Center for Human-Machine Systems Research

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

AD-A242 619

Intelligent Tutoring for Diagnostic Problem Solving in Complex Dynamic Systems

Vijay Vasandani

DTIC

Technical Report CHMSR-91-4

September 1991

Reproduction in whole or in part is permitted for any purpose of the United States Government.

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

Approved for public release; distribution unlimited.

91-14569

91 034

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1991 September	Technical

4. TITLE AND SUBTITLE

Intelligent Tutoring for Diagnostic Problem Solving in Complex Dynamic Systems

5. FUNDING NUMBERS

C: N00014-87-K-0482
PE: 0602233N
PR: RM33M20

6. AUTHOR(S)

Vijay Vasandani

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Georgia Institute of Technology
Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
765 Ferst Drive, Atlanta, GA 30332-0205

8. PERFORMING ORGANIZATION REPORT NUMBER

CHMSR-91-4

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
Cognitive Science Program (Code 1142CS)
800 North Quincy Street
Arlington, VA 22217-5000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

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

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Maintenance training for diagnostic problem solving in complex dynamic systems is carried out either on the job or on simulators. When simulators are used for training, their effectiveness can be improved by integrating intelligent tutoring systems (ITS) into the training programs. Research results from ITSs developed for simpler task domains are generally not very useful in complex engineered domains due to lack of appropriate knowledge representation techniques. The focus of our research is the development of a methodology for decomposing, organizing, and representing domain knowledge of complex dynamic systems for building functional computer-based intelligent tutors.

Using our knowledge representation methodology, we 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. Performance of the subjects was analyzed using measures such as percentage of premature and correct diagnosis and percentage of relevant and irrelevant diagnostic tests were used. Results show that a simulator alone is inadequate, whereas a simulator in conjunction with an ITS can help develop efficient troubleshooting skills.

14. SUBJECT TERMS

fault diagnosis; problem solving; training; maintenance; intelligent tutoring systems; intelligent computer assisted instruction; interactive learning environments; marine power plants; simulation

15. NUMBER OF PAGES

391

16. PRICE CODE

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

INTELLIGENT TUTORING FOR DIAGNOSTIC PROBLEM SOLVING IN COMPLEX DYNAMIC SYSTEMS

A THESIS
Presented to
The Academic Faculty of Graduate Studies

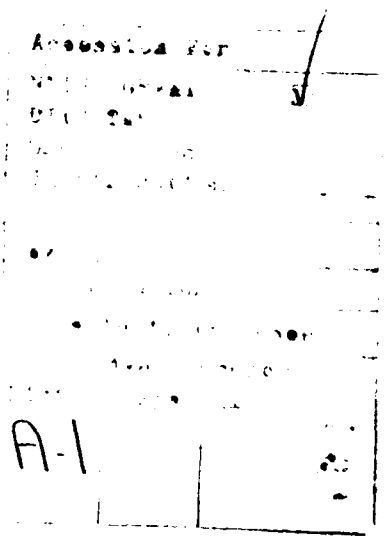
By

Vijay Vasandani

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in the School of Industrial and Systems Engineering

Georgia Institute of Technology
September 1991

Copyright © 1991 by Vijay Vasandani



Dedicated to
my parents, Padma and Vasdev
my brother, Sanjay
and my wife, Geetanjali
for their love and support

ACKNOWLEDGEMENTS

This research was supported by contract N00014-87-K-0482 from the Manpower R&D Program, Office of Naval Research (Dr. Susan E. Chipman, contract monitor). The author gratefully acknowledges the financial support provided by this contract which enabled him to pursue his dreams of higher education.

The author would like to thank Dr. T. Govindaraj, chairperson of his advisory committee, for the moral support and advice he provided throughout the duration of the author's stay at Georgia Tech. Dr. Christine Mitchell deserves a special thanks for her help with many design and implementation issues. Her insightful comments contributed a great deal towards improving the quality of this work. The author is also grateful to other members of his dissertation reading committee, Dr. Alex Kirlik, Dr. Ashok Goel and Dr. Bill Johnson for providing invaluable suggestions.

The author also expresses his gratitude to the faculty and staff of the School of Industrial and Systems Engineering. In particular, the author is indebted to Dr. R. Heikes for his help with statistics. A word of thanks also goes to Richard Robison for having the skills and untiring energy to keep the computers at the Center for Human-Machine Systems Research in good "health".

The author acknowledges the dedication of the thirty Navy ROTC cadets who voluntarily participated as subjects in the experiment and contributed as best as they could to make this research endeavor a success. The author is also thankful to Lt. Geoffrey L. Owen and Lt. William A. Marriot. They helped in getting the experiment organized. Lt. William A. Marriot's experience and knowledge of technical matters contributed to the progress of this report.

The author also wishes to express his appreciation for all his other colleagues at the Center for Human-Machine Systems Research: Jim Armstrong, Charlene Benson, Jim Bushman, Todd Callantine, Julie Chronister, Sally Cohen, Ed Crowther, Kelly Deyoe, Suzanne Dilley, Brenda Downs, Janet Fath, Dick Henneman, Patty Jones, Merrick Kossack, Steve Krosner, Wendy Markert, Donald Mead, Laura Moody, S. Narayanan,

Tom Pawlowski, David Resnick, Doug Robinson, Ling Rothrock, Kenny Rubin, Serena Verfurth, Kim Vicente, Lauren Weisberg, Jim Williams, Chang Yoon, and Wan Yoon.

Finally, the author thanks his immediate family. He is grateful to his parents, Padma and Vasdev for providing him with the opportunity to pursue higher education, brother, Sanjay for the encouragement, and wife, Geetanjali for the love and support that cannot be described in words.

TABLE OF CONTENTS

LIST OF TABLES ix

LIST OF ILLUSTRATIONS xi

SUMMARY xiv

CHAPTER I: INTRODUCTION 1
- Background: Diagnostic Problem Solving 1
- Training for Diagnostic Problem Solving 2
- Research Objectives 3
- Proposed Methodology 3
- Scope 4
  - Domain Characteristics 4
  - Task Characteristics 5
- Implementation 5
- Validation 6
- Organization of Dissertation 6

CHAPTER II: INTELLIGENT TUTORING IN COMPLEX SYSTEMS 8
- Brief Review of Relevant Research 8
- General Architecture of ITS 10
- Characteristics of the Constituents of an ITS 12
  - Expert Module 12
  - Student Module 15
  - Instructional Module 16
  - Training Simulator 17
  - Interface 18
- Problems and Research Objective 19
  - Why the Slow Progress 19
  - Research Agenda 21

CHAPTER III: SIMULATION AND KNOWLEDGE ORGANIZATION METHODOLOGIES FOR INTELLIGENT TUTORS IN COMPLEX DYNAMIC DOMAINS 22
- Review of Research Goals 22
  - Simulation of Complex Dynamic Systems 22
  - Knowledge Organization in Intelligent Tutors 23
- An Architecture 24
- Simulation Via Qualitative Approximation 26
- Knowledge Organization 28
  - System Knowledge 30
    - Schematics 30
    - Functional Subsystems 31

Fluid paths 31
Components 32
Structural Knowledge 32
Functional Knowledge 33
Behavioral Knowledge 33
Troubleshooting Knowledge 34
Task Knowledge Organization 37
Knowledge to Evaluate Misconceptions 38
Instructional Strategies 39
Interactive Interfaces and Student-Tutor Interaction 41
Summary 43

CHAPTER IV: AN ITS IMPLEMENTATION: TURBINIA-VYASA 45
The Domain 46
Marine Power Plant 46
Steam Generation 46
Steam Expansion 46
Steam Condensation 49
Feed 49
Automatic Boiler Control System 49
Automatic Combustion Control System 52
Feed Water Control System 52
Makeup and Excess Feed Control System 53
Troubleshooting Task 53
Student Operator 55
Turbinia: The Simulator 55
Vyasa: The Computer-Based Tutor 59
System Knowledge 60
Fluid paths 60
Functional Subsystems 62
Schematics 64
Components 64
Failure Knowledge 65
Modes of Failure 65
Specific Failures 65
Knowledge of Student Actions 67
Knowledge to Update the Student Model 67
Knowledge to Evaluate Misconceptions 68
Instructional Knowledge 69
Content 69
Form 70
Time and Duration 73
Instructions with Intervention 73
Instructions without Intervention 74
Knowledge Representation and Control Structure 75
Knowledge Representation 75
Fluid paths 75
Functional Subsystems 79
Schematics 84
Graphical Objects 87
Components 87

Connectors 87
Icons 90
Active Regions 92
Components 94
Gauges 108
Icons 108
Failure Modes 110
Specific Failures 110
Control Structure 116
Blackboard 117
Tutor Behavior 124
Student Behavior 124
Knowledge Sources 124
Summary 127

CHAPTER V: STUDENT-TUTOR INTERFACE OF TURBINIA-VYASA 128
The Interface 128
Interaction with Turbinia-Vyasa 129
Schematic Menu 131
Requests Menu 131
Clock 131
Tutor Dialog 133
Symptom Dialog 133
Interaction with Turbinia 135
Interaction with Vyasa 148
Passive Mode 148
Active Mode 163
Hypothesis Menu 166

CHAPTER VI: EXPERIMENTAL EVALUATION OF TURBINIA-VYASA 176
Informal Evaluation 176
Checking for consistency and correctness 176
Pilot Study 177
Formal Experiment 178
Experimental Design 178
Equipment 179
Experiment 179
Experimental Materials 180
Experimental Procedure 181
Training Phase 181
Data Collection Phase 183
Training and Test Problems 183
Performance Measures: The Dependent Variables 184
Product Measures 184
Process Measures 185
Possible differences in performance 187

CHAPTER VII: EXPERIMENTAL RESULTS 190
Analysis of Data 190
Fixed Effects 192

Effect of Training Condition 192
Effect of Seen Status 202
Interaction Effect: Training Condition by Seen Status 209
Random Effects 214
Effect of Subject 214
Effect of Problem 215
Subjective Evaluation 216
Discussion of Results 218
Conclusions 219

CHAPTER VIII: CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH 225
Implications of the Research 226
Future Research 228

APPENDIX A: INSTRUCTION MANUALS 231
APPENDIX B: SUBJECT CONSENT FORM 333
APPENDIX C: SURVEY FORM 336
APPENDIX D: QUESTIONNAIRES 338
APPENDIX E: SAMPLE COMPUTATIONS OF TEST STATISTICS-I 350
APPENDIX F: SAMPLE COMPUTATIONS OF TEST STATISTICS-II 360
APPENDIX G: SOME COMMENTS FROM SUBJECTS 364
APPENDIX F: SOME RESPONSES TO QUESTIONNAIRE 3 368
BIBLIOGRAPHY 371
VITA 377

LIST OF TABLES

Table 4.1 Typical Abnormal System Behavior 66
Table 4.2 Description of Object Class FluidPath 76
Table 4.3 Description of Object Class FluidPathStructureWithinSchematic 76
Table 4.4 Description of *FuelOilPath* 78
Table 4.5 Description of Object Class FunctionalSubsystem 80
Table 4.6 Description of Object Class SubsystemStructureWithinSchematic 81
Table 4.7 Description of Object Class SubsystemStructureWithinFluidPath 81
Table 4.8 Description of *CombustionSubsystem* 83
Table 4.9 Description of Object Class Schematic 85
Table 4.10 Description of *BoilerSchematic* 86
Table 4.11 Description of Object Class GeneralGraphicObject 88
Table 4.12 Description of Object Class Connector 89
Table 4.13 Description of Object Class SchematicIconGraphicObject 91
Table 4.14 Description of Object Class IconGaugeGraphicObject 91
Table 4.15 Description of Object Class ActiveRegion 93
Table 4.16 Description of Object Class Primitive 96
Table 4.17 Description of Object Class PrimitiveGeneralStructure 96
Table 4.18 Description of Object Class SimplePrimitive 98
Table 4.19 Description of Object Class SimplePrimitiveSpecificStructure 98
Table 4.20 Description of Object Class CompositePrimitive 99
Table 4.21 Description of Object Class CompositePrimitiveSpecificStructure 99
Table 4.22 Description of Object Class StateVariable 100
Table 4.23 Description of Object Class NormalBehaviorProperties 102
Table 4.24a Description of Object Class Capacitor 102
Table 4.24b Description of Object Class CapacitorBehaviorProperties 102
Table 4.25a Description of Object Class Controller 103
Table 4.25b Description of Object Class ControllerBehaviorProperties 103
Table 4.26a Description of Object Class Reactor 104
Table 4.26b Description of Object Class ReactorBehaviorProperties 104

