solving skills. The selection of appropriate sets of instructions and their presentation is guided by strategies, some of which are discussed next.

Instructional strategies

The instructional module of an ITS contains knowledge that specifies how the tutor should respond to various student actions. Many of the instructional modules rely on a rule-based structure to create instructions (e.g., Burton and Brown, 1982; Clancey, 1987). More recently, Woolf and McDonald (1984) and Macmillan et al. (1988) have proposed architectures for dynamic instructional planners in adaptive environments. However, in any architecture, the key issues to be addressed are the instructional content, its form and time of presentation.

Instructional content depends upon the instructional objectives. Several units of instruction may be available that satisfy these objectives. Selection of a particular unit of instruction is governed by instructional strategies chosen for the tutor. Such strategies may, under different situations, include preference for hints or discussion of generalities as opposed to solutions or discussion of specifics, preference for graphical instead of textual mode of presentation, and preference for presenting material with or without intervention. Instructions without intervention can only be provided at the end of a training session. While non-intervention has some advantages because it does not disturb the student's thought process, intervention at critical stages of diagnostic activity may be an effective way of emphasizing a point. Furthermore, having a flexible control structure for instruction selection that can be manipulated for different contexts is desirable.

The tutoring system architecture and the knowledge organization proposed in this paper have been integrated into a tutoring system. This implementation is described next.

Implementation

To evaluate the proposed architecture and the knowledge organization methodology, a tutoring system is being implemented on an Apple Macintosh II computer in Alcoa's Common Lisp with object-oriented extensions. This section provides an introduction to the training environment and discusses the representation of system and troubleshooting task knowledge, the normative model of student's actions, and instructional strategies.

The training environment

The simulator, QSTEAM, is an application of qualitative approximation techniques which simulates the dynamics of a marine power plant under a number of failure situations. QSTEAM also provides an environment for training operators to troubleshoot the marine power plant. Its interface with the trainee has a fixed menu displaying icons representing the various schematics in QSTEAM. A schematic can be called by clicking on the icon associated with the schematic. There are seven schematics: steam, boiler, feed-water, fuel-oil, control-air, saltwater and lube-oil. Together, the seven schematics provide a comprehensive, yet economical, layout of the marine power plant (Figure 2).

In the schematics of QSTEAM, all the major stages of power generation and fluid paths can be distinctly identified. Configuration of components responsible for a higher level system function is mostly confined to a single schematic. A functional subsystem may span over multiple schematics; however, related schematics can be accessed from each other. To move between these related schematics, one clicks on an icon representing the related schematic. In moving from one schematic to another, visual momentum (Woods, 1984) is essential and is provided by (a) high-lighting an icon in the new schematic to establish orientation with respect to the old schematic; (b) making it possible to switch back to the old

![Figure 2. Section of steam-schematic](image-url)
schematic by clicking on the high-lighted icon; and (c) providing a smooth visual transition between schematics, as if the schematics are portions of one big schematic and moving from one schematic to another is equivalent to scrolling. In addition, representation of a component in multiple schematics is indicated on the schematic by icons adjacent to such components. These icons mark regions from which the schematics they represent can be accessed.

The student is provided with the facility to investigate gauges attached to any of the components displayed in the current schematic. A mouse click on a component displays all the gauges attached to it, including their types (e.g., pressure, temperature, or flow) and locations along the fluid paths. Clicking on any of the displayed gauges shows the gauge reading. At any instant, only the most recently investigated component has its gauges and the observed gauge readings displayed in a schematic. However, if the student wishes to keep a gauge and its reading locked on a schematic for future reference, it is possible to do so.

Dialog boxes and menus are used to facilitate other valid interactions with the system. A student's interactions with the system are monitored to evaluate misconceptions and provide instructions.

Knowledge representation

The method of knowledge representation in the tutor uses objects to represent knowledge structures. This approach is supported by an object-oriented environment that facilitates the creation and manipulation of abstract data types without replication of code. For example, components that are instances of the same functional primitive have similar object representations and share the same methods that create and manipulate them. Examples of objects used for representing knowledge of subsystems, fluid paths, components, failure schematics, system behavior associated with failure modes, and the cause and effects of each failure are shown in Tables 1-5.

Object: schema

Object: object-vector

Object: component-faults

Object: fluid-path

Table 1. Description of subsystem

Table 2. Description of fluid-path

Table 3. Description of simple-primitive

Table 4. Description of schema

Table 5. Description of objects component-faults and fault-description

The normative model of student behavior is also represented by objects. For each failure condition, the model has a list of affected components, fluid paths, gauges, and the schematics in which these affected system entities may be found. All student actions are recorded and compared with actions specified in the normative model for the failure situation being investigated. This comparison is made by the instructional module of the tutor. If differences appear, the tutor has a rule-based framework to classify the student deficiencies into three types of misconceptions resulting from lack of: (1) structural knowledge of the system, (2) functional knowledge of the system and components, and (3) knowledge of system behavior associated with failure.

Even though it is impossible to completely understand a student's behavior, a context-specific guess is made using a set of rules for evaluating the misconception. For instance, if a student's investigative actions do not involve any of the schematics specified for the failure, the tutor interprets them as a lack of knowledge of system structure. If, on the other hand, the student investigates components, fluid paths or gauges that do not contribute to the process of problem solving, the tutor deduces a problem related to lack of knowledge of the system functions and their interactions. Sometimes, to determine the type of misconception, the tutor relies on feedback from the student. For example, consider the case where an investigator investigation of a component are made even when conclusive evidence has already been gathered that eliminates any possibility of failure in the component. Such a case indicates a deficiency in the
student’s ability to relate suspected failures with system behaviors. However, before providing remedial instructions, the tutor queries the student to ascertain the hypothesis suggested by his actions.

**Instruction**

The instructional module of the tutor monitors and evaluates every action of the student. If the student’s action matches one of those specified by the normative model, the tutor may compliment the student for his efforts. If not, the tutor invokes rules for evaluating misconceptions to determine the cause of the mismatch. If the cause cannot be traced to any of the three preconceived types of misconceptions, no intervention by the tutor is considered necessary. On the other hand, if the type of misconception is identified, another rule-based structure containing instruction selection rules is activated to suggest a general remedy that can perhaps rectify the misconception. General remedial procedures for each type of misconception are encoded as consequences of the instruction selection rules. Example of such a procedure may be show-fluid-paths. The instructional module then manipulates the general remedies to prepare an instructional unit suitable for the current context. For example, the general procedure show-fluid-paths may only be applied to fluid paths in the context of the current failure. Finally, a presentation rule decides the mode of presentation of instructions.

In addition to instructions through intervention, the instructional module provides the students with the facility to request help or hints and explore the expert’s knowledge base through an instructional menu. Using this menu, a student can check different modes of failure in a component, its behavioral characteristics during failed states, the composition of functional subsystems, feasible failure hypotheses at any time and the next best diagnostic test to refine the set of hypotheses. Aid provided to the student in this manner is “student-initiated” as opposed to “tutor-initiated.”

**Validation**

Experiments have been planned to determine the feasibility and utility of the proposed architecture using the implementation described above. The experiments will explore the benefits of using the instructional system in addition to using a training simulator, to train operators to troubleshoot complex dynamic systems. The misconception inferring capability of the system will also be evaluated.

**Acknowledgements**

This research is supported by contract N00014-87-K-0482 from the Manpower R&D Program, Office of Naval Research (Dr. Susan E. Chipman, contract monitor).

**References**


Qualitative Simulation and Intelligent Tutoring Aids for Training in the Operation of Complex Dynamic Systems

T. Govindaraj, Principal Investigator

Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205
E-mail: tg@chmsr.gatech.edu
(404) 894 3873

27 July 1990

Progress Report for June 1989 - June 1990

Submitted to

Dr. Susan E. Chipman
Program Manager, Cognitive Science, Office of Naval Research
Manpower, Personnel, and Training R&D Program
Contract Number: N00014-87-K-0482
Progress report for June 1989 - June 1990

In this report we describe the progress of our research on the development of an intelligent tutoring system for the marine power plant simulator during the third phase of the three-year contract. Under the current ONR contract, we have three major objectives: 1) to develop an architecture for intelligent tutoring systems for diagnostic problem solving in the supervisory control operation of complex dynamic systems; 2) to implement the tutor for the marine power plant simulator; and 3) to develop and evaluate a training program using the tutor implementation. The research described here concentrates on item (3), i.e., the implementation of the ITS. Progress during the period under review is described below.

At the end of the second year, the design of the tutoring system was completed. A substantial portion of the system had also been implemented, including the graphical interfaces. In the third year, an architecture for knowledge representation in the tutor was developed and implemented. A paper describing the details of knowledge representation was presented at the IEEE International Conference on Systems, Man, and Cybernetics in November 1989. Briefings, including software demonstrations, were given to the OCNR MPT R&D Committee in October 1989 and at NPRDC in January 1990. Another presentation was given at the Workshop on AI and Advanced Automation Techniques for Fault Diagnosis and Recovery in June 1990 at NASA Johnson Space Center.

Major effort during this reporting period involved the development and implementation of the three modules or components of the tutor and the enhancement of the graphical interface. An expert module, a student module, and an instructional module constitute the tutor. The domain knowledge and troubleshooting strategies constitute an expert model of the operator's task. The instructional module uses this model to train students to use proper diagnostic problem solving strategies. Knowledge of student actions can help the instructional module to infer possible misconceptions that students may have. Finally, knowledge of tutoring goals and how they are realized guides the instruction and its communication. The intelligent tutoring system integrates the model of the tutor with the simulator and incorporates a direct manipulation graphical interface.