Table 4.27a Description of Object Class HeatExchanger 105
Table 4.27b Description of Object Class HeatExchangerBehaviorProperties 105
Table 4.28a Description of Object Class AbnormalBehavior 107
Table 4.28b Description of Object Class FailureEffect 107
Table 4.29 Description of Object Class Gauge 109
Table 4.30 Description of Object Class IconData 109
Table 4.31 Description of Object Class FailureMode 111
Table 4.32 Description of Object Class ExpectedAbnormalBehavior 111
Table 4.33 Description of Object Class SpecificFailureCase 112
Table 4.34 Description of *FailureTwo* 113
Table 4.35 Description of Object Class BlackBoard 119
Table 4.36 Description of Object Class PendingEvent 120
Table 4.37 Description of Object Class StudentAction 120
Table 4.38 Description of Object Class TutorBehavior 122
Table 4.39 Description of Object Class StudentBehavior 123
Table 6.1 Summary of Performance Measures 188
Table 7.1 Expected Mean Squares Expression 191
Table 7.2 Summary of Training Condition Effect 193
Table 7.3 Summary of Seen Status Effect 203
Table 7.4 ANOVA Tables 221

LIST OF ILLUSTRATIONS

Figure 2.1 General ITS Architecture 11
Figure 3.1 Major Components of Instructional System 25
Figure 3.2 Summary of Knowledge Components 29
Figure 3.3 Summary of Knowledge Organization in an ITS 44
Figure 4.1 Steam Generation Phase 47
Figure 4.2 Steam Expansion or Power Generation Phase 48
Figure 4.3 Power Generation Phase 50
Figure 4.4 Feed Phase 51
Figure 4.5 System Hierarchy 57
Figure 4.6 System Knowledge Decomposition 61
Figure 4.7 Interacting Subsystems of Marine Power Plant 63
Figure 4.8a Instructional Template to Suggest Hypothesis Revision 71
Figure 4.8b Instructional Template to Indicate Lack of Evidence 71
Figure 4.8c Instructional Template to Suggest Diagnostic Test to Strengthen or .....
Weaken a Hypothesis 72
Figure 4.8d Instructional Template to Express Inability to Provide Help 72
Figure 4.9 Primitive Class Hierarchy 95
Figure 4.10 Control Architecture 118
Figure 5.1 Configuration of Screens 130
Figure 5.2 Schematic Menu 132
Figure 5.3 Requests Menu 132
Figure 5.4 Tutor Dialog 134
Figure 5.5 Symptom Dialog 134
Figure 5.6a Steam Schematic 136
Figure 5.6b Boiler Schematic 137
Figure 5.6c Feed Water Schematic 138
Figure 5.6d Fuel Oil Schematic 139
Figure 5.6e Control Air Schematic 140
Figure 5.6f Saltwater Schematic 141

Figure 5.6g Lube Oil Schematic 142
Figure 5.7 Qualitative State Representation 144
Figure 5.8 Boiler Schematic Showing Gauges on the Steam Drum 146
Figure 5.9 Tutor's Instructions to Select the Suspected Component 147
Figure 5.10 Message of Congratulations on Correct Diagnosis 147
Figure 5.11 Error Dialog for Incorrect Diagnosis 147
Figure 5.12 Help-Levels Dialog 149
Figure 5.13 Help-Levels Dialog with "System" Buttons Enabled 151
Figure 5.14 Passive Tutor Dialog to Select Subsystem 151
Figure 5.15 Passive Tutor Dialog to Access Subsystem Related Knowledge 152
Figure 5.16 Passive Tutor Dialog to Display Selected Subsystem 152
Figure 5.17 Combustion Subsystem Highlighted in Boiler Schematic 153
Figure 5.18 Summary of Interactions with Passive Tutor Dialogs to Access System Knowledge 155
Figure 5.19 Passive Tutor Dialog to Display Failure Mode Knowledge 156
Figure 5.20 Summary of Interactions with Passive Tutor Dialogs to Access Failure Knowledge 158
Figure 5.21 Clipboard 159
Figure 5.22 Extended Portion of the Clipboard 159
Figure 5.23 Clipboard Indicating Change in Gauge Reading 160
Figure 5.24 Clipboard Indicating Blocked-Shut Mode of Failure 161
Figure 5.25 Examples of Instructions from Vyasa 164
Figure 5.26 Tutor Soliciting Hypotheses from the Student 165
Figure 5.27 Student Communicating the Suspected Mode of Failure 165
Figure 5.28 Example of Hypothesis Aiding with Intervention 167
Figure 5.29 Hypothesis Menu 167
Figure 5.30 Review Hypotheses Dialog 168
Figure 5.31 Delete Hypothesis Dialog 169
Figure 5.32 Advice Hypothesis Dialog 170
Figure 5.33 Example of Hypothesis Aiding without Intervention 171
Figure 5.34 Dialog to Request Solution 173
Figure 5.35 Solution for Students Trained on Simulator 173
Figure 5.36 Solution for Students Aided by the Tutor 174
Figure 5.37 Explanations for Abnormal System Behavior 175

Figure 7.1 Number of problems solved (training condition effect) 195
Figure 7.2 Average troubleshooting time per problem (training condition effect) 195
Figure 7.3 Number of informative actions (training condition effect) 197
Figure 7.4 Percentage of relevant informative actions (training condition effect) 197
Figure 7.5 Percentage of guesses (training condition effect) 199
Figure 7.6 Number of investigations in unaffected schematics/subsystems/fluid-paths (training condition effect) 199
Figure 7.7 Nature of diagnosis (seen status effect) 201
Figure 7.8 Number of problems solved (seen status effect) 204
Figure 7.9 Average troubleshooting time per problem (seen status effect) 204
Figure 7.10 Number of informative actions (seen status effect) 206
Figure 7.11 Percentage of relevant informative actions (seen status effect) 206
Figure 7.12 Percentage of guesses (seen status effect) 206
Figure 7.13 Number of investigations in unaffected schematics/subsystems/fluid-paths (seen status effect) 208
Figure 7.14 Nature of diagnosis (seen status effect) 208
Figure 7.15 Number of problems solved (interaction effect) 210
Figure 7.16 Average troubleshooting time per problem (interaction effect) 210
Figure 7.17 Number of informative actions (interaction effect) 211
Figure 7.18 Percentage of relevant informative actions (interaction effect) 211
Figure 7.19 Percentage of guesses (interaction effect) 211
Figure 7.20 Number of investigations in unaffected schematics (interaction effect) 212
Figure 7.21 Number of investigations in unaffected subsystems (interaction effect) 212
Figure 7.22 Number of investigations in unaffected fluid-paths (interaction effect) 212
Figure 7.23 Number of premature diagnosis (interaction effect) 213
Figure 7.24 Number of timely diagnosis (interaction effect) 213
Figure 7.25 Number of overdue diagnosis (interaction effect) 213

SUMMARY

One of the major problem solving skills required of human supervisory controllers concerns their ability to diagnose faults. Training for diagnostic problem solving is currently either provided on-the-job or on simulators. While on-the-job training has many problems, training on simulators alone is also inadequate. Many researchers feel that a simulator coupled with an intelligent computer-based tutor can provide better training.

The concept of using computers as training aids is not new. But, research in intelligent tutoring and training systems has traditionally focused on relatively simple tasks. Useful results and techniques established through previous research have, therefore, not been successfully applied to complex engineering systems. This research is aimed at overcoming some of the difficulties in extending the applications of computer-based training systems to complex engineering domains.

This research focuses on developing a methodology for decomposing, organizing and representing domain knowledge of complex dynamic systems for building functional computer-based intelligent tutors. In addition to developing a knowledge organization methodology, the research proposes, implements and evaluates a coherent architecture for constructing intelligent instructional systems. The results of the research demonstrate the viability of using computers to train operators to troubleshoot large engineering systems.

CHAPTER I

INTRODUCTION

Background: Diagnostic Problem Solving

In the operation of complex dynamic systems such as aircrafts and power plants, vast quantities of information must be processed 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. Thus, 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).

One of the major problem solving skills required of human supervisory controllers concerns their ability to diagnose faults. Fault diagnosis usually involves the identification of the primary cause of abnormal system behavior. It is the process of identifying malfunctioning components by observing abnormal states of the system, forming hypotheses about failure, and verifying each hypothesis by conducting diagnostic tests.

The diagnostic problem solving task in modern engineering systems is often complicated by the size, interactions and dynamics of the system. Size refers to the number of components in the system. An increase in size increases the probability of failure in the system and makes troubleshooting difficult by increasing the alternatives that can explain the abnormal system behavior. Interaction between parts of the system makes

diagnosing faults difficult due to the increased information processing load on the operator. In addition, the diagnostic process is complicated by the propagation of abnormal system behavior. Therefore, timely intervention by the operator and appropriate actions are necessary.

Successful fault diagnosis in engineering systems 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.

However, even when operators are familiar with the 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 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

Training for diagnostic problem solving is either provided on-the-job or on simulators (Johnson, 1988; Kearsley, 1987; Goldstein, 1986; Naval Training Command, 1973; Bureau of Naval Personnel, 1963). 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 training. 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 misconceptions from observed actions. A simulator coupled with an intelligent computer-based tutor may, however, improve training effectiveness. Such a combination of simulator and tutor constitutes an intelligent tutoring system (ITS).

The concept of using computers as training aids is not new. But, research in intelligent tutoring and training systems has traditionally focused on relatively simple tasks

concerned with imparting basic skills in mathematics, electricity, physics and computer programming (Wenger, 1987; Sleeman and Brown, 1982). While these domains have been useful for exploring research ideas, they are characterized by the absence of complex interactions between subsystems that are present in most engineering domains. Because of this inability to represent complexity, most ITS design principles have not been successfully extended from simpler, less constrained domains to complex engineering systems (Burns et al., 1991; Frasson et al., 1990; Psotka et al., 1988). A good part of this inability to represent complexity stems from the lack of a methodology to decompose and organize knowledge about large dynamic systems.

Research Objectives

Inspite of useful results and techniques established through previous ITS research, not much progress has been made in applying the findings in a functional engineering system. This research is aimed at overcoming some of the difficulties in extending the application of computer-based training systems to complex engineering domains.

This research focuses on developing a methodology for decomposing, organizing and representing domain knowledge of complex dynamic systems for building functional, computer-based intelligent tutors. In addition to developing a knowledge organization methodology, the research objectives include developing, implementing and evaluating a coherent architecture for constructing intelligent instructional systems. The viability of using computers to train operators to troubleshoot large engineering systems will be demonstrated as a result of this research.

Proposed Methodology

The framework for knowledge decomposition and organization proposed here for representing knowledge in intelligent tutors is based on an ITS architecture that separates domain knowledge from pedagogical knowledge. This framework suggests the decomposition of domain knowledge into system and task knowledge, and decomposition of the pedagogical knowledge into knowledge to plan and execute the pedagogical functions of the tutor that includes evaluation and rectification of misconceptions.

The proposed framework organizes system knowledge using a structure-function-behavior model of the system and its components. It organizes task knowledge in a manner that facilitates evaluation of student misconceptions. The organization of pedagogical knowledge proposed by this framework facilitates planning of pedagogical functions including inference of misconceptions and delivery of instructions using a blackboard-like control architecture.

Scope

The application of the knowledge organization methodology and the intelligent tutoring system architecture proposed in this research are limited to a class of complex domains and tasks. It is important to recognize the salient characteristics of the domain and the task that make them suitable for the knowledge decomposition and organization scheme of the proposed framework. These salient characteristics of the domain and task are described next to give some idea about the applicability and scope of this research.

Domain Characteristics

The methodology for decomposing, organizing and representing knowledge proposed in this dissertation is suitable for engineering domains (e.g., power plants, aircrafts, automobiles etc.) that are characterized by complexities such as size, high degree of interaction between subsystems and dynamics. In addition, these domains typically exhibit slow response to changes in the control settings. Furthermore, these systems seldom attain steady state. Even when steady state values are attained, it may take a long time to do so.

Also, most control operations in such complex dynamic systems are automated and human intervention becomes necessary only when the system exhibits abnormal behavior. Malfunctioning components responsible for abnormal system behavior cause the performance of the system to deteriorate progressively but seldom do they cause catastrophic situations. However, effects of failure can propagate to many interconnected portions of the system. Even components that are not physically connected to the failed component may be affected. Therefore, even a single fault can lead to a catastrophic situation due to cascading of failures.

Task Characteristics

Knowledge organization for intelligent tutoring proposed here is best suited for a diagnostic problem solving task that supervisory controllers of complex dynamic systems are often required to perform. The diagnostic task involves identification of a malfunctioning component in the system that is responsible for observed abnormal system behavior. In addition, the supervisory control task generally includes repair or replacement of the faulty component as well. In this research, the emphasis is on the identification of the failed component because it is a pre requisite for repair and replacement.

There are several characteristics of the troubleshooting task considered in this research. First, since the troubleshooting task involves identification of a failure that is usually not catastrophic, the task is not severely constrained by time. Second, since the time constraint is not severe, well defined mandatory procedures for identifying faults do not exist. Third, due to dynamics of the system, the cause-effect associations are time dependent and abnormal system states cannot be associated with specific failures without reference to a point in time. Furthermore, since steady state conditions often do not even exist, it becomes impossible to uniquely associate abnormal system behavior to specific faults. Therefore, operators have to depend upon their ability to successfully prune various alternatives before diagnosing the fault. Finally, limited availability of the gauges constrains the operators from observing all abnormal system behaviors. Thus, effective utilization of diagnostic information available is important for successful fault diagnosis.

Implementation

An important element of this research endeavor is the implementation of the proposed knowledge organization methodology to develop an experimental instructional system. The purpose of developing the instructional system was to validate the proposed ITS architecture and to demonstrate the viability of using computers for training in diagnostic problem solving. The domain of this experimental instructional system is a marine power plant. The instructional system comprises of a simulator and a tutor. The simulator is capable of simulating large number of failure situations and provides the training environment. The task of the tutor is to improve the student's troubleshooting skills in addition to providing practice and exposure to realistic situations. Knowledge

organization based on structure-function-behavior is used for both the simulator and the tutor.

Validation

Another important element of this research is the experiment that was conducted to evaluate the proposed ITS architecture. The experiment involved comparing the performance of subjects trained with and without the tutoring system. The experimental results demonstrate that training on simulator alone is inadequate. However, a simulator coupled with an intelligent computer-based tutor can enhance diagnostic problem solving performance. The data also indicate that the strategies developed by those aided by the tutor are different from those trained without the tutor. While the strategy used by those trained by the simulator relies heavily on pattern matching to recognize familiar cause-effect associations, the strategy developed by those trained with the tutor in more coherent and involves formulation of hypotheses and systematic elimination of less likely alternatives based on observed abnormal behavior. In addition, there is some evidence to suggest that training received from the tutor is better transferred to unfamiliar situations.

Organization of Dissertation

The purpose of this chapter was to provide a synopsis of the diagnostic problem solving task in complex dynamic domains, discuss the issues related to training human operators for this task and present the research objectives. This chapter also provided an overview of the research, its applicability and results. The remaining chapters of this dissertation describe the different phases of the research activity in more detail.

Chapter II provides a review of the general architecture of intelligent instructional systems for diagnostic problem solving in complex dynamic domains. The major constituents of the ITS architecture are described. The problem in extending the existing ideas to a wider spectrum of domains and the goals of current research are also discussed.

Chapter III proposes a methodology for organizing knowledge in intelligent tutors for diagnostic problem solving in complex dynamic domains. An instructional system architecture that uses this methodology is also described.

An implementation of the proposed instructional system architecture is described in Chapter IV. Implementation details of an intelligent training system to train operators to troubleshoot an oil-fired, steam propelled, marine power plant are described.

The student-tutor interface of the training system is described in Chapter V. Details of interaction at the interface are also discussed.

An experiment to study the effectiveness of the training system is described in Chapter VI. The purpose of this study was to validate the instructional system architecture.

The results of the experiment are presented in Chapter VII. Performance of subjects trained with and without the tutoring system are compared.

Finally, a summary of the results of this research are presented in Chapter VIII. Some recommendations for further research are also discussed.

CHAPTER II

INTELLIGENT TUTORING IN COMPLEX SYSTEMS

Brief Review of Relevant Research

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), Psotka et al. (1988), Frasson et al. (1990) and Burns et al. (1991) provide extensive surveys of existing intelligent tutoring systems of which only a few deal with engineering domains. Some examples of these systems that have useful applications in an operator training program are briefly discussed below.

SOPHIE (Brown et al., 1982) was perhaps one of the first applications of ITS in an engineering domain. It was built to teach troubleshooting in electrical circuits. Work on SOPHIE started in the 1970s and its development was carried out in three phases. In the first phase, an electronic troubleshooting expert was developed that could critique student's diagnostic problem solving performance on a simulator. In the second phase, the capability of the expert troubleshooter was enhanced to solve arbitrary faults introduced by the student. Finally, in the third phase, SOPHIE was equipped with a more powerful reasoner.

Since SOPHIE, a great deal of progress has been made in computer-based training for electronic troubleshooting. SHERLOCK* (Lajoie et al., 1990; Lesgold, 1990a; Lesgold, 1990b; Lesgold et al., in press), developed recently for a complex electronic troubleshooting job in the Air Force, has a far richer representation of the work environment than SOPHIE. It provides students with realistic means of practicing the task with context-specific support and feedback. Instead of imposing a particular troubleshooting strategy it

* (includes the entire SHERLOCK family)

provides help that is relevant to the current performance of the students when an impasse is reached. In addition, a model of the student's competence and performance gives SHERLOCK the capability to provide personalized instructions.

In domains such as power plants, ITS research on operator training has produced STEAMER (Hollan et al., 1984), The Recovery Boiler Tutor (Woolf et al., 1986), and AHAB (Fath, 1987; Fath et al., 1990). STEAMER does not teach any specific task. Instead, it uses innovative graphical techniques to display the behavior of portions of the power plant. STEAMER has neither means to evaluate the needs of the students nor can it provide help upon request to improve the student's understanding of the power plant. Thus, it is not really an *intelligent* tutoring system although it provides powerful graphical interfaces for interactive inspection of simulated faults in steam power plants.

The Recovery Boiler Tutor simulates the thermal and chemical processes in the boiler unit of a power plant. It has knowledge of boiler operating procedures for normal and emergency situations. The tutor uses this knowledge to teach the steps involved in controlling the situations arising from emergencies. The tutor also provides the students with the facility to interact with the simulator and stop the boiler processes to engage in activities needed to improve the understanding of the system.

AHAB addresses the complexities of a power plant such as size and interactions between subsystems better than any other training system in its domain. Using a Discrete Control Modeling methodology (Miller, 1985; Mitchell and Miller, 1986) to model the operator's task, AHAB teaches symptomatic and topographic search strategies (Rasmussen, 1986) to troubleshoot failures. It provides help to the student in the form of useful context-specific symptomatic and topographic diagnostic tests and evaluates the performance of the student based on deviations from the strategy prescribed by its task model.

Another complex domain that has been a target for computer-based training research is aviation. Intelligent Maintenance Training System (IMTS) provides an interactive environment for constructing domain-specific simulations and training scenarios (Towne et al., 1988). It has been used to develop a maintenance training program for SH-3 Helicopter Bladefold System. The Helicopter Bladefold System is a moderately complex, electrically controlled hydraulic system. The maintenance training program uses *Profile*, a generic troubleshooting expert of IMTS, to aid operators improve their diagnostic performance. The techniques for assessing and supporting the student's performance in

this training program, along with the ones used in SHERLOCK, are the most extensive among all existing systems.

MACH-III (Massey et al., 1988) is yet another training system for a complex domain. It will be used to train maintenance operators of HAWK radar systems to troubleshoot complex electronic devices at tactical installations. It is claimed that this system uses model-based qualitative reasoning techniques to generate explanations of normal and faulty radar operations for use in training.

In the field of spacecraft ground control, GT-VITA (Georgia Tech Visual Inspectable Tutor and Aid) being developed at the Center for Human-Machine Systems Research in Georgia Tech is likely to have significant impact on the training program for satellite ground control (Chu, 1991). GT-VITA teaches operations associated with ground control of satellite missions. The tutor uses GT-POCC (Georgia Tech Payload Operations Control Center) a real-time simulator of the domain as the training environment (Chu, 1991; Jones, 1991; Chu et al., 1991) and OFMspert's architecture (Rubin et al., 1987) to represent and interpret operator actions. During training, the participation of the tutor in helping the student understand the various operations and the task is progressively decreased as more actions indicate that the student is acquiring the knowledge necessary to perform the task.

The purpose of this chapter is to review the architecture of these tutoring systems and determine the reasons for a slow progress in the development of such applications. The rest of this chapter is divided into three sections. In the first section, the major constituents of the instructional system architecture are identified. In the second section, important characteristics of each constituent are discussed. Finally, in the third section, reasons for the slow development are outlined and an agenda for this research is presented.

General Architecture of ITS

In general, all intelligent tutoring systems have a similar architecture. Figure 2.1 illustrates the major constituents of such a system. The basic ITS architecture is comprised of an expert module, a student module and an instructional module. In addition, a simulator provides the training environment. The expert module contains the domain expertise which is also the knowledge to be taught to the student. The student module contains a model of the student's current level of competence. The instructional module is