Structures for knowledge representation are based on our analysis of expert's fault diagnosis task. The domain knowledge is represented in multiple, but complementary, views of the system's structure, function, and behavior. The representations used are the schematics, the functional subsystems, and the fluid paths. The troubleshooting knowledge is a combination of system knowledge and diagnostic strategies, including general knowledge of failures and cause-effect relationships about commonly occurring failures. Knowledge about components, in both normal and failed states, are also represented in terms of structure, function, and behavior. Results from our own past research as well as those of others were used to develop instructional strategies and means for inferring student intent and misconceptions. These instructional strategies are implemented as heuristic rules.
Details of realistic failures are used to develop and implement knowledge of troubleshooting strategies. When the tutoring system is used for training, students will troubleshoot the cause of these failures based on symptoms provided.

Essential details of the knowledge representation used in the tutoring system are described in Vasandani and Govindaraj [1990] (Appendix A). Further details and descriptions will appear in a paper to be included in the proceedings of the 1990 IEEE International Conference on Systems, Man, and Cybernetics.

We had hoped to complete the tutor implementation and experimental evaluation by the end of the third year of the contract. However, the complexity of the domain, the model of knowledge representation, and the software hampered our progress. The instructional system that integrates the simulation and the tutor is comprehensive and robust, and hence is expected to be very effective for training.

In summary, the implementation of the tutoring system has been completed. Pilot experiments are expected to begin in August. We plan to start conducting experiments to evaluate the tutor and the training program in the fall quarter of 1990.
Appendix A

Knowledge Representation and Human-Computer Interaction
in
an Intelligent Diagnostic Problem Solving Tutor

Vijay Vasandani and T. Govindaraj

Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332
(404) 894-4322
vijay@chmsr.gatech.edu

Abstract

In the supervisory control of complex dynamic systems, diagnostic problem solving requires knowledge of the functional properties of the system and the skills to compile, integrate and apply this knowledge. Intelligent tutoring systems (ITS) that train operators to troubleshoot such dynamic environments help to organize system knowledge and operational information. The human-machine interface of such ITSs serves as a medium of knowledge communication between the tutor and the student. The interface, therefore, is an essential constituent of the instructional system. The instructional system describes the structure, function, and behavior of the controlled system to the student via this interface. It is important to design a good interface because it influences the manner in which the student will eventually conceptualize the domain, visualize the problems, and cope with the complexities of the task and the domain. Whereas a well designed interface promotes cognitive compatibility with users, a poorly designed interface can diminish the effectiveness of the entire instructional system. This paper focuses on how knowledge and human-machine interaction should be effectively organized in an intelligent tutor to be used for training operators to troubleshoot large dynamic systems.

The knowledge requirements of computer-aided instructional systems vary with the distribution of teaching and learning responsibility between the student and the tutor (Rickel, 1989). Tutors that attempt to maintain a balance of control between the student and the tutor (i.e., mixed-initiative tutors) require the largest amount of structured

1. This research is supported by contract N00014-87-K-0482 from the Manpower R&D Program, Office of Naval Research (Dr. Susan E. Chipman, contract monitor).
knowledge as compared to tutor-dominated traditional CAI programs or student-dominated discovery learning environments (Sleeman and Brown, 1982; Wenger, 1987; Psotka et al., 1988).

Successful implementation of an intelligent mixed-initiative tutor for diagnostic problem solving depends upon the availability of (1) appropriately structured domain knowledge, (2) knowledge to troubleshoot failures, (3) knowledge to infer a student's possible misconceptions from observed actions, and (4) pedagogical knowledge to realize the tutoring objectives. While architectural details of such an ITS are discussed in Vasandani et al. (1989), this paper describes a methodology for representing knowledge in the ITS.

The knowledge organization and the human-computer interaction proposed in this paper have been integrated into an instructional system. The instructional system will train operators to troubleshoot a simulated marine power-plant described in Govindaraj (1987). This paper concludes by providing details of the implementation.

System knowledge for an intelligent tutor can have several representations. These representations can have different levels of description or detail depending on the task for which the knowledge is required (Rasmussen, 1986). For most large, complex, mechanical systems, such as a steam power-plant, a comprehensive representation of system or domain knowledge is via functional subsystems, fluid paths and schematics. A functional subsystem is a collection of components responsible for performing a higher level system function such as combustion in a power-plant. Fluid paths are a way of visualizing the system as a collection of different fluids. The fluids may be fuel, steam and air in a steam power-plant, or electric current in an electrical circuit. A schematic is a pictorial representation of the system structure through its components and gauges. For large systems, a pictorial representation usually spans over several schematics. Although the three representations are complementary rather than mutually exclusive, multiple representations of the system knowledge are necessary to enhance diagnostic performance (Rasmussen, 1986).

Each of the three system representations is an aggregation of mechanical components. System knowledge at the component level of description concerns a component's structure, function and behavior at an appropriate level of detail necessary for successful fault diagnosis. A component's structure refers to its input and output connections to other components, the fluids carried by it, the gauges attached to it, and its association with a functional subsystem. A component's function depends upon its structure and defines the purpose of the component in the system and its contribution to the higher level system functions. The manner in which the system state values are affected by the presence of a component in both the normal and failed states constitutes behavioral knowledge.

System knowledge, although essential, is not adequate knowledge for a diagnostic problem solving tutor. Troubleshooting knowledge, in addition to system knowledge, is equally important and includes general knowledge about types of common failures in the system, detailed information on certain common failures, cause-effect associations for familiar failures and behavior of components and subsystems under failure conditions (Govindaraj and Su, 1988; Fath et al., 1989).
Troubleshooting task knowledge, like the system knowledge, can have several representations. However, the representation most suitable for expert performance is not necessarily the best representation of the troubleshooting task knowledge for a tutor (Clancey, 1987). The tutor's troubleshooting task knowledge has to be organized in a manner that facilitates inference of possible misconceptions based on observed actions. Therefore, expressing troubleshooting task knowledge, for each failure, in terms of valid student actions at the student-tutor interface is useful for a tutor. Such a representation provides a normative task model of troubleshooting and offers a convenient way to interpret and analyze student actions.

The knowledge requirements of an intelligent tutor also include knowledge to evaluate and rectify misconceptions. When a student's actions deviate from the normative task model of troubleshooting, they may be explained in terms of misconceptions related to system structure, function or behavior. The knowledge that relates a deviant action to a probable misconception and the knowledge to rectify the misconceptions are both essential elements of the instructional system. A rule based structure is one efficient way to represent knowledge to evaluate and rectify misconceptions.

Finally, pedagogical knowledge concerning instructional content, form and time of presentation also needs to be addressed in the design of the instructional system. Instructions can be provided with or without intervention. Instructions without intervention is provided at the end of a training session. While non-intervention has the advantage of not disturbing the thought process of the student, intervention at critical stages of the training period is an effective way of emphasizing a point.

System, troubleshooting and pedagogical knowledge are important for an instructional system, but without a good student-tutor interface, knowledge alone is insufficient to make a tutor effective. A well designed interface facilitates knowledge communication, 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). In a diagnostic problem solving task, schematics often serve as a convenient student-tutor interface for knowledge communication. These schematics minimize the external-internal task mapping problem when the valves and gauges that are usually under the control of the student appear in them as manipulable objects. This paper discusses factors such as grouping of components and the role of graphical techniques in the design of schematics as an effective student-tutor interface.

Grouping of components in schematics for a tutoring system depends upon factors such as degree of logical proximity between components and subsystems; the extent of diagnostic actions necessary in a region 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).

Instructional systems with schematic interface make strong use of graphics and icons (Hollan et al., 1984). The graphical objects or icons 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. A trapezoidal shape representation of a steam turbine 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 may also be represented by meaningful icons.

Schematics not only provide an interface between the student and the simulated system but also between the student and the tutor. The tutor uses the schematics to highlight components that constitute a subsystem or share a common fluid path. Such graphical techniques promote visualization of functional subsystems and their interaction. Schematics are also used by the tutor to animate fault propagation by highlighting gauges as they turn abnormal for 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 problems and hypothesis. 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. This paper also discusses how the student and tutor initiated interaction should be organized effectively in a diagnostic problem solving tutor for a complex dynamic domain.

In student-initiated communications, the interface has to assume the responsibility of guiding the user into asking the right type of questions. For instance, when seeking information concerning the system’s structure, function or behavior, it is helpful to make students select appropriate and context relevant queries from a set of menus. Such menus when organized hierarchically 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 design involves getting the student to understand and correctly respond to the queries. Where a student can respond to a query in multiple ways, the student options have to be recognized and the choice restricted to these options. For example, when inquiring about the suspected mode of failure during a troubleshooting task, the student’s response has to be confined to the set of failure modes known to
the tutor. Therefore, selection from viable alternatives rather than unguided response is a better approach to interface design.

References


19 July 1988

Dr. Susan E. Chipman
Program Manager, Cognitive Science
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000

Dear Susan:

Enclosed please find twenty copies of the annual report on my ONR Contract (N00014-87-K-0482) for the period June 1987 - May 1988.