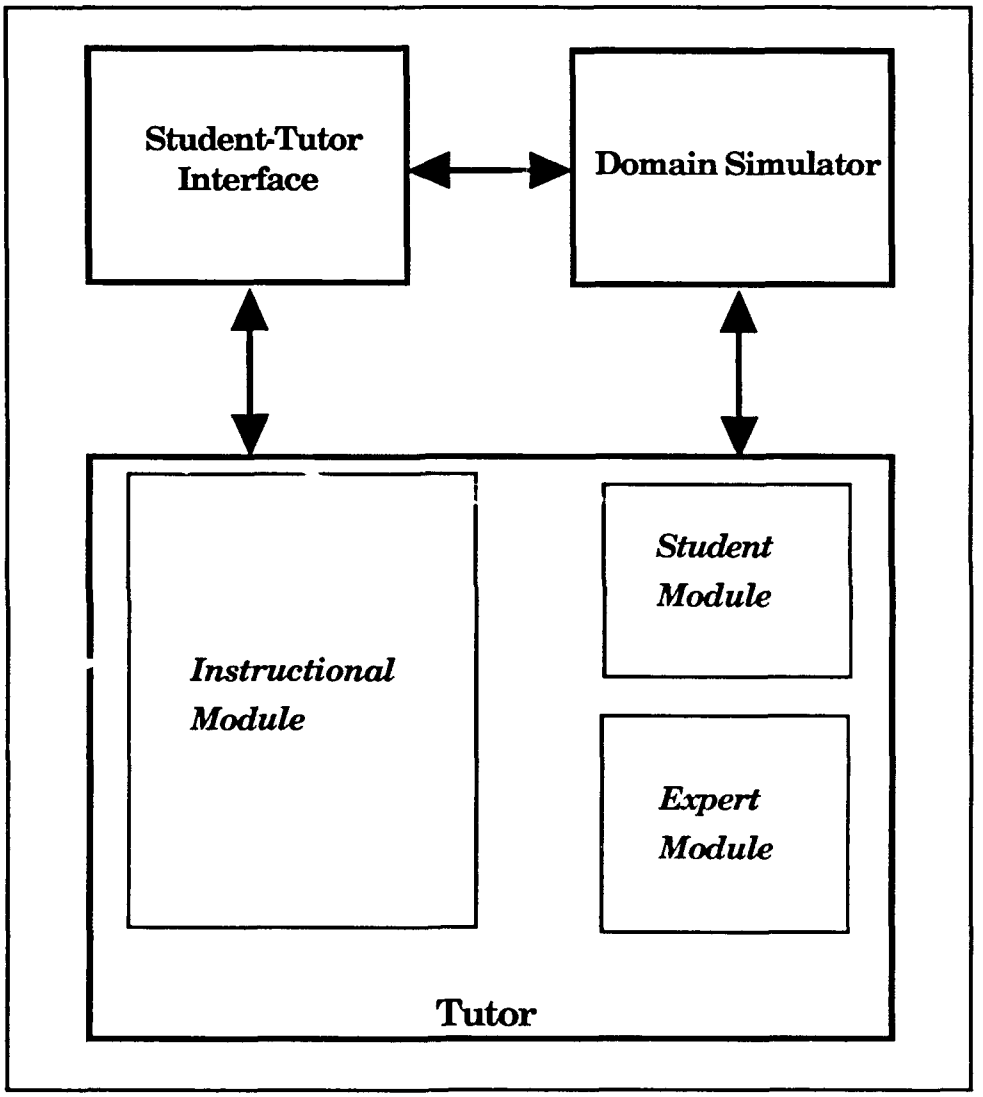


Figure 2.1 General ITS Architecture

designed to sequence instructions and tasks based on the information provided by the expert and student models. Also, the interface used to communicate knowledge to the student can be treated as a separate component of these systems. In the next section, each constituent of the intelligent tutoring system is discussed in further detail.

Characteristics of the Constituents of an ITS

This section describes the five major constituents of an ITS: expert module, student module, instructional module, the training environment and the interface. For the expert, student, and instructional modules, the discussion focuses on the knowledge represented in each. For the training environment and the interface, the discussion focuses on their importance in an ITS.

Expert Module

The expert module is the heart of the ITS and provides the domain intelligence and expertise. The domain knowledge embodied in this module refers to the subject matter as it relates to the task for which it will be used. It is not limited to procedures for executing the task and includes material that provides the foundation and justification for the application of procedures. Thus, domain knowledge is of two types: *declarative* and *procedural* (Anderson, 1988; Charniak and McDermott, 1985; Winston, 1984; Rich, 1983; and Nilsson, 1980). Whereas declarative knowledge refers to objects in the domain, facts about them and their interrelationship, procedural knowledge refers to set of compiled rules for executing the task.

Domain knowledge can be decomposed into system and task knowledge. System knowledge concerns the knowledge about the structure, function and behavior of the system and its components. Task knowledge includes procedures, facts, and some model of causation that facilitates reasoning in operator tasks such as troubleshooting. Both system and task knowledge can be declarative, procedural or both.

System knowledge is an essential component of the operator's domain knowledge. Even though this knowledge by itself is inadequate to perform the troubleshooting task, it enhances the troubleshooter's ability to solve problems. Therefore, representation of system knowledge in an ITS must be explicit and the knowledge must be communicable to

the students. It is not adequate to have a representation of the system knowledge that can be applied to simulate the training environment but cannot be presented in a comprehensible form to the student.

Likewise, the representation of task knowledge in an ITS must be explicit. In addition, this representation should be human-like (Clancey, 1987). This is important not only because the knowledge has to be communicated for comprehension by a human but also because this type of representation is ideal for evaluating misconceptions in a student. If misconceptions are to be inferred from student's actions, these actions should map to the representation of the task in the expert module.

Furthermore, the deployment of task knowledge for the purpose of executing the task must be separable and communicable to the students. When the knowledge representation is such that it can generate the correct operator responses for a task without being able to explain the computation of the responses, it is of little use for delivering instructions in an ITS. On the other hand, when the representation of knowledge underlying the expertise is articulate and inspectable by the student, it is more amenable to tutoring.

In most existing tutoring systems for complex domains, the domain knowledge is represented in various forms. For instance, the expert module of SOPHIE (Brown and Burton, 1975) does not have system knowledge represented in an explicit communicable form. The task knowledge in SOPHIE was initially not even supported by any model of causal reasoning and hence could not explain its diagnostic decisions that were computed by solving mathematical equations. In that form, the expert module was of little help to the instructional system. Later, however, the program was modified to contain a more articulate expert that could also explain its diagnostic behavior (Brown et al., 1986).

STEAMER, the interactive simulation-based tutor for propulsion engineering, has system knowledge expressed as mathematical models of processes in the steam power plant. These models can simulate the behavior of the system for various settings of control devices. Manipulation of these control devices is done through an interactive interface. The student is expected to learn the operation of the power plant by merely observing the changes in the system behavior for varied control settings. Although a powerful graphical interface is expected to enhance this learning process, the training system is incapable of explaining the computation of simulated system behavior to aid the student understand the process better.

The Recovery Boiler Tutor that teaches boiler processes does not support an explicit expert module. Instead, it has a knowledge base of emergencies and operating conditions expressed as scenarios. The system knowledge is based on mathematically accurate formulation of boiler processes that do not represent the mental model that the student is expected to develop.

Profile, the expert in IMTS, is based on a generic model of diagnostic task. It contains generalized troubleshooting rules rather than domain specific data. The domain specific data such as the list of essential context-specific diagnostic tests must be supplied by a human expert. Using the list of essential tests, *Profile* can generate useful symptom-cause data and failure hypotheses. Due to the vast amount of data generated and a large number of alternatives explored, the performance of *Profile* is often inefficient. Still, *Profile* is perhaps the only expert module that has been used to provide diagnostic support in the maintenance of many similar systems.

The expert modules of AHAB and SHERLOCK also do not have system knowledge represented in explicit communicable form. However, they do have rigorous models of the troubleshooting task for their respective domains. Unlike IMTS, the task models are extremely domain-specific. They use context-specific knowledge to quickly narrow the problem space to a manageable number of possible malfunctions that may be responsible for the abnormal system behavior.

Thus, in most instructional systems in which the system knowledge is represented, it is expressed in a mathematical form. While this type of representation is suitable for computing the system states, it is not necessarily suitable for helping students build appropriate mental models of the processes in a complex system. An alternative representation is required that goes beyond the non-causal framework of mathematical equations and helps the students to understand the structure, function and behavior of the system and its components.

Similarly, alternative methods of representing task knowledge are desirable. Task knowledge, when expressed as procedures, overly constrains the activities of the student and forces the training systems to impose a single troubleshooting strategy on the student. This results in ineffective training, particularly for non-procedural troubleshooting tasks that cannot be learned by rote alone. For such tasks, the tutor needs a representation of the

task that can help evaluate the relevance or importance of a diagnostic test at any instant. Using such a representation, the tutor can emphasize the effective utilization of diagnostic data instead of imposing a normative strategy when none exists.

Student Module

A student module in an ITS maintains a model of the student's current understanding of the domain. The student model is used to evaluate the student's need and help the instructional module in preparing appropriate individualized instructions. It stores actions taken by the student and has some means of representing the student's knowledge derived from recorded actions. Representation of data in such a student model must facilitate its comparison with the expert model of the task to enable evaluation of misconceptions in the student.

A variety of student modeling techniques have been used in tutoring systems. Most of these techniques fall under two categories depending upon the type of student model they use; overlay models and buggy models.

Overlay models consider the student knowledge as a subset of expert knowledge. Any missing or incorrect student knowledge is identified by overlay models but no explanation is provided for these inadequacies. GUIDON (Clancey, 1987), a tutor for diagnosing and prescribing therapy for infectious diseases, uses an overlay model.

Buggy models, on the other hand, have knowledge about errors and can provide explanations for incorrect behavior of the student. DEBUGGY (Burton, 1982), a program that teaches multi-digit subtraction, uses such a model to explain causes of student errors.

Student modeling in ITSs for complex dynamic domains has been very limited. SOPHIE has no student model. The student model in STEAMER merely records the engineering principles demonstrated to the student. In The Recovery Boiler Tutor, the student model only records the actions carried out by the student and identifies them as correct or incorrect

IMTS and SHERLOCK have better and more extensive models of the student. They model the student at multiple levels. They measure competence as well as performance of the student using overlay models consisting of predetermined goals. For each of these goals,

the level of capability of the student is recorded. In addition, IMTS maintains a fairly elaborate representation of the student's conceptual model of the system which is a subset of a normative model of an expert.

AHAB too has an overlay model and is supported by an additional buggy-type error model of the student. The overlay model is derived from the structure of the task model. The task knowledge of the student is evaluated by comparing the student's actions to the actions prescribed by the task model. The error model categorizes incorrect student actions into types of error that relate to inadequacies in the student's conceptual knowledge of the troubleshooting task. Instructions are generated depending upon the inadequacies in the student's conceptual knowledge as determined by the error model.

Instructional Module

The instructional module of an ITS is responsible for several activities. Its primary function is to control the curriculum, that is, select the material to be presented and its form of presentation. In addition, the instructional module evaluates student's misconceptions based on observed actions. To achieve these objectives, the instructional module makes use of pedagogical rules pertaining to presentation methods, query response and conditions for tutorial intervention. It also incorporates an algorithm that facilitates comparison of knowledge in the expert and student models and a framework for evaluating misconceptions based on this comparison.

In general, the instructional module adds to the instructional system's ability to teach a task that cannot be done efficiently by merely presenting problem situations on a simulator. Instructional modules try to enhance learning by providing help to the students in more than one way. Burton (1988) has categorized the various forms of help provided by instructional modules as: *help*, *assistance*, *empowering tools*, *reactive help*, *modeling*, *coaching* and *tutoring*. *Help* involves providing advice on request. *Assistance* is helping by doing part of the task. *Empowering tools* aid learning by reifying the problem solving process. *Reactive help* is responding to student's action in a manner that extends the understanding of the implications of actions. *Modeling* involves displaying how an expert executes the task. *Coaching* involves providing suggestions without adapting to the needs of the student. *Tutoring* is when the instructional module adapts to the individual needs of the students.

The form of help and the instructional strategies used by the instructional modules of most ITSs depend upon the pedagogical philosophy adopted by the instructional system. However, the strategies that have been effective in one domain may not necessarily be equally effective in another. For instance, the instructional strategies that have been successfully implemented in static domains are not applicable to dynamic systems (Munro et al., 1985). The timing of feedback, for example, is much more important when interacting with dynamic systems because the relevance of feedback is established with respect to states of the system that are continuously changing in a dynamic environment. When the feedback timing is incorrect, the instructions are less effective. In most instructional systems, it is not easy to incorporate changes in instructional strategies. Only RAPIDS (Towne et al., 1990), an authoring tool for planning instructions, facilitates building systems with instructional strategies amenable to quick changes.

Existing ITSs for training in complex domains have a loosely structured instructional module that serves its limited purpose in specific domains. These instructional systems make a feeble attempt at incorporating an instructional module that controls the curriculum and evaluates misconceptions. Most of these systems merely depend upon the knowledge in the expert module for teaching purposes without adapting to the individual needs of the student. Although this is an active area for research, evidence of implementation of research results or their effectiveness is hardly visible in the existing systems for training in complex dynamic domains.

Training Simulator

In complex real world systems, it is often impossible to generate situations merely for the purpose of training. Therefore, inspite of being costly and labor extensive, simulators provide the only practical alternative for any training program. Although simulators have, in the past, been used alone to provide instructions (Towne, 1986), these simulators by themselves lack the ability to evaluate student misconceptions. However, these simulators are an important part of any training program and, when coupled with computer-based tutors, enhance the effectiveness of an instructional system.

Simulator-based training, as opposed to text book learning, is more effective as it encourages students to explore and rapidly gather experience that cannot possibly be acquired otherwise (Sleeman and Brown, 1982). Simulation-based training is all the more useful for dynamic systems. It allows the student to learn by observing the effects of control

actions without the fear of safety hazards. It also allows the student to visualize the effects of failure propagation due to system dynamics.

However, not all training systems in dynamic domains actually simulate system dynamics. In some domains such as electronics, a troubleshooting task rarely involves investigation of transient behavior of the circuit. Steady state values are sufficient to diagnose the fault. For such domains, system dynamics may not be important. Therefore, training systems like SOPHIE and SHERLOCK can do without a dynamic simulator.

In domains where transient system behavior has a large amount of diagnostic information or where sometimes steady state may not even exist, real-time system dynamics cannot, however, be ignored. While this is true of most mechanical systems, most training systems including the Helicopter Bladefold maintenance training system built with IMTS, ignore real-time system dynamics. In the bladefold maintenance training system, the states of the simulated system attain a steady state value soon after the introduction of a malfunction. Although the state changes that occur over time in response to student actions are appropriately modeled and incorporated, the real-time dynamics is not faithfully represented.

In contrast to the systems discussed above, PEQUOD, the marine power plant simulator of AHAB, models the transient behavior of the malfunctioning power plant although it does not allow the controls of the system to be manipulated by the student. Hence, changes over time in response to student actions are not modeled.

Interface

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).

Among the training systems for complex dynamic systems, STEAMER has best utilized the power of graphics and icons. Using interactive graphical interface, iconic representations and flow animation, STEAMER demonstrates the functioning of a

simulated power plant and various properties of its operation unlike any other system of its kind.

IMTS and SHERLOCK both use direct manipulation interfaces where all student actions involve selecting an item and clicking on it using a mouse. In both these systems there is a good mapping between the actions taken at the interface and the actions relevant to the task in the domain. Objects that are manipulable by the operator in the domain are also manipulable at the interface.

AHAB too has a direct manipulation user interface. However, unlike IMTS and SHERLOCK, no control devices can be manipulated by the student. The student uses direct manipulation techniques to switch between screens and pick components to investigate its gauges.

This concludes a discussion of the five major constituents of an instructional system for training operators of complex dynamic systems. The next section describes the difficulties associated with extending and implementing ideas from existing systems to a wider range of complex domains. It also sets the agenda for the research.

Problems and Research Objective

Why the Slow Progress?

From the review of existing systems it is clear that despite the fact that all ITSs share a more or less common structure, implementing a computer-based training program for operators of complex real-world systems continues to remain a complicated exercise. Clearly, something more than the top-level modular architecture or the underlying computational machinery appears to be responsible for the slow pace of progress. Two reasons for the slow progress in implementation of computer-based training programs can be identified.

The first reason is the lack of appropriate simulation techniques. Intelligent training systems need a domain simulator of at least moderate levels of dynamic, structural and temporal fidelity (Su, 1985). For most real-world systems, due to their shear size and complexity of processes and interactions involved, developing such a training

environment is a tedious task. In the absence of an appropriate tool that supports rapid construction of these training environments, the task of building large simulators is made even more difficult. Moreover, constructing simulators for large systems using conventional modeling techniques is computationally expensive. For an ITS which must conserve its computational resources for the tutor rather than consume it on the simulator, computer power is of extreme importance.

Shortage of computational resources, however, is not likely to be a problem in the future due to the rapid pace of advances in computer technology. There is still a need to explore simulation techniques that can be used for rapid construction of training environments for instructional systems. If, in addition, the simulation technique can also be employed to develop simulators which require moderate amount of computational power and yet exhibit the fidelity suitable for the training program, it will surely enhance the development of training systems.

The second reason for the slow progress in developing training systems in engineering domains is the volume of knowledge and its organization. Knowledge in the tutoring systems needs multiple representation at various levels of detail and abstraction. For most real-world systems, this makes the volume of knowledge to be represented overwhelming. Furthermore, this knowledge is interrelated and tightly coupled.

While representing huge volume of knowledge is a problem, the complexity does not necessarily stem from a lack of knowledge representation methods. Recent advances in object-oriented programming techniques have provided efficient and economical ways to store knowledge. They have made knowledge representation for large complex systems in an ITS relatively less tedious and computationally inexpensive.

But, as knowledge is decomposed for efficient representation, there is concern about the level of detail that is necessary and adequate for achieving the pedagogical goals. Moreover, since the knowledge to be represented in the instructional system is highly interrelated, it cannot be stored as isolated modules. Knowledge must be integrated into proper contexts and mental models that can be easily recognized and comprehended. Inability to achieve this knowledge integration is precisely what is lacking in the ITS technology. A framework that can help integrate the large volume of knowledge associated with the safe operation of real-world systems is, therefore, needed.

The effectiveness of the existing training programs has also suffered due to a lack of technology supporting the development of interactive interfaces. However, advances in graphics, animation and direct manipulation interface designs in recent years have now made it possible to develop creative interface ideas on various computational platforms. These technologies need to be explored further to improve the methods of knowledge communication between a computer-based tutor and a student.

Research Agenda

The issues raised in the preceding section open up several avenues for further research in the field of intelligent tutoring systems. The research being reported here endeavors to address some of the issues and seeks to reduce the difficulties associated with the construction of intelligent tutoring systems.

The objective of this research is to reinforce the technology of developing instructional systems for supervisory controllers of complex systems with appropriate tools. Specifically, the research focuses on developing a methodology for decomposing, organizing and representing system and task knowledge of a large complex dynamic system.

The research has four goals. The first goal is to identify a suitable modeling technique for building simulators of moderate fidelity for training programs. The second goal is to propose a methodology for organizing knowledge of complex dynamic systems for use in intelligent instructional systems. The third goal is to implement the proposed methodology in a coherent ITS architecture to develop a training program for operators of complex dynamic systems. The final goal is to experimentally demonstrate the effectiveness of the instructional system developed through the proposed architecture.

Although this research is by no means aimed at solving all the issues associated with ITS research, it is directed at bridging the gap between theory and implementation. This research provides a pragmatic approach for extending the applications of ITS to real-world systems. It is expected that the results of this research will stimulate development of better training programs for supervisory controllers of realistic, complex dynamic systems.

CHAPTER III

SIMULATION AND KNOWLEDGE ORGANIZATION METHODOLOGIES FOR INTELLIGENT TUTORS IN COMPLEX DYNAMIC DOMAINS

Review of Research Goals

Successful applications of intelligent tutors for training in the domain of complex engineering systems are scarce. Two possible reasons for their limited applications were identified in Chapter II. One was the lack of a simulation methodology that prevented developers from rapidly constructing simulators of desired fidelity with minimum computational power. The other was the lack of a principled way of organizing the huge volume of interrelated knowledge into a coherent, feasible architecture for instructional systems. If more applications of intelligent tutoring systems are to be developed in engineering domains it would help to explore new simulation techniques and develop a methodology for organizing knowledge in them. Characteristics of the simulation technique and knowledge organization that will help the cause of improving the status of ITS technology are briefly discussed next.

Simulation of Complex Dynamic Systems

Rapid prototyping of a large scale dynamic simulator for training systems is extremely important. IMTS (Towne et al., 1988) is one of the few systems that provides simulation tools for constructing large scale simulators of complex systems quickly for training purposes. It maintains a library of graphical objects created with the help of an editor. The same editor is also used to specify the object behavior. Equipment specific simulations are built by interactively assembling the graphical objects. During simulation, the system states are computed using the behavioral knowledge of the objects and their graphical connections.

However, the simulation methodology adopted in IMTS does not capture the real-time dynamics of the simulated system. It uses only the steady state values to describe the state of the dynamic system. Steady state values are adequate for most operator tasks including diagnosing faults in many complex domains. However, transient state values are necessary in other real-world domains such as a power plant, where a steady state may not exist or where troubleshooting for faults begins well before the system attains a steady state. Thus, simulators for such domains must incorporate knowledge of the system's structure, function, and behavior in a manner that facilitates evolution of system states with time. Furthermore, this knowledge must be available at multiple levels of abstraction so that the evolution and propagation of system states across interacting subsystems can be described to the student.

A methodology that facilitates the simulation of large complex dynamic systems with reduced computational effort is described in this chapter. This methodology is currently not supported by an interactive user interface that can help build simulations rapidly. However, in conjunction with an IMTS-like interactive interface, it can serve as a useful tool for building dynamic simulators.

Knowledge Organization in Intelligent Tutors

Besides the simulator and the knowledge of system dynamics, what really makes the training system *intelligent* is the organization of knowledge. Absence of systematic organization of knowledge adversely affects the effectiveness of a computer-based tutor. In an effective ITS, the domain knowledge must be separated from teaching knowledge and made explicit. Having explicit domain knowledge makes it easily accessible and communicable to the student. Separation of knowledge also makes it possible for the tutor to present the domain knowledge in more than one way. This allows an ITS to function with more flexibility and present instructional material in different styles.

For a large, complex, dynamic, real-world system, the problems related to the knowledge and its representation in an instructional system are two-fold. First, the volume of knowledge is enormous. Second, its organization is critical for the success of the instructional system. Both the type and organization of knowledge in an instructional system also varies 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) have the largest amount of

structured knowledge as compared to tutor-dominated traditional computer-aided instructional programs or student-dominated discovery learning environments (Sleeman and Brown, 1982; Wenger, 1987; Psotka et al., 1988).

Thus, a consistent knowledge organization methodology cannot only increase the pace of progress by cutting down on the development time but also ensure effectiveness of the instructional system.

This chapter describes a knowledge organization methodology that can be used to develop effective intelligent tutoring systems to train operators of engineering systems. Details of the methodology are presented with reference to an ITS architecture proposed for developing intelligent instructional systems for complex dynamic systems.

An Architecture

This section describes an ITS architecture. Figure 3.1 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 three major requirements: (1) a domain simulator; (2) organization of knowledge that supports the functions of the three major elements of the tutoring system; and (3) an interactive student-tutor interface.

A complete description of the architecture is provided in the next three sections. The first section is a discussion of a simulation methodology suited for developing simulator-based training systems. The second section provides a methodology for decomposing and systematically organizing knowledge concerning complex dynamic systems. The framework proposed by this methodology can be used to decompose knowledge into smaller and easily comprehensible units for use in instructional systems. The third section is a discussion of interactive interfaces and student-tutor interaction.