Sincerely yours,

T. Govindaraj
Associate Professor of
Industrial and Systems Engineering

Enclosures
Qualitative Simulation and Intelligent Tutoring Aids for Training in the Operation of Complex Dynamic Systems

T. Govindaraj, Principal Investigator

Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205
E-mail: tg@cmmsr.gatech.edu
(404) 894 3873

19 July 1988


Submitted to

Dr. Susan E. Chipman
Program Manager, Cognitive Science
Office of Naval Research, Manpower R&D Program
Contract Number: N00014-87-K-0482
In this report we describe the progress of our research on the development of an intelligent tutoring system for the marine power plant simulator that we had designed and implemented during the previous ONR contract. The simulator, implemented in Interlisp-D, runs on Xerox 1100 series of Lisp machines. Under the current ONR contract, we have three major objectives: 1) to develop an architecture for intelligent tutoring systems for diagnostic problem solving in the supervisory control operation of complex dynamic systems; 2) to implement the tutor for the marine power plant simulator; and 3) to develop and evaluate a training program using the tutor implementation. During the first year of the project, a number of subgoals were identified to accomplish our objectives. The subgoals and the progress during this period are described below.

When the contract started in June 1987, Vijay Vasandani, a doctoral student with backgrounds in mechanical, industrial, and systems engineering, joined the project. Our immediate goal was to help him become proficient with the details of the simulation so that he can contribute to the objectives of the project. Even though he has had some practical experience with steam power plants, the complexity of the steam power plant simulation and his lack of familiarity with the marine power plant domain and LISP, necessitated the need to devote a large amount of time to familiarize him with the principles and implementation details of the simulation and the overall nature of the project. In addition to learning INTERLISP and the operation of the Xerox 1109 Lisp machine, we spent time on the details of how the qualitative approximation technique is implemented in developing the power plant simulator.

While learning INTERLISP and the details of the simulation, Vasandani also conducted a literature survey to become familiar with the state of the art in knowledge structures and control strategies presently being used in tutoring and training systems. This exercise contributed to the understanding of the domain better and enabled the assessment of the degree of difficulty and complexity associated with the domain in relation to that exhibited by other tutoring and training domains.

In the months following the initial familiarization, a small scale preliminary model of expert's fault diagnosis task was developed. This model was based on ideas proposed in an earlier study conducted during our previous ONR contract where data and protocols were collected from experts performing the troubleshooting task on the simulator of the marine power plant. Relevant information about two common modes of failure for a single component were compiled into frames which constitute our choice of knowledge base for representing failure knowledge. Given the
obvious symptoms of failure, a control strategy has been designed to first identify frame or frames that provide a plausible explanation for the symptoms. Information contained in the failure frames and the symptoms is used to proceed with the task in a fashion typical of any forward chaining process. The process represents some of the flavors of schema instantiation, generate and test paradigms, and symptomatic search strategies adopted by experts to diagnose a complex dynamic system. This approach is expected to facilitate upgrading of the model by extending its fault detection capabilities to include other failures with the incorporation of new failure frames. An abstract based on this work was submitted to a conference§ and has been accepted for presentation. A paper describing the details of the work is under preparation and will be completed in August 1988.

In addition to the educational efforts concerning the simulation details, we began converting the simulation into Common Lisp, first on the Xerox Lisp machines and then on Apple Macintosh II computers (Allegro Common Lisp). This translation to Common Lisp became necessary since the anticipated computational needs of the simulator and the tutoring system exceeds the capabilities of the D-machines. Also, translation to Common Lisp on a personal computer/workstation at this time appears to provide the best opportunity because of Xerox's decision to leave the workstation market and the possibility to complete the conversion while the new students are still in the process of learning the principles and implementation details of the simulation. Porting the simulator to the Macintosh environment at this time has the advantage of enabling the students become fully familiar with the project while affording us the opportunity to develop a more robust and versatile version of the simulator.

We have completed rewriting major portions of the code for the simulation using objects in the Common Lisp Object System (CLOS). The primitives that form the basis for the simulation have been implemented as objects. The manner in which the system states and connectivity of components are represented and the states are propagated has been improved considerably by the use of objects. In addition, the Interlisp-independent functions have been translated for the ACTIVEREGIONS package needed for the direct manipulation, graphical interface.

As a result of the conversions and updates, we now have code that is more modular and hence easier to maintain. This should enable us to more easily implement and incorporate the tutor. Also, new users can become conversant with the simulation in less time than in the previous version. Redoing

§ Sixth Symposium on Empirical Foundations of Information and Software Sciences (EFISS), 19-21 October 1988, Atlanta, Georgia.
the simulation in CLOS has already improved the efficiency and effectiveness with which changes can be introduced as the program evolves.

In addition to his efforts to understand the details of the simulation, participation in the porting and translation of the simulation from Interlisp-D to Common Lisp, and implementation of portions of the code in CLOS, Vasandani has been working on developing control interfaces into the simulation. The original simulation was not designed to handle operator inputs that altered the state or the configuration of any of the components. In the version of the simulation under development, this will be changed so that an operator can open and close valves, change the operating conditions, etc.

A considerable amount of effort during the first year was spent in bringing new students up to speed on the details of the simulation. While we had hoped to complete the translation into Common Lisp by the end of the first year and have the program running on the Mac II, the complexity of the dynamic system operation and the simulation details have resulted in the need to invest a large amount of time in training new students. We have managed to involve a doctoral student (Vasandani) who is now familiar with the details of the simulation and complete a major portion of the simulation. Even with a background in mechanical engineering, including some experience with steam power plants, it has taken Vasandani more than three quarters to become familiar with the details of the simulation and system operation. Due to the nature of the problems and the level of difficulty, it seems inevitable that a large portion of time needs to be spent in maintaining the simulation.

Progress was also hampered by the difficulty of finding qualified students capable of understanding the complexity of dynamic systems such as power plants. For instance, a master's student, Lauren Weisberg, joined the project during the second quarter. While she had some experience with Common Lisp, she lacked any experience with power plants or other complex dynamic systems. After spending three quarters on the project learning the basics of the simulation, during which period she participated in the translation effort, she decided not to pursue a thesis and hence left the project. The newer version of the simulation running on Apple Macintosh II computers is expected to help attract and retain capable students to work on various aspects of the project.

We plan to complete the simulation by the end of September 1988. Designs for an intelligent tutor are already being planned. A preliminary version of the tutor should be ready by the end of November. After pilot experiments during December 1988 - January 1989, we plan to start conducting experiments in March 1989.
Dear Susan:

Enclosed please find five copies of the final report on my ONR Contract (N00014-87-K-0482) for the period June 1987 - December 1990. A technical report to be published later this year will provide complete details of the instructional system and detailed analysis of experimental data.

Sincerely yours,

T. Govindaraj

Associate Professor of
Industrial and Systems Engineering

Enclosure
Qualitative Simulation and Intelligent Tutoring Aids for Training in the Operation of Complex Dynamic Systems

T. Govindaraj
Vijay Vasandani

Technical Report CHMSR-91-1

Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205
+1 404 894 3873

1991 February 28

Final Report

for

Office of Naval Research
Contract N00014-87-K-0482
1987 June 1 - 1990 December 31

Approved for public release: distribution unlimited. Reproduction in whole or in part is permitted for any purpose of the United States Government.

This report was prepared under the Navy Manpower, Personnel, and Training R&D Program of the Office of the Chief of Naval Research under Contract N00014-87-K-0482.
Qualitative Simulation and Intelligent Tutoring Aids for Training in the Operation of Complex Dynamic Systems

T. Govindaraj and Vijay Vasandani

Supervisory control operators of complex dynamic systems must possess thorough knowledge of the systems, and skills to apply this knowledge to maintain system operation. Use of intelligent tutoring systems (ITS) can be economical for training such operators. An ITS can help organize system knowledge and operational information and also provide practice-to-develop appropriate skills. We have developed and implemented an ITS on an Apple Macintosh II computer for the marine power plant domain. The ITS is comprised of a simulated power plant, the tutor, and mouse-based direct manipulation graphical interfaces. The ITS was experimentally evaluated using Naval ROTC cadets as subjects under three training conditions: (1) simulator only, (2) a passive tutor that provided help upon request and without intervention, and (3) an active tutor with intervention. Performance in all three conditions was analyzed in identical data collection sessions. Performance measures such as percentage of premature and correct diagnosis and percentage of relevant and irrelevant diagnostic tests were used. Experimental results show that a simulator alone is inadequate, whereas a simulator in conjunction with an ITS can help develop efficient troubleshooting skills. However, not all students are equally receptive to every tutoring strategy and provisions must be made in training programs for individual preferences and differences in abilities and styles.
ABSTRACT

Supervisory control operators of complex dynamic systems must possess thorough knowledge of the systems, and skills to apply this knowledge to maintain system operation. Use of intelligent tutoring systems (ITS) can be economical for training such operators. An ITS can help organize system knowledge and operational information and also provide practice to develop appropriate skills. We have developed and implemented an ITS on an Apple Macintosh II computer for the marine power plant domain. The ITS is comprised of a simulated power plant, the tutor, and mouse-based direct manipulation graphical interfaces. The ITS was experimentally evaluated using Naval ROTC cadets as subjects under three training conditions: (1) simulator only, (2) a passive tutor that provided help upon request and without intervention, and (3) an active tutor with intervention. Performance in all three conditions was analyzed in identical data collection sessions. Performance measures such as percentage of premature and correct diagnosis and percentage of relevant and irrelevant diagnostic tests were used. Experimental results show that a simulator alone is inadequate, whereas a simulator in conjunction with an ITS can help develop efficient troubleshooting skills. However, not all students are equally receptive to every tutoring strategy and provisions must be made in training programs for individual preferences and differences in abilities and styles.

INTRODUCTION

Human supervisory control

In the operation of complex dynamic systems such as aircrafts and power plants, human supervisory control operators must process vast quantities of information promptly to maintain desirable levels of system performance. Various subsystems of a complex system generate a large amount of information. This information about the system state must be combined with external inputs from the environment and processed promptly. Even though computers and automatic control systems are generally employed to process information in real time, complete automation based on fully autonomous systems is not possible. Human presence is still required to set high level system goals, monitor system states, and intervene and compensate for problems that the automated control systems are unable to handle.