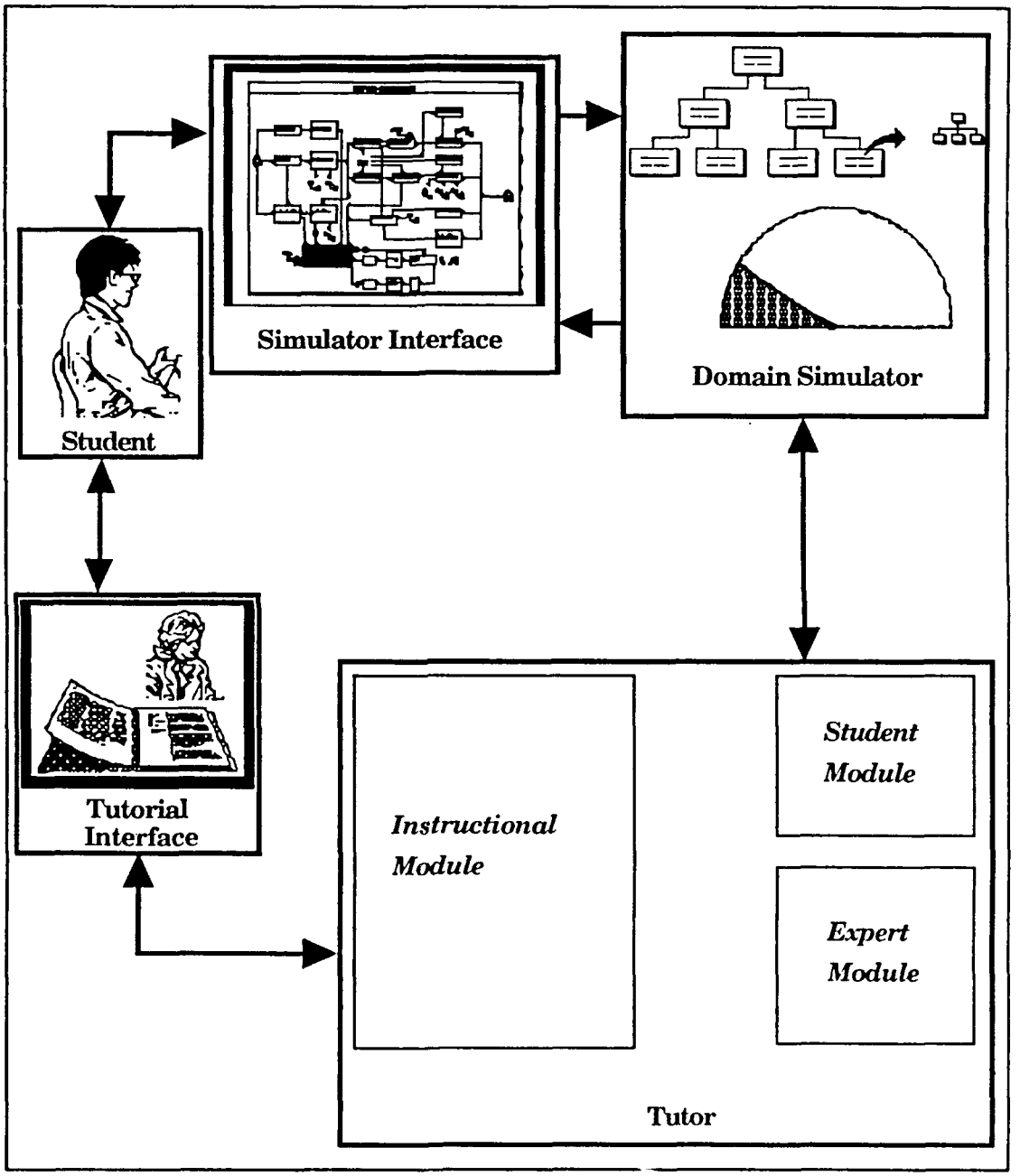


Figure 3.1 Major Components of Instructional System

Simulation Via Qualitative Approximation

In this section, a qualitative approximation methodology for the design of moderate fidelity simulators is described. Basic principles of this methodology were developed by Govindaraj (1987).

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 involved in troubleshooting 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.

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, feed water, steam, lube oil, and control air. Some components might belong to more than one subsystem. For example, the condenser is part of the feed water 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 qualitative approximation. 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.

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 (Takahashi et al., 1972). 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. The function, characterized by appropriate differential and algebraic equations, is the same for a primitive regardless of the component which it represents. Behavior of the primitives are based on approximate functional equations of system dynamics and parameter values. 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 built through this technique to maintain cognitive compatibility with trainees. This is because the system states computed using qualitative approximation are similar to state descriptions used by the human (Govindaraj, 1988). 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. Therefore, in training simulators, there is no need for 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.

Knowledge Organization

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 turbogenerator. In addition, the operator must know the nominal values of the state variables and parameters. Problem solving and compensation for failures require processing of information from various subsystems using efficient troubleshooting strategies. Therefore, an intelligent tutoring system must be capable of organizing and presenting knowledge about the system and the troubleshooting task at several levels of granularity or detail.

Successful implementation of an intelligent tutor for diagnostic problem solving in complex dynamic domains depends upon the availability of (1) a large amount of system knowledge organized to facilitate evolution of system states with time, (2) troubleshooting task knowledge, (3) knowledge to infer a student's possible misconceptions from observed actions, and (4) pedagogical knowledge to realize the tutoring objectives.

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

The remaining portions of this section describe the framework for decomposing and organizing knowledge for a computer-based diagnostic problem solving tutor in complex dynamic domains. Figure 3.2 summarizes the components of knowledge identified by this framework. Each knowledge constituent is described in detail next.

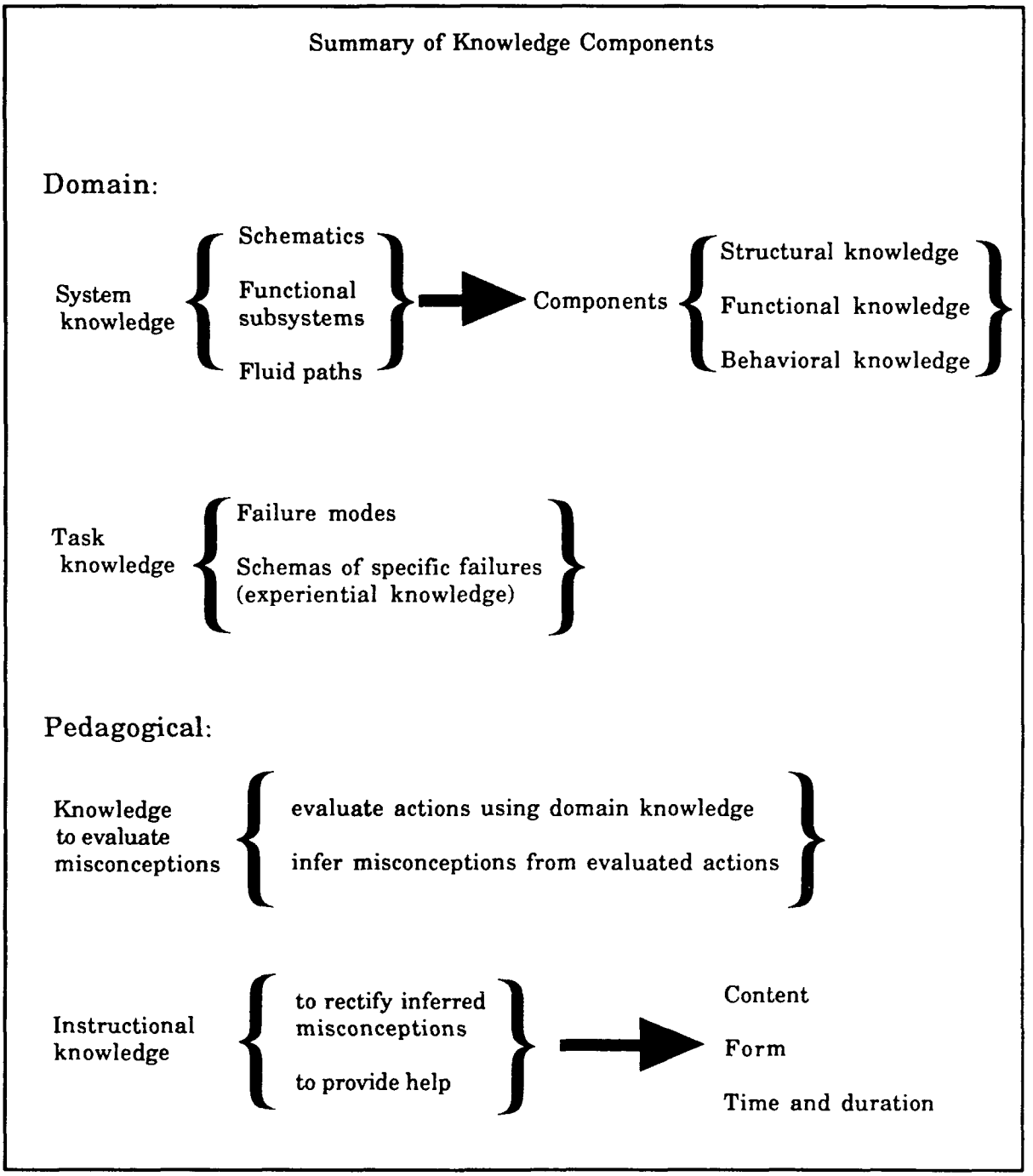


Figure 3.2 Summary of Knowledge Components

System Knowledge

Successful fault diagnosis in complex dynamic 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 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 and ***fluid paths*** help in visualizing the system in terms of different fluids that flow through the system. Thus, the three representations of the system knowledge are complementary rather than mutually exclusive. A detailed description of system knowledge decomposition into schematics, functional subsystems and fluid paths 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.

Components

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 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.

System 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. The nature of this diagnostic problem solving knowledge is described here.

A mechanical component in a physical system such as a steam power plant can fail in more than one way. There are four common modes of failure in components: (a) blocked-shut, (b) stuck-open, (c) leak-in, and (d) leak-out (Fath et al., 1989). Faults in components fit one or more of these four mode types. Not all components, however, fail in all four 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.

Each failure mode exhibits a typical system behavior (Fath, 1987; Fath et al., 1990). 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., 1990). 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. 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.

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, they 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 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 switches between the two strategies many times to complete the task.

The discussion here has provided an overview of an expert's troubleshooting knowledge and the 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 suitable for expert performance is not necessarily suitable for instruction or for evaluating student's misconceptions (Clancey, 1987). An alternative organization of the expert's task knowledge that may help evaluate a student's misconceptions is required.

Task Knowledge Organization

An important feature of an intelligent computer-based 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. A structure of such a model that can provide the tutor with a capability to evaluate misconceptions is 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., 1990). 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 that describes the valid actions of a student for each failure condition can thus be used to evaluate students' misconceptions. The knowledge required to evaluate misconceptions using the normative model is described next.

Knowledge to Evaluate 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.

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. The proposed method of organizing the system and task knowledge is also helpful in organizing categories of misconceptions. Misconceptions can be categorized as those related to a lack of (1) structural knowledge of the system, (2) functional knowledge of system and components, and (3) knowledge of system behavior resulting from failures. Identification of each type of misconception is described next in further detail.

The lack of system structural knowledge makes the student investigate portions of the system unrelated to the failure. For example, if the abnormal system behavior in a power plant are initially observed in the boiler, the student is expected to investigate gauges

mounted on the boiler or on the components in the vicinity of the boiler. If, however, the student fails to call up the schematic that contains the boiler or struggles to locate it in the schematic, it can be attributed to inadequate knowledge of system structure.

If, on the other hand, the student calls up the relevant schematic for investigations but checks components and gauges in the fluid paths unaffected by failure, it indicates a lack of understanding of different system functions and their inter-relationships. For instance, if the observed abnormality concerns low water level in the boiler, persistent investigations along flue gas path is unlikely to yield any useful diagnostic information. Such an action is clearly an indication of the student's inability to integrate functional information about the boiler and the interactions between the fluid paths through the boiler.

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. For example, if the pressure gauge on the condensate pump displays a normal reading, it is unreasonable to suspect a blocked-shut mode of failure in the condensate pump. Continued suspicion of a component in spite of evidence available to the contrary suggests inability on the part of the student to link failures to abnormal system behavior resulting from 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 pedagogical strategies outlined in the instructional module of the ITS.

Instructional Strategies

The instructional module of an ITS contains pedagogical 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., Clancey, 1987; Burton and Brown, 1982). 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 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.

Similarly, the form of presentation may be governed by another set of instructional rules. These rules may specify preference for either graphical or textual mode of presentation under various situations. These preferences may be based on context or norms formulated through experience by human instructors.