In such supervisory control environments, the effectiveness of the human operator and his ability to identify and integrate relevant information in a timely manner depends upon his knowledge of system operation, his problem solving skills, and the assistance provided by the system in the form of appropriately chosen and processed information. In this report, we describe re-
search aimed at developing problem solving skills via moderate fidelity simulators and intelligent computer aids.

**Operator training for problem solving**

Problem solving skills in complex dynamic environments are typically acquired by training on actual systems or on simulators after some basic level of domain-specific knowledge has been acquired. Training on actual systems is usually very expensive. Furthermore, it is undesirable, and often impossible, to simulate failure conditions on actual systems. Hence, simulators can be used to achieve the training objectives at a relatively moderate cost.

In our previous study of training, we investigated the levels of simulator fidelity necessary for teaching problem solving skills (Su, 1985). The domain was oil-fired marine steam power plants. We identified three dimensions of fidelity that were important for training: physical, structural, and dynamic. Physical fidelity relates to the appearance of the simulator in relation to the actual system. Structural fidelity refers to the functional relationship between components. Dynamic fidelity concerns the response and behavior of the simulator states with reference to those of the system. We developed a low fidelity simulator that had moderate structural fidelity, but low physical and dynamic fidelity. This simulator, called FAIL, was used in a training program in which problems based on realistic failures were used (Su & Govindaraj, 1986; Govindaraj & Su, 1988). Trainees using FAIL appeared to follow two distinct stages of troubleshooting: hypothesis formation and hypothesis evaluation. During the diagnostic process, they seemed to use symptom knowledge together with hierarchically organized system knowledge in a combination of backward and forward reasoning strategies. Qualitative descriptions of system states along with compiled or chunked knowledge were used for troubleshooting.

Even when simulators have reasonably high levels of fidelity, major portion of the operating costs in simulator training results from the need for expert instructors. If simulators can be made more intelligent, substantial cost reductions can be achieved while increasing training effectiveness. This can be accomplished by developing simulators at moderate levels of fidelity, and integrating intelligent tutors as part of the simulator design. A moderate fidelity marine power plant simulator that we have developed and integrated with an intelligent tutor and experimental results to evaluate its effectiveness for training are described in this report.

**Overview**

This report begins with a discussion of a qualitative approximation methodology for the design of complex dynamic system simulators and a description of a marine power plant simulator implemented using qualitative approximation. Architecture for an intelligent tutoring system and some implementation details of the tutor are described next. In the section that follows, impor-
tance of interactive interfaces for tutoring and features of the student-tutor interface are discussed. Details of the experiment to evaluate the tutoring system are provided next, followed by a discussion of the results. The report concludes with a summary of accomplishments and observations.

A MARINE POWER PLANT SIMULATOR VIA QUALITATIVE APPROXIMATION

General

In this section, a qualitative approximation methodology for the design of moderate fidelity simulators is described. In simulators using qualitative approximation, the system states are represented by qualitative measures such as “pressure low” and “flow rate has been steadily decreasing”. Exact numerical values are not used. The qualitative state representation aids the human operator by eliminating the need to compare observed state values to nominal values.

Also, large systems can be simulated with a moderate amount of computational power due to reduced computational requirements. Basic principles of the design methodology were developed during previous research supported by ONR (Govindaraj, 1987). The current effort enhances these principles by extending them to more general dynamic systems and implementing them in a marine power plant simulator.

The design principles of qualitative approximation are illustrated with reference to a marine power plant with approximately 500 components. The resulting simulator has a moderate level of physical fidelity, and fairly high levels of dynamic, structural, and temporal fidelity. Physical fidelity is concerned primarily with the representation of the control and display interface, and ambient conditions such as noise, vibration, temperature, and humidity. Dynamic fidelity refers to the evolution of system states over time, including the accuracy and the proportion of the total number of states represented. Structural fidelity measures the completeness with which the various components and subsystems are represented. Temporal fidelity is, in a strict sense, a subset of dynamic fidelity, in that it is concerned with the sequential order in which events occur and states evolve. Temporal fidelity is especially important in a qualitative representation of dynamic systems, since even though exact state values are not important, a plausible sequence of state evolution is important.

Hierarchies within a dynamic system

The simulator design methodology is based on a hierarchical description of the system. System components are grouped into a number of subsystems based on their function. For instance, an oil-fired steam power plant on a ship is comprised of the following primary subsystems: fuel oil, feedwater, steam, lube oil, and control air. Some components might belong to more than
one subsystem. For example, the condenser is part of the feedwater subsystem as well as the steam subsystem. Components are classified into a number of generic types, which are then broken down into a small number of primitives. A condenser as well as an economizer, therefore, can be classified as heat-exchangers. This is a rather simple arrangement or design of the hierarchy based on the physical nature of components that form the system.

The primitives form the basic units in the qualitative approximation method. The primitives are the simplest form of components performing a single operation or a function, e.g., providing a path for some fluid in the case of a conduit. Primitives defined in this methodology include: conduit, source, sink, heat exchanger, and resistor. A component such as a condenser can be broken down into two sets of sources and sinks, gains and conduits, and a transfer agent. These hierarchical descriptions follow the natural arrangements of various components and subsystems in the real system.

**Description and evolution of the system states**

The most significant part of the modeling process in simulator design is the qualitative description of the state space. The states are represented as deviations from their nominal values. This technique, commonly used in modeling dynamic systems, is called the perturbation approach. The key difference from traditional applications, however, is that in the simulator design methodology described here, the perturbed states evolve using approximate functional representations rather than exact representations of the primitives.

Each of the primitives has a structure and a set of parameter values. Structures of the primitives are based on approximate functional equations of system dynamics. The structure, characterized by appropriate differential and algebraic equations, is the same for a primitive regardless of the component which it represents. The parameter values depend on the component of which a particular primitive is a part. Parameters associated with the primitives of a component are tuned to maintain temporal fidelity of state evolution.

System state is updated in a two-step process: during the first step, the states of individual components are updated; during the second step, the updated states are propagated to successor components. Numerical values corresponding to deviations from nominal values are used to represent the states in the simulation. Since these numerical state values are derived from functionally approximate system equations, they represent system states only qualitatively. The state values are transformed into qualitative descriptions, e.g., pressure low and level high, before presenting them to the operator.

An approximate, qualitative representation of system states enables the simulator to maintain cognitive compatibility with trainees. The simulator is said to be cognitively compatible with
its human operator when the qualitative states are similar to state descriptions used by the human. Humans often use qualitative descriptions of system states, e.g., pressure is low and temperature is fluctuating, rather than specific values, e.g., the pressure is 1150 psi. The simulator uses similar states; in training, there is no need for an extremely precise numerical state description. Although the simulation evolves qualitatively, temporal fidelity is maintained since the sequence of state changes that occurs as a result of an event is the same as it would be in a real system.

Implementation of marine power plant simulator

Qualitative approximation methodology was used to design a marine power plant simulator (Govindaraj, 1987). The simulator was based on a hierarchical representation of subsystems, components, and primitives together with necessary physical and logical linkages among them. This simulator, called QSTEAM, was implemented on a Xerox 1100 series LISP machine and provided low physical fidelity. High degrees of structural and dynamic fidelity, however, were achieved.

We have implemented the current version of the simulator, called Turbina, in Allegro Common Lisp (now Macintosh Common Lisp) with Common Lisp Object System (CLOS) and runs on Apple Macintosh II family of computers (Vasandani & Govindaraj, 1989, 1990). This new implementation was dictated by our desire to build the entire system on a widely available computer that is also more easily affordable. The basic hierarchical approach, characteristic of qualitative approximation, was retained. CLOS was used to represent all important entities as objects, including knowledge of the domain and troubleshooting knowledge. Modeling the entities as objects results in a very robust, modular implementation that is easy to maintain and evolve. Mouse-based direct manipulation graphical interfaces were used for the simulator and the tutor. Details of the interface are described in a later section. More complete descriptions the entire system are given in Vasandani (1991).

This section described a qualitative approximation methodology used for designing training simulators of complex dynamic systems. A marine power plant simulator was designed using this methodology. An intelligent tutoring system was used along with the simulator for training Naval ROTC cadets to help develop good problem solving skills. Details of the architecture for the intelligent tutoring system are given in the next section.

ARCHITECTURE FOR AN INTELLIGENT TUTORING SYSTEM

Although work on intelligent tutoring systems have been in progress for over two decades, computer power and developments in cognitive science and ITS research have not been suffi-
ciently harnessed for applications in complex dynamic engineering domains. Among the extensive surveys found in Sleeman and Brown (1982), Wenger (1987), and Psotka, Massey, and Mutter, (1988), only a few deal with engineering domains. STEAMER (Hollan, Hutchins, & Weitzman, 1984), IMTS and its successors (Towne & Munro, 1988), AHAB (Fath, Mitchell, & Govindaraj, 1990), and SHERLOCK (Lesgold, Lajoie, Bunzo, & Eggn, 1989) are some examples that have useful applications in an operator training program.

In general, ITSs have an expert module, a student module, and an instructional module (Sleeman & Brown, 1982; Wenger, 1987; and Psotka et al., 1988). In addition a simulator provides the training environment. The expert module of an ITS contains the domain expertise which is also the knowledge to be taught to the student. The student module contains a models of the student’s current level of competence. 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 the knowledge to the student can be treated as a separate component of the ITS.

We describe below an architecture for an ITS suitable for training operators of complex dynamic systems. Figure 1 illustrates the major components of the instructional system. 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. The instructional system has two major requirements: (1) a domain simulator and (2) organization of knowledge that supports the functions of the three major elements of the tutoring system. A brief discussion of the simulator preceded this section. Details of knowledge organization are discussed below.

Representation of knowledge for tutoring

Operators of complex dynamic systems, in which interdependent subsystems have some level of autonomy, 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 turbo-generator. 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 intelligent tutoring system must be capable of organizing and presenting knowledge about the system and troubleshooting task knowledge at several levels of granularity or detail. Pedagogical knowledge concerning instructional strategies is also necessary. A brief discussion of these three components of knowledge follows.

System knowledge for an intelligent tutor can be represented in several ways. These representations differ in the levels of description or detail depending on the task for which the knowl-
Figure 1. Instructional System
edge is required (Rasmussen, 1986). For most large, complex, mechanical systems, such as steam power plants, a comprehensive representation of system or domain knowledge is via **functional subsystems, fluid paths and schematics**. A functional subsystem is a collection of components responsible for performing a higher level system function such as combustion in a power plant. Fluid paths are a way of visualizing the system as a collection of different fluids. The fluids may be fuel, steam and air in a steam power plant, or electric current in an electrical circuit. A schematic is a pictorial representation of the system structure through its components and gauges. For large systems, a pictorial representation usually spans several schematics. Although the three representations are complementary rather than mutually exclusive, multiple representations of the system knowledge are necessary to enhance diagnostic performance (Rasmussen, 1986).

Each of the three system representations is an aggregation of mechanical components. System knowledge at the component level concerns a component’s **structure, function and behavior** at a level of detail necessary for successful fault diagnosis. A component’s structure refers to its input and output connections to other components, the fluids carried by it, the gauges attached to it, and its association with a functional subsystem. A component’s function depends upon its structure and defines the purpose of the component in the system and its contribution to the higher level system functions. The manner in which the system state values are affected by the presence of a component in both the normal and failed states constitutes behavioral knowledge.

System knowledge, although essential, is not adequate for diagnostic problem solving. **Troubleshooting knowledge**, in addition to system knowledge, is equally important. Troubleshooting knowledge includes general information about modes of failure in components, detailed information on certain common failures, cause-effect associations for familiar failures, and behavioral information about components and subsystems under different failure conditions (Govindaraj & Su, 1988; Fath et al., 1990).

Troubleshooting task knowledge, like the system knowledge, can have several representations. However, the representation most suitable for expert performance is not necessarily the best representation for a tutor (Clancey, 1987). The tutor’s troubleshooting task knowledge must be organized in a manner that facilitates inference of possible misconceptions based on observed actions. Therefore, expressing troubleshooting task knowledge for each failure in terms of valid student actions at the student-tutor interface is useful for a tutor. Such a representation provides a **normative task model** of troubleshooting and offers a convenient way to interpret and analyze student actions.

Knowledge requirements of an intelligent tutor also include knowledge to evaluate and rectify misconceptions. When a student’s actions deviate from the normative task model of troubleshooting, they may be explained in terms of misconceptions related to system structure, func-
tion or behavior. Knowledge that relates a deviant action to a probable misconception and the knowledge to rectify the misconceptions are both essential elements of the instructional system. An efficient way to represent such a knowledge is in the form of rules.

Finally, pedagogical knowledge concerning instructional content, form and time of presentation must also be addressed in the design of the instructional system. Instructions can be provided with or without intervention. Instructions with intervention must be provided during the session while those without intervention are provided at the end of a training session. Intervention at critical stages of the training period is an effective way of emphasizing a point; non-intervention has the advantage of not interfering with the thought process of the student.

System, troubleshooting and pedagogical knowledge are important for an instructional system. However, knowledge alone is insufficient to make a tutor effective. It must be appropriately structured and implemented so that the tutor can access relevant pieces of knowledge correctly and promptly as the need arises. Certain details concerning how the knowledge described above is implemented are described next.

Knowledge representation in Turbinia-Vyasa

Turbinia-Vyasa is an instructional system that trains operators to troubleshoot marine power plants. Turbinia\(^1\) is the name of the simulated marine power plant used in the instructional system and Vyasa\(^2\) is the computer-based tutor that teaches the troubleshooting task using Turbinia. This instructional system has been implemented on a dual screen Apple Macintosh II computer in Allegro Common Lisp with object-oriented extensions.

The tutor uses objects to encapsulate knowledge about the system and the failures. This knowledge is represented in declarative as opposed to procedural form and is amenable to changes. For example, components in the power plant that are instances of the same functional primitive have similar representations and share methods that create and manipulate data. Examples of some abstract data types used for representing knowledge are described below.

**System:** System knowledge is decomposed into nine subsystems, thirteen fluid-paths, and seven schematics. Knowledge concerning these functional subsystems, fluid-paths, and schematics is represented in instances of class objects defined for each. Table 1 shows how knowledge concerning combustion subsystem is represented.

---

1. Turbines were first used in marine propulsion by Sir Charles Parsons in 1897 in the *Turbinia*. It was an experimental vessel of 100 tons, fitted with turbines of 2,100 hp driving three propeller shafts. *Turbinia* attained the then record speed of 34.5 knots (Burstall, 1965, p.340).
2. Ancient Indian sage, scholar, and teacher.
Functional-subsystem:

*combustion-subsystem* is an instance of subsystem

<table>
<thead>
<tr>
<th>subsystem-name</th>
<th>combustion-subsystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-schematics</td>
<td>(fuel-oil-schematic boiler-schematic)</td>
</tr>
<tr>
<td>fluids</td>
<td>(flue-gas combustion-air fuel-oil steam)</td>
</tr>
<tr>
<td>components</td>
<td>(...)</td>
</tr>
<tr>
<td>connectors</td>
<td>(...)</td>
</tr>
<tr>
<td>function</td>
<td>“To mix the combustion-air with fuel and ignite it in the burner to release thermal energy”</td>
</tr>
<tr>
<td>steam-schematic</td>
<td>nil</td>
</tr>
<tr>
<td>boiler-schematic</td>
<td>subsystem structure within the schematic components: (...) connectors: (...)</td>
</tr>
<tr>
<td>feedwater-schematic</td>
<td>nil</td>
</tr>
<tr>
<td>fuel-oil-schematic</td>
<td>subsystem structure within the schematic components: (...) connectors: (...)</td>
</tr>
<tr>
<td>control-air-schematic</td>
<td>nil</td>
</tr>
<tr>
<td>saltwater-schematic</td>
<td>nil</td>
</tr>
<tr>
<td>lube-oil-schematic</td>
<td>nil</td>
</tr>
</tbody>
</table>

Table 1. Description of combustion subsystem

**Components:** All components are instances of subclasses of either simple or composite primitives. There are nine subclasses of simple and one subclass of composite primitive defined for the system. There are methods for each subclass of primitives that update the system state across each component.

**Failure modes:** The system behavior associated with component failure modes is described in instances of failure-mode class of objects. There are four failure-mode objects, one for each of the four failure types: blocked-shut, stuck-open, leak-in, and leak-out. Knowledge about a failure mode includes information about the upstream and downstream system behavior along the affected fluid-path.

**Failures:** Knowledge of the individual faults in the component is stored in objects of class component-fault. Knowledge of specific faults includes information concerning affected schematics, subsystems, fluid-paths, and gauges along with explanations of cause-effect associations.

Knowledge concerning evaluation and rectification of student’s misconceptions is represented

Knowledge organization that captures system structure, function, and behavior and troubleshooting task knowledge are insufficient for the success of a tutoring system. Properly designed interactive interfaces can also play a major role in imparting knowledge about the system and its operation during normal and abnormal situations. In the next section, we describe the importance of such interfaces and how a student interacts with the instructional system.

INTERACTIVE INTERFACES AND STUDENT-TUTOR INTERACTION

Direct manipulation, interactive, graphical interfaces

A good interface makes the knowledge of the tutor transparent to the student and helps 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).

In a diagnostic problem solving task, a set of schematics often serve as a convenient student-tutor interface for knowledge communication. 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 appear as manipulable objects. Other factors such as grouping of components and graphics also influence the design of these schematics.

Grouping of components in 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 et al., 1984). Graphical objects 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 tur-
bine. 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 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. 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 has to 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 design involves getting the student to understand and correctly respond to the queries. Where a student can respond to a query in multiple ways, the student options have to 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.

The issues related to the design of interfaces described above have been implemented in an intelligent tutor for diagnostic problem solving. Implementation details of the student-tutor inter-
face designed to communicate its knowledge to the student are discussed next.

Student-tutor Interface and Operator Interaction

The interface to Turbinia-Vyasa consists of seven schematic windows and a large number of dialogs and menus that establish communication between the student and the tutor. A single button computer mouse is the only input device used to interact with this direct manipulation interface.

The seven schematics display the physical connections between the components of the power plant. Figure 2 is an example of the boiler schematic. The student can access these schematics through the seven icons displayed in a schematic menu. The schematics are used to investigate components and probe gauges attached to these components. Investigating a component involves moving the cursor on to the component and clicking the mouse button. The same action is also used to indicate diagnosis, pick the component for which information is desired, and indicate hypotheses. Thus, identical student actions on a schematic have different responses. The actual response depends upon the mode of the system in which the interaction occurs. There are four system modes in which interaction occurs: troubleshooting mode, diagnose mode, student-initiated tutor dialog mode and tutor-initiated dialog mode. Except for the tutor-initiated dialog mode, the student is responsible for switching between the remaining three modes of the system. The current mode of the system is indicated by a highlighted icon representing the mode and also by the shape of the cursor.