Finally, time of presentation of the instructional material is equally critical. There are usually two conditions under which the tutor is expected to deliver instructions. First, when explicit queries are raised by the student. Second, when a student's misconception is identified by the tutor. In the first case, the response should be immediate. In the second case, the response can be with or without intervention. Instructions without intervention are usually 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.

This concludes the discussion on knowledge organization in intelligent tutors for diagnostic problem solving in complex dynamic domains. However, knowledge organization that captures system structure, function, and behavior, troubleshooting task knowledge, knowledge to evaluate and rectify misconceptions, and instructional strategies are insufficient for the success of a tutoring system. Properly designed interactive interfaces also play a major role in imparting knowledge about the system and its operation during normal and abnormal situations. In the next section, the importance of such interfaces and how a student interacts with an instructional system are described.

Interactive Interfaces and Student-Tutor Interaction

A good interface is needed to make the knowledge of the tutor transparent to the student and help the student understand the complex structure, function, and behavior of the controlled system. In addition, a well designed interface addresses the external-internal task mapping problem (Moran, 1983) and establishes a semantic link between the actions relevant to the task in the domain and the actions to be taken at the interface (Miller, 1988).

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

Graphics and icons in a schematic interface can enhance the performance of instructional systems (Hollan 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 turbine. The trapezoidal shape also reminds the viewer that the steam expands in the turbine as it moves from a smaller cross section inlet to a larger cross section outlet. Similarly, concepts such as blocked-shut valve and functional subsystems of a power plant can also be represented by meaningful icons.

Schematics provide an interface between the student 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 helping 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.

Summary

An architecture for building intelligent training systems for supervisory controllers in complex dynamic domains was described in this chapter. First, a simulation methodology was discussed that can be used to develop dynamic simulators of large engineering systems at a low computational cost. Next, elements of knowledge were identified that constitute the expert, student and instructional modules of intelligent training systems in complex dynamic domains. Organization of this knowledge into various modules of an ITS architecture was also described. Figure 3.3 summarizes this organization. Finally, issues related to the design of interactive interfaces for student-tutor interaction were discussed.

The next chapter describes **Turbinia-Vyasa**, an implementation of the ITS architecture discussed in this chapter. This implementation uses qualitative approximation to simulate its domain; an oil-fired steam-propelled marine power plant. The major components of this ITS were built using the framework for knowledge organization discussed earlier in this chapter.

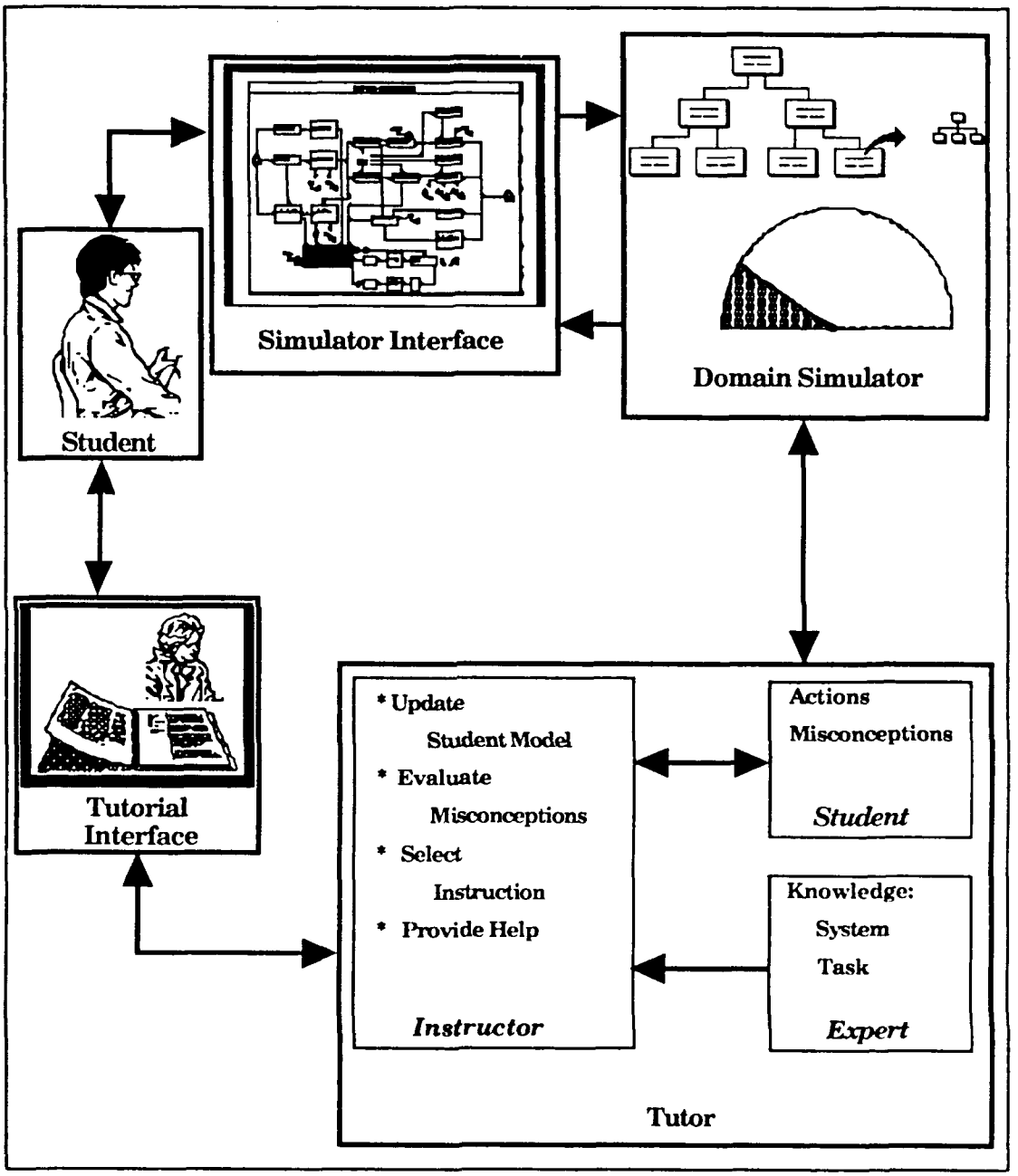


Figure 3.3 Summary of Knowledge Organization in an ITS

CHAPTER IV

AN ITS IMPLEMENTATION: TURBINIA-VYASA

The primary goal of this research was the development of a methodology for organizing knowledge in a diagnostic problem solving tutor for complex, dynamic domains. To determine the feasibility of the proposed knowledge organization methodology in an actual training environment, an ITS architecture was developed and implemented. The implementation consists of a domain simulator, **Turbinia*** ; and a computer-based tutor, **Vyasa**** . Together, **Turbinia** and **Vyasa** constitute an instructional system that trains operators to troubleshoot oil-fired steam-driven marine power plants. Implementation features of the instructional system are described in this chapter.

This chapter is organized into four major sections: (1) a brief description of the domain; (2) a description of **Turbinia,** the domain simulator; (3) a description of the computer-based tutor, **Vyasa**; and (4) implementation details of both **Turbinia** and **Vyasa**. The description in sections 2 and 3 covers the organization of knowledge in the instructional system. Implementation details of knowledge representation are provided in section 4.

---

* Turbines were 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 (Burstal, 1965, p.340).

** Ancient Indian sage, scholar and teacher

The Domain

Marine Power Plant

The domain of **Turbinia-Vyasa** is an oil-fired steam-propelled marine power plant. This power plant is installed on Navy vessels to produce the power required to drive the ship. In such a power plant, the chemical energy of fossil fuel is converted to thermal energy which is carried by steam to *turbines* for transformation into mechanical work. This section describes the functioning of an oil-fired steam-propelled marine power plant.

The process of producing mechanical work in a steam-propelled marine power plant can be decomposed into four stages (Gritzen, 1980). Each stage is associated with one of the four phases in the steam cycle: generation, expansion, condensation and feed. Together, the four phases of the steam cycle form a closed loop.

Steam Generation

The steam generation phase of the steam cycle takes place in the *boiler* (Figure 4.1). The boiler is comprised of *tubes* and a *steam drum*. The boiler tubes contain water that is heated by flue gases resulting from the fuel burned in the *furnace*. Heat transfer is by conduction through the tube walls. Heating of water in the tubes produces steam. This steam accumulates over the water surface in the steam drum and is called *saturated steam*. The saturated steam does not contain enough thermal energy to operate the turbines at their best efficiency. Thermal energy of steam is increased by passing it through tubes in the section of the boiler closest to the furnace. This section of the boiler is commonly known as the *superheater*. The steam from the superheater is at high pressure and is called *superheated steam*. The superheated steam then enters the expansion phase of the steam cycle.

Steam Expansion (or Power Generation)

Steam expansion takes place in two steps. First, the superheated steam from the boiler expands in a *high pressure turbine* to convert thermal energy to mechanical work. Then, since the steam still contains a considerable amount of thermal energy, it is expanded further in a *low pressure turbine* connected to the exhaust of the high pressure turbine. Figure 4.2 shows the arrangement of low and high pressure turbines.