Student-initiated interaction with the tutor is accomplished by clicking on either the stop icon in the requests menu or a menu item in the hypothesis menu (see Figure 2). This action halts the simulation temporarily, enabling the student to interact with the tutor while preserving the information concerning system states.

When the tutor is invoked using the stop icon in the requests menu, the student can explore the tutor’s knowledge-base. The student’s exploration is guided by a set of hierarchically organized dialog menus that reflects the inherent system hierarchy. 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 or graphical form.

The interface offers substantial flexibility to the student and maintains context-sensitivity. For example, when the student inquires about fluid-paths following an inquiry about a subsystem, the system first offers a choice of selecting a fluid-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 independently to deduce which fluid-paths lie in which subsystems.
Figure 2. Boiler Schematic and Menus
The student can also invoke the tutor by selecting any of the four items on the hypothesis menu (see Figure 2). The first item, “View”, is used by the student to review his own hypotheses concerning failure. The “Add” and “Delete” items are used by the student to modify the hypotheses. “Advice” is used to seek assistance from the tutor in refining current hypotheses. The assistance provided is in the form of diagnostic tests that can strengthen or weaken an indicated hypothesis. This helps the student develop an association between failures and its effect on system behavior.

The tutor intervenes to initiate communications with the student in three situations. First, the tutor intervenes when it becomes necessary to notify the student about an error. The error could be an incorrect diagnosis or an invalid action. Second, the tutor intervenes when a possible misconception concerning the system’s structure, function or behavior is identified based on the student’s actions. When possible misconceptions are identified, the tutor provides canned, but context-sensitive instructions to rectify the student’s misconceptions. Third, the tutor sometimes prompts the student for an action or requests for hypotheses concerning failure. In all the three situations, the tutor’s communication begins with a beep to draw the student’s attention.

In this section we described the importance of interactive interfaces and details concerning student-tutor interaction. Experiments were conducted to evaluate the tutoring system and to determine the benefits of using the instructional system in addition to a simulator for training operators in troubleshooting dynamic systems. Details concerning the experimental evaluation are discussed next.

EXPERIMENTAL EVALUATION OF THE TUTORING SYSTEM

Prior to beginning formal experiments to test the effectiveness of the tutor, the tutor was evaluated with the help of a naval officer who is also an ROTC instructor experienced with steam power plants. This evaluation was informal and somewhat subjective, and was aimed at testing the aiding material and instructional strategies. A set of formal experiments designed to measure the diagnostic performance of operators trained with and without the aid of Turbinla-Vyasa followed. The discussion in this section deals with the details of the evaluation.

Checking for consistency and correctness

The primary goal of the subjective evaluation was to ensure that the instructional material presented to the student was technically correct, properly stated and consistent with the current training program for engineers in the US Navy. A secondary goal was to gather suggestions for improving the interface to Turbinla and Vyasa.
During the subjective evaluation, the experimenter solved problems on Turbinia-Vyasa while a subject matter expert, a Naval ROTC instructor, observed the interactive performance feedback from the tutor. The instructor was requested to report any inconsistencies that he observed. He was also asked to make suggestions and comments concerning the design of display and operator interaction. Notes were made of the changes suggested. These notes were later discussed in detail. Following the discussion, several of the suggested changes were incorporated. However, the most useful outcome of this analysis was the reaffirmation of the experimenter’s faith in the technical validity of the material presented to the student.

At the conclusion of this evaluation, both the subject matter expert and the experimenter were confident that the students would be receptive to Turbinia-Vyasa’s tutoring strategy and that the instructional system would be a worthwhile contribution to the Naval ROTC training program. Formal experiments were started after this evaluation.

Formal experiments

In the formal experiments, performance of subjects trained with and without the tutor was compared. There were two goals in the experiment: (1) determining the effectiveness of the tutoring architecture and methods for knowledge representation and (2) establishing the usefulness of computer-based training programs over traditional means of training operators to troubleshoot complex dynamic systems. In addition, the experiment provided an opportunity for comparing the effect of passive and active tutoring strategies. The above goals can be formulated in terms of the following questions: (1) is it feasible to build an effective computer-based tutor by implementing the proposed architecture, (2) is the training by computer-based tutor better than the training provided by the simulator alone, and (3) how does the level of aiding during the course of training affect performance. Details of the experiment to find answers to these questions are discussed below.

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; (b) training with the aid of a passive tutor; and (c) training with the aid of an active tutor. During data collection, trained subjects from all three conditions attempted to solve the same set of problems unaided by the tutor. A complete description of the experimental design follows.

Experimental design

There were two main factors of interest in the experiment: training condition and seen status of the problem. There were three training conditions: unaided simulator, aiding with passive tutor, and aiding with active tutor. The first was a baseline condition where training was provided using just Turbinla. The second and the third training conditions used the computer-based tutor
Vyasa. In the second condition Vyasa functioned in the passive mode. It functioned in the active mode in the third condition.

The second factor of interest was the seen status of the problem. We anticipated that this factor would influence performance and therefore should be investigated. Seen status had three levels: seen once, seen twice and unseen. Seen once status applied to problems that were seen once before by the student during training. Similarly, seen twice status applied to problems that were seen twice during training. Unseen status referred to problems that were not seen by the student until the test phase. Since the instructional system was expected to train for not just familiar situations, the effect of seen status was important to analyze the transfer of training from familiar to unfamiliar situations.

Subjects and problems were two other factors that could account for variations in the experimental data. While subjects were nested within training condition, problems were nested within seen status. Subjects and problems along with the main factors and their interactions make a complete list of sources of variation considered in this experiment.

Training condition and seen status were fixed factors and subjects (nested within training condition) and problems (nested within seen status) were random factors. Therefore, a mixed model was used to analyze data from the experiment.

Equipment

For the experiment, Turbinla (simulator) and Vyasa (tutor) were installed on a dual screen Apple Macintosh II workstation with a 40MHz accelerator board. This computer was used for all data collection sessions. An Apple Macintosh II cx was used for few sessions during initial stages of training. Training instructions for the three experimental conditions were tape-recorded in a female voice on separate tapes to be played back to the subjects during their first training session.

A pilot study

The actual experiment was preceded by a pilot experiment. The purpose of the pilot experiment was to validate the instructional material and to determine software errors that may have gone undetected. The pilot study thus offered the experimenter an opportunity to evaluate the software under experimental conditions.

Four graduate students from Georgia Institute of Technology participated in the pilot study. All four came from engineering backgrounds, had several courses in thermodynamics and were well exposed to research issues related to human-machine systems.
The pilot study was conducted with Turbinia-Vyasa operating in the active mode. Since the simulator and the passive tutor were also functional in this mode they were not subjected to separate pilot studies. Each of the four pilot subjects participated in four sessions. In the introductory session, the subjects read the instructional manual to become familiar with the Turbinia-Vyasa interface. Then, using the instructions, the subjects attempted to solve a single problem designed specifically for the first session. In the subsequent sessions, each subject saw four problems in every session.

The pilot study identified several discrepancies in the instructional manual and errors in software. These were corrected promptly. Furthermore, five problems out of the twenty-nine supplied by the Navy were eliminated from the set to be used in the full-scale experiment because they were either redundant or exhibited inconsistent behavior. Of the remaining twenty-four problems, those that had software errors were modified, sometimes more than once, and resolved by the pilot subjects till the error was corrected.

Apart from detecting errors in the instructional manual and the software, the pilot study was also useful in estimating time taken to solve problems. This helped the experimenter design the appropriate duration of the introductory as well as subsequent training and data collection sessions.

At the conclusion of the pilot study, the experimenter was better equipped to answer questions from subjects participating in the full-scale experiment. This was important as answers to similar questions by subjects in the three experimental conditions had to be consistent and required advance preparation.

The experimenter could not get an initial estimate of the full-scale experimental results however, since the pilot subjects were not drawn from the same population as that of subjects that were to participate in the full-scale experiment.

Experiment

Thirty cadets from the Georgia Institute of Technology Naval ROTC unit participated as subjects in the experiment. All except one subject were male. Subjects were required to have a basic understanding of the theory of marine power plants. Therefore, sophomores and juniors who had taken the freshman-level course in Naval Engineering offered by Navy ROTC were considered. Among those cadets who volunteered, 24 sophomores and 6 juniors were selected. Although some of the subjects had additional exposure to thermodynamics through course work or had experience operating marine power plants, effects of these factors were not considered. Therefore, the assignment of subjects to the three experimental groups was done randomly.
Each subject was paid $6 per session for every session completed in both the training and the testing phase. 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.

Subjects were told about the performance measures prior to the experiment. (Performance measures are described in a later section.) They were also informed that the number of problems correctly diagnosed in minimum time was the only measure for the purpose of determining the award.

Experimental materials

There were separate written instructional manuals for each of the three experimental conditions. The manual for subjects using just the simulator (Turbinia) provided an introduction to the marine power plant, its automatic boiler control system, a description of the common modes of failure and a guide to make the subjects familiar with Turbinia's interface. For subjects using Turbinia-Vyasa in the passive mode additional instructions were included that described the features and the interface of the passive tutor. Further instructions describing the capabilities of the tutor were added to the manual for subjects using Turbinia-Vyasa in the active mode.

Other material used in the experiment included a subject consent form and a survey form. The survey form was designed to obtain a feel for the subjects' academic background, shipboard experience and computer skills.

Subjects were required to answer three written questionnaires at three key points in the experiment. The first questionnaire was used to evaluate the subjects' operating knowledge of the marine power plant, its components and their behavior under normal and failed states prior to the start of training. The second questionnaire was identical to the first and was used at the end of the training sessions. Answers to the two questionnaires provided the experimenter with some means to measure the knowledge acquired during training. Subjects answered the third questionnaire at the end of the last data collection session to provide subjective opinions about various aspects of the training methods.

In addition, the subjects were provided with pencil and paper in every session to take notes, if any, pertaining to the problems and to comment on the tutor and its strategies.

Experimental procedure

The experiment was conducted in two phases: training and data collection. In the training phase, three instructional methods were employed to train three groups of subjects to troubleshoot a simulated marine power plant. The effect of training was evaluated in the data collec-
tion phase where all subjects were exposed to identical problems on Turbinia without the aid of the tutor.

The subjects were randomly assigned to three groups of ten each. Group I was trained using just Turbinia, the simulator. Subjects in Groups II and III were aided by the computer-based tutor Vyasa. While Vyasa functioned in the passive mode for Group II, it functioned in the active mode for Group III.

Subjects in each of the three groups filled a consent form, completed a survey form and answered questions in Questionnaire 1 prior to the start of the experiment. The subjects were then given written instructional manual appropriate for each group. They were advised to read the instructional manual before starting the first session.

Subjects were distributed into three batches. Each batch was identified by the date on which it started training. Although the assignment of subjects to batches was done randomly, at least three subjects from each group were assigned to every batch. When the subjects from the first batch did not require further assistance in learning the system, a second batch started training on a second computer with fully compatible but slightly slower hardware. After the first batch completed the training and data collection sessions, the second batch was moved to the first computer. Similarly, the third batch started training on the second computer once that machine was available and the experimenter was no longer occupied with the second batch. They moved to the first computer when it became available. Thus, data collection sessions for all batches were done on the same machine. Even though we realized the possible impact of excluding the batching factor in the experimental design, we did not consider it worthwhile to study its effect.

Training phase

The first ten sessions were used for training of subjects in each of the three groups. Each session had a maximum duration of 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. All interactions between the subject and the interface for this first session were controlled through these instructions. The capabilities of the passive and active tutor were also demonstrated through a predetermined set of actions performed by the subject upon request. The experimenter was present for the entire duration of the first session to answer any questions. Variability of information shared
by the experimenter with the subjects was controlled as far as possible so that consistency could be maintained between subjects and across training groups.

After the first session, subsequent training sessions had three problems each. A subject had a maximum of 13 minutes to solve each problem. If the subject solved the problem early, 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 45 minutes.

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

For the first three training sessions in each group the experimenter was present to answer any questions. For the subsequent sessions the presence of the experimenter in the room was not considered necessary although he was still available to answer questions. It was only on a couple of rare occasions that the subjects sought any clarification from the experimenter past the third session.

At the end of each training session past their third, the subjects were asked to make subjective comments about the instructional system. Although the subjects were free to write anything they were encouraged to identify their “likes” and “dislikes” for the system. At the end of their last training session, the subjects were asked to answer Questionnaire 2.

Data collection phase

The data collection phase consisted of two sessions. These sessions were run on consecutive days immediately following the completion of training. During the data collection sessions, the subjects interacted with the simulator only, unaided by any tutor, irrespective of their training condition.

Each data collection session was approximately 50 minutes long and consisted of 5 problems. If the subject solved the problem within the 10 minute period allocated for each problem, the next problem was presented immediately. However, unlike the training sessions, no solution was provided to the student at the end of each problem. At the end of the data collection sessions, each of the subjects completed an “exit” questionnaire.

Training and test problems

Twenty-four problems were identified for use in the experiment. Since we wished to study the effect of training methods on performance, for familiar as well as novel situations, 5 out of the
24 problems were exclusively reserved for the data collection phase.

For the training phase comprised of 1 single-problem session and 9 three-problem sessions, a total of 28 problems were required. With 5 of the 24 problems reserved for test sessions, there were only 19 available for use in training. Therefore, 9 problems were shown twice to the subjects during training. Selection of these 9 problems was done randomly. However, the same problem was never presented the second time within a span of 3 consecutive sessions.

In the test sessions, in addition to the 5 unseen problems, subjects were given 5 problems from among those seen during training. Three of these problems appeared twice during training and two were seen only once.

The order in which problems were presented during training and test was identical for all groups. In the test sessions, seen and unseen problems were alternated beginning with an unseen problem. The actual order in which the problems appeared was, however, determined randomly.

Performance measures: Dependent variables

Although the ultimate goal in troubleshooting is to successfully identify the failed component, there were several other performance measures considered in the experiment. This section describes the measures used to evaluate the troubleshooter's performance. Subject actions were recorded for computing the performance measures. These measures were obtained directly or derived from the data.

Number of problems solved: Successfully diagnosing a fault is an important measure of troubleshooting ability. Since there is a time limit imposed on solving the problems, a problem is considered solved if a correct diagnosis is made within the time (10 minutes) allocated for each problem in the test sessions.

Solving a problem correctly is not sufficient, however. In real world there are costs associated not only with the troubleshooter's inability to solve problems but also with how the diagnosis was made. Therefore, efficiency of the diagnostic process must also be considered. The remaining performance measures focus on efficiency of troubleshooting.

Troubleshooting time: For those subjects who were successful in solving the problems, the total amount of time taken for solution is a valuable measure of their performance rating. Those who take less time are considered better troubleshooters.

Number of informative actions: A student may take many actions at Turbinia's in-
terface. However, not all actions are informative. Only those actions that are taken to obtain gauge readings are informative since they alone can help the students access the system state information needed to solve the problems. Therefore, the total number of such informative actions provides a measure of the student’s overall troubleshooting ability. The smaller the number of informative actions taken to solve a problem, the better is the diagnostic performance.

**Percentage of relevant informative actions:** Even though every informative action has some information content, only some are directly relevant to the failure. Since the total number of relevant informative actions necessary to solve a problem is dependent on the problem, the percentage of informative actions that are relevant for a failure is a better measure.

**Nature of diagnosis:** In order to isolate a failed component it is necessary to conduct diagnostic tests that eliminate other probable hypotheses. Due to the limited availability of gauges in the system, the troubleshooter may not be able to isolate the fault completely. However, for each of the failures, there are some gauges that are affected and must be checked to justify pursuing that hypothesis. If a student correctly solves a problem but has not gathered sufficient evidence to do so, the diagnosis is considered premature. On the other hand, if the problem has been seen before then it may not be necessary to gather all evidence before correctly identifying it. However, while attempting to solve a seen problem a student may make several incorrect diagnoses. If these diagnoses suggest hypotheses that are not probable, then the final diagnosis is still considered premature. The rationale for calling such a diagnosis premature is that if a student incorrectly diagnoses a seen problem, then further investigations for that failure should proceed along the lines of an unseen problem.

There may also be times when the student is unable to diagnose the fault even after sufficient evidence implicating the failed component has been gathered. This indicates the student’s inability to integrate the diagnostic information and make effective use of it. Such diagnoses are termed as overdue.

Finally, when the student integrates diagnostic information properly, the diagnosis is neither premature or overdue, and hence is considered timely. Categorizing diagnoses as premature, timely or overdue provides a subjective measure of diagnostic performance.

**Percentage of guesses:** At any time during the troubleshooting process there are likely candidates for the failed component, based on the observed abnormal system states. The likelihood that a component may have failed increases or decreases as more diag-
nostic tests are conducted. While quick diagnosis of a problem saves time and money, incorrect diagnosis costs additional time and money. Even so, selecting a likely component as the cause of abnormal system behavior is not as bad as picking a component that cannot have failed. Thus, an incorrect diagnosis that implicates a component that could not have failed, based on observed symptoms, is a consequence of pure guesswork or inaccurate troubleshooting knowledge. Such an incorrect diagnosis is considered a guess and fewer guesses indicate better troubleshooting performance. Since the number of incorrect diagnoses, in terms of guesses and probable hypotheses, may depend on the problem, percentage of diagnosis for each problem that were guesses is a reasonable measure of troubleshooting performance.

Number of unaffected schematics/subsystems/fluid-paths investigated: For each failure, only a few schematics, subsystems, and fluid-paths are affected. Affected schematics are those that have gauges with abnormal readings. Investigating components in schematics that are unaffected by the failure reflects the student’s inability to relate symptoms to the structural location of the power plant. Thus, investigating components in unaffected schematics is undesirable and the number of such schematics wrongly investigated is a measure of performance.

Likewise, investigating unaffected subsystems and fluid-paths reflects the student’s inability to relate symptoms to the functional location of the power plant. The number of subsystems and fluid-paths wrongly investigated in this manner are also measures of performance. The fewer the number of unaffected subsystems or fluid-paths investigated by the student in solving a problem, the better is the performance.

The dependent variables listed above are closely related to the troubleshooting strategy that the student is likely to follow. We expected that those trained on the simulator alone would perhaps develop an unguided search strategy to locate the fault. Therefore, they are likely to make more guesses, take a lower percentage informative actions that are relevant, make more premature diagnoses and investigate more unaffected schematics, subsystems and fluid-paths. However, since they would rarely be using concrete reasoning, they would perhaps solve the problem in less time but with many attempts of incorrect diagnosis. Furthermore, it was anticipated that comparing the performance of those trained with the passive and active tutors might reveal individual differences in performance.

In addition to the performance measures described above, a subjective evaluation of the instructional system was also performed. The exit questionnaire (#3) was designed for this purpose. Results of analyzing this questionnaire and an analysis of the data collected from the experiment are discussed in the next section.
EXPERIMENTAL RESULTS

Analysis of the experimental data revealed that subjects in the unaided group developed a troubleshooting strategy distinctly different from those of subjects that were aided by Vyasa during training. The unaided group did not devise any good and consistent troubleshooting strategy and often relied on unguided search for the failure. As such, they conducted a large number of diagnostic tests and very few of these tests were relevant to the failure. In the absence of a guided strategy and in their pursuit for a quick diagnosis, the unaided group provided a large number of incorrect diagnoses and investigated a large number of unaffected schematics, subsystems and fluid paths.

On the other hand, the two groups aided by the tutor hypothesized probable failures and conducted diagnostic tests to either strengthen or weaken their hypotheses. Most of the informative actions they took were relevant to the failure. In spite of an incentive to solve problems quickly, the aided groups did not indulge in much guesswork. Most of their guesses appeared to result from panic that set in when they were running out of time. Also, since their investigations were more focussed on hypotheses that explained observed abnormal behavior, the aided groups investigated fewer unaffected schematics, subsystems and fluid paths.

While subjects in each group solved approximately the same number of test problems, on an average the unaided group solved problems in shorter time. This was not at all surprising since this group relied heavily on guessing. Thus, their good performance based on quick diagnosis of failures was offset by the numerous incorrect diagnoses for each problem.

For both the aided and unaided groups, performance on most measures was better with seen than unseen problems. Among seen problems, those seen twice were often better recalled by all groups, indicating that practice helped subjects develop symptom-cause associations. The two aided groups performed better than the unaided group with unseen problems indicating that the training they received was successfully transferred to unfamiliar situations as well.

There were slight differences in the performance between the two aided groups. Those trained with the active tutor made a smaller number of guesses and hence fewer premature diagnoses, but they also solved a smaller number of problems. Most of the variation in the performance between the two aided groups seems to have been caused by individual differences. In the active tutor group, there were a few subjects who became somewhat dependent on the tutor; their performance suffered once the aid was withheld in the test sessions. There were others in both aided groups that became overly conservative and often investigated and eliminated less likely alternatives as well. This may not necessarily be a bad troubleshooting strategy, since incorrect
diagnosis are costly. However, any delay in diagnosing the fault could also be costly. Such accuracy-time trade-offs stem more due to individual preferences rather than the instructional strategy used for training.

Discussion of results

From an analysis of the results, it is clear that the tutor in both the passive and the active modes helped the students to develop useful troubleshooting strategies. The tutor was also helpful for forming plausible failure hypotheses based on observed symptoms and for systematically eliminating 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.

In summary, results of the experiment show that a simulator alone is inadequate. A simulator in conjunction with a computer-based tutor can help develop efficient troubleshooting skills. However, not all students are equally receptive to every tutoring strategy and provisions must be made in training programs for individual preferences and differences in abilities and styles. A more complete and thorough analysis of the experimental data and results will appear in Vasandani (1991).
CONCLUSIONS

The research program described in this report has contributed to the field of training for diagnostic problem solving in realistic, complex dynamic system domains. We have shown the viability of designing and implementing an effective intelligent tutoring system for supervisory control operation. Instructional systems that integrate an ITS with a simulator and provide access via direct manipulation graphical interfaces can contribute greatly to an effective training program. We hope that our research results will stimulate further research and development of tutoring and training systems and implementation within the navy and elsewhere. As a result of our research, we have gained a better understanding of diagnostic problem solving in complex engineering domains. In addition, we have developed good insights and expertise in modeling large dynamic systems, knowledge representation for tutoring, and interactive interfaces. To benefit from our experience and expertise, we are planning a program of research to develop models of expertise that will further enhance tutoring and training systems and intelligent operator associates.

ACKNOWLEDGMENTS

We wish to thank the commanders and the officers of the Georgia Tech Naval ROTC unit for their cooperation in providing access to their cadets and for technical help. Lt. William A. Marriott was generous with his time and helped us with many technical matters. We thank the thirty volunteer cadets for participating as subjects in our experiment. We are grateful to Dr. Susan Chipman, Contract Monitor, for her critical comments and suggestions on methodological issues relevant to cognitive science and training throughout the duration of the project and for her efforts to make our research known to various naval units. The comments and suggestions of Drs. S. Collyer, H. W. Sinaiko, and W. Vaughn at briefings during various stages of the project were helpful. Dr. Christine Mitchell at Georgia Tech deserves our thanks for her help with many design and implementation issues and with statistics.
REFERENCES


As of 11/1/90
One copy to each addressee except as noted:

Director, Contract Research Department
Office of Naval Research (Code 11)
Arlington, VA 22217-5000

Chairman, MPT R&D Committee
Office of the Chief of Naval Research
Code 222
Arlington, VA 22217-5000

Program Manager, Operations Research (Code 1111MA)
Office of Naval Research
Arlington, VA 22217-5000

Director, Life Sciences (Code 114)
Office of Naval Research
Arlington, VA 22217-5000

Director, Cognitive & Neural Sciences (Code 1142)
Office of Naval Research
Arlington, VA 22217-5000

Cognitive Science (Code 1142CS)
Office of Naval Research
Arlington, VA 22217-5000

Perceptual Science (Code 1142PS)
Office of Naval Research
Arlington, VA 22217-5000

Defense Technical Information Center*
DTIC/DDA-2
Cameron Station, Building 5
Alexandria, VA 22314

CDR J. S. Hanna, Office of the Deputy Asst. Secretary of the Navy (Manpower)
5D800, The Pentagon
Washington, DC 20350-1000

Head, Manpower, Personnel, and Training Branch
Office of the CNO (Op-813)
4A478, The Pentagon
Washington, DC 20350-1000

Assistant for Manpower and Training
Office of the CNO (Op-911H)
5D772, The Pentagon
Washington, DC 20350-2000

Assistant for Planning and Technology Development
Office of the DCNO(MPT) (Op-01B2)
Department of the Navy, AA-1822
Washington, DC 20350-2000

Deputy Director Total Force Training and Education Division
Office of the DCNO(MPT) (Op-11B)
Department of the Navy
Washington, DC 20370-2000

R&D Coordinator, Attn: Jan Hart
Office of the DCNO(MPT) (Op-11K1)
Department of the Navy, AA-G817
Washington, DC 20370-2000

Deputy Director Military Personnel Policy Division
Office of the DCNO(MPT) (Op-13B)
Department of the Navy, AA-1825
Washington, DC 20370-2000

Head, Military Compensation Policy Branch
Office of the DCNO(MPT) (Op-134)
Department of the Navy, AA-2837
Washington, DC 20370-2000

*Note: 12 copies go to DTIC
Headquarters U.S. Marine Corps
Code MA
Washington, DC  20380-0001

Head, Leadership Branch
Naval Military Personnel Command
Attn: LT Gary Kent, NMPC-621
Department of the Navy, AA-1603
Washington, DC  20370-5620

Director, Research & Analysis Division
Navy Recruiting Command (Code 223)
4015 Wilson Boulevard, Room 215
Arlington, VA  22203-1991

Technical Director Silva
Attn: Dr. Kobus
Naval Health Research Center
P.O. Box 85122
San Diego, CA  92138-9174

Head, Human Factors Division
Naval Training Systems Ctr. (Code 26)
12350 Research Parkway
Orlando, FL  32826-3224

Naval Training Systems Center
ATTN: Dr. Eduardo Salas, (Code 262)
12350 Research Parkway
Orlando, FL  32826-3224

Naval Training Systems Center
ATTN: Dr. Robert Hays, (Code 262)
12350 Research Parkway
Orlando, FL  32826-3224

Technical Director
Navy Personnel R&D Center
Code 01
San Diego, CA  92152-6800

Director, Manpower Systems Dept.
Code 11
Navy Personnel R&D Center
San Diego, CA  92152-6800

Director, Personnel Systems Dept.
Code 12
Navy Personnel R&D Center
San Diego, CA  92152-6800

Director, Testing Systems Department
Navy Personnel R&D Center
Code 13
San Diego, CA  92152-6800

Director, Training Systems Dept.
Code 14
Navy Personnel R&D Center
San Diego, CA  92152-6800

Director, Training Technology Dept.
Code 15
Navy Personnel R&D Center
San Diego, CA  92152-6800

Director, Organizational Systems Dept.
Code 16
Navy Personnel R&D Center
San Diego, CA  92152-6800

Naval Ocean Systems Center
Command Support Technology Division
Attn: Mr. Jeffrey Grossman, Code 4402
San Diego, CA  92152-5000

Chairman, Dept. of Admin. Sciences
(Code AS)
Naval Postgraduate School
Monterey, CA  93943-5100

Chairman, Department of Operations
Research (Code OR)
Naval Postgraduate School
Monterey, CA  93943-5100

Director, Instructional Development and
Educational Program Support Dept.
Naval Education and Training Program
Management Support Activity
(NETPMSA)
Pensacola, FL  32509-5100
Academic Programs and Research Branch
Naval Technical Training Command
Code N6, NAS Memphis, Bldg. C-1
Millington, TN 38054-5056

Director, Defense Personnel Security Research and Education Center
Suite E, Building 455
99 Pacific Street
Monterey, CA 93940-2481

Technical Director
U.S. Army Research Institute for the Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Chief Scientist
Air Force (AFHRL)
Brooks AFB, TX 78235

Director, Manpower & Training Program Center for Naval Analyses
4401 Ford Avenue
P.O. Box 16268
Alexandria, VA 22302-0268

Library- Code 231
Navy Personnel R&D Center
San Diego, CA 92152-6800

Library
Naval War College
Newport, RI 02940

Chief, Survey and Market Analysis Division
Defense Manpower Data Center
1600 Wilson Boulevard, #400
Arlington, VA 22209-2593