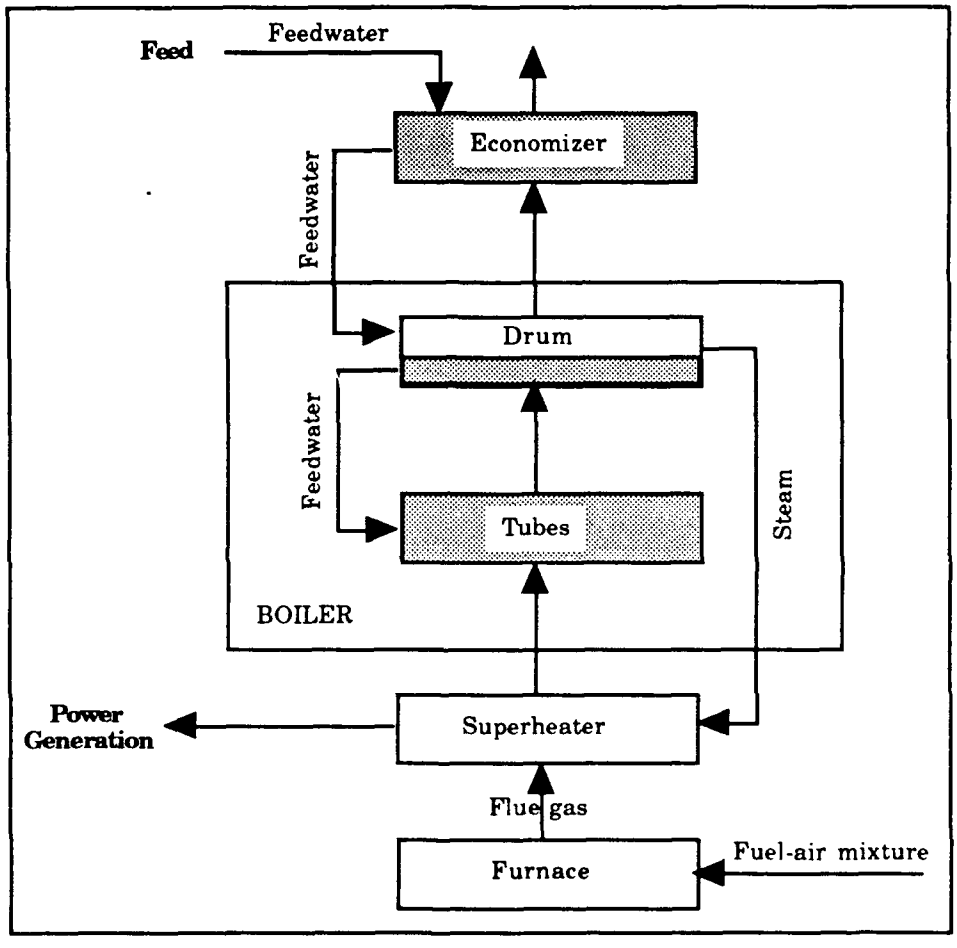


Figure 4.1 Steam Generation Phase

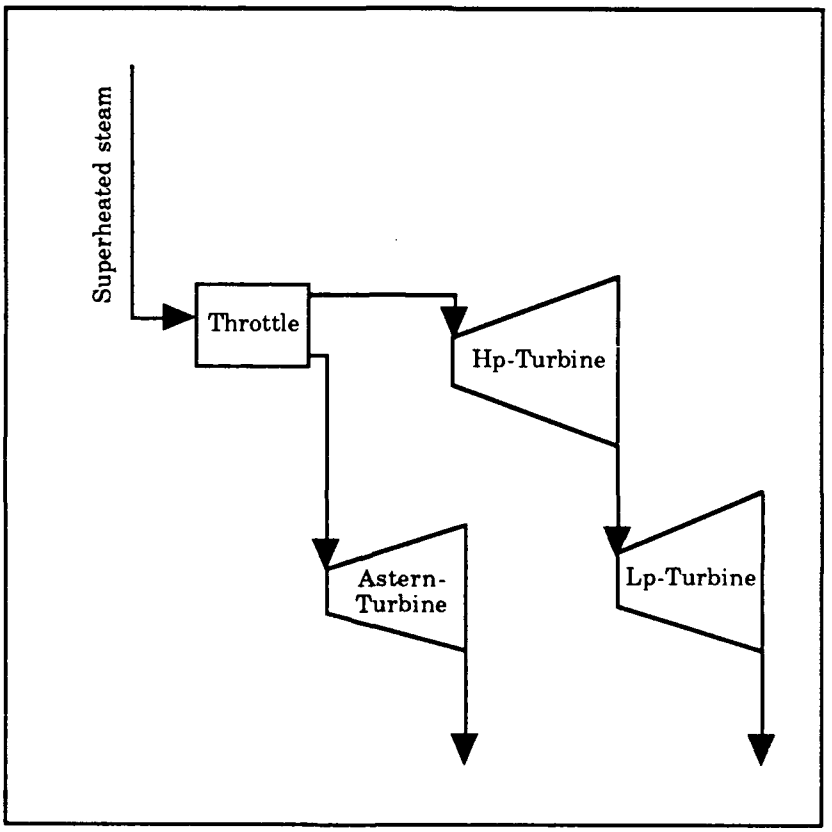


Figure 4.2 Steam Expansion or Power Generation Phase

Steam Condensation

After expansion, the third phase of the steam cycle is steam condensation which takes place in the main *condenser* (Figure 4.3). The condenser is a heat-exchanger made up of tubes that carry cold sea water. When steam passes over these tubes it loses latent heat to cold water. After sufficient energy is removed from the steam, it changes phase and turns back into water, called *condensate*. The condensate is pumped into the *deaerating feed tank* before it is re-used in the boiler.

Feed

Feed, the last phase of the steam cycle, begins at the deaerating feed tank. The deaerating feed tank is a storage tank for feed water. A feed pump pumps the feed water from this tank to the boiler. Enroute to the boiler, the feed water is preheated in an economizer to salvage the remaining thermal energy from flue gases leaving the boiler. Preheating of feed water also improves the thermal efficiency of the boiler. Figure 4.4 shows the configuration of components in the feed phase of the steam cycle.

Several of the components in the steam cycle of a power plant are control devices. The control settings of these devices can be adjusted to safely meet the varying demand for power. Often the operating conditions of these devices are altered by automated control systems. The boiler control system is the most important among all control systems in a marine power plant. The boiler control system of most modern Navy vessels is sophisticated and needs minimum human intervention. Some components of the automatic boiler control system (ABC) are described next.

Automatic Boiler Control System

Navy vessels typically have the following three ABC systems: *automatic combustion control* (ACC), *feed water control* (FWC), and *makeup and excess feed control systems*. These control systems perform the functions of measuring, comparing, computing and correcting. In each control system, a state value of interest is measured first. Then, the measured value is compared to a desired value. Next, if there is deviation between the measured and the desired value of the state, a new operating condition is computed. Finally, if necessary, the operating conditions are corrected or reset to reduce the observed deviation in the state value.

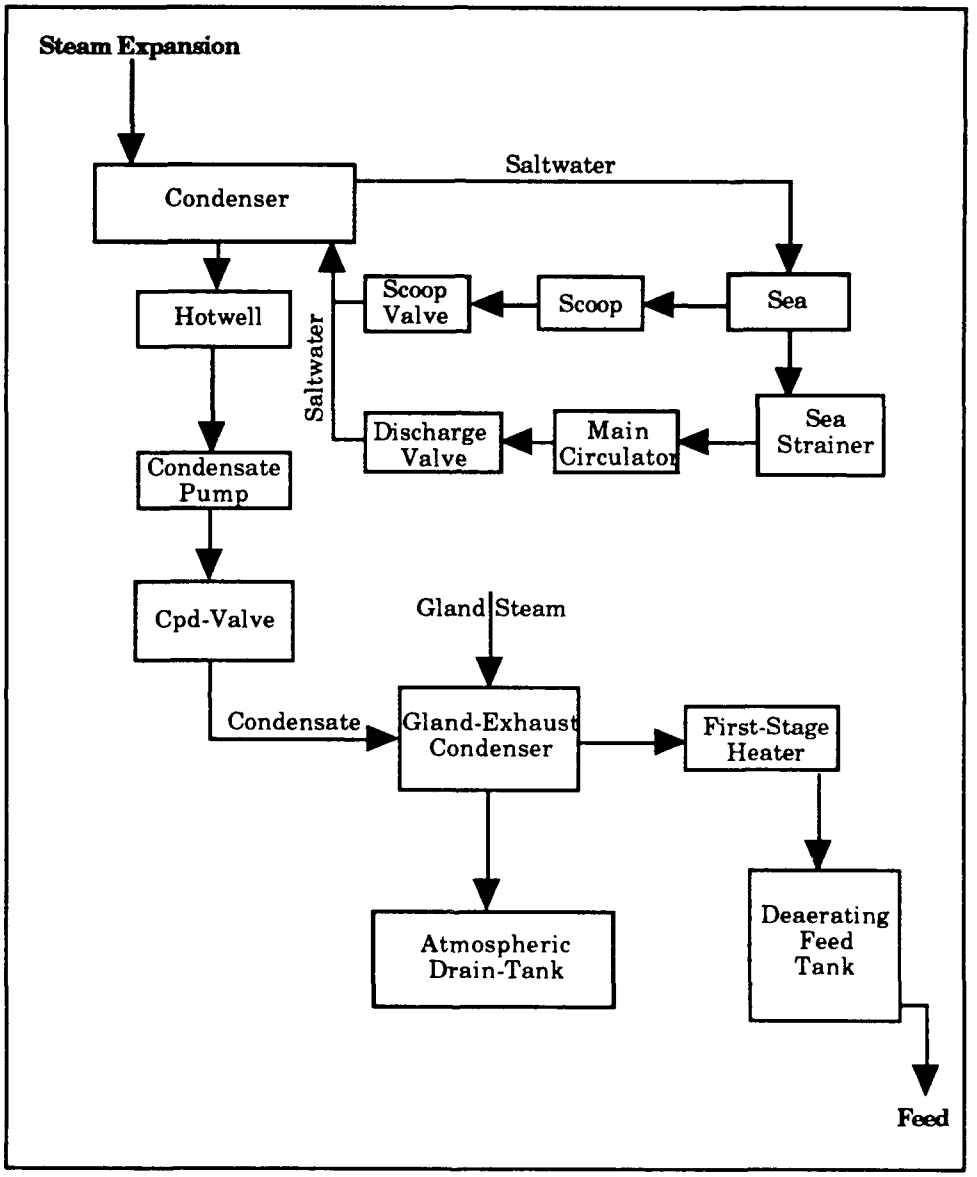


Figure 4.3 Steam Condensation Phase

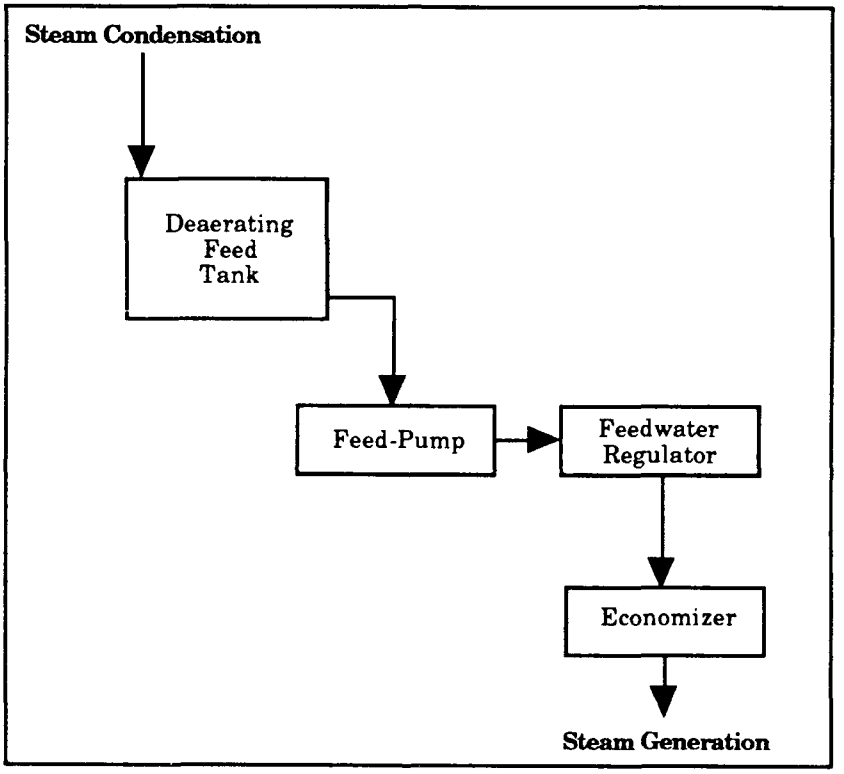


Figure 4.4 Feed Phase

Automatic Combustion Control System

The function of the automatic combustion control system is to maintain the boiler drum pressure at a constant value during steady and changing load conditions. The ACC system accomplishes this task by

(1) constantly measuring the steam drum pressure and combustion air flow;
(2) comparing the steam drum pressure to the specified designed value;
(3) computing the amount of change, if any, in the furnace combustion; and
(4) correcting furnace combustion as needed.

When the steam demand on the boiler is increased, the steam drum pressure decreases because the rate of steam withdrawal from the drum becomes greater than the rate of steam production in the boiler. This pressure drop is sensed by the ACC system and an increase in furnace combustion is computed to meet the increase in the demand for steam. Computing the increase in furnace combustion involves computing the increase in the supply of combustion air and a proportionate increase in the supply of fuel oil to assure complete combustion. The ACC system controls the combustion air flow by regulating the supply of steam to the forced draft fan turbine and controls the fuel oil flow by positioning the main fuel oil control valve. Air flow measurement provides the ACC system with the feedback necessary to perform this function.

Feed Water Control System

The function of the feed water control system is to maintain a constant water level in the steam drum. The FWC system automatically does this by

(1) measuring the steam drum water level and the feed water flow rate to the boiler;
(2) comparing the measured water level in the drum to a designed value;
(3) computing the required change, if any, to the rate of feed water flow; and
(4) correcting the feed water flow rate as needed.

When the load is steady, the feed water flow rate into the boiler equals the rate of steam consumption and the water level in the steam drum is normal. But, when the load changes, so does the demand for steam. Any change in this demand is detected and the feed water flow rate is adjusted to equal the steam flow rate out of the boiler. The actual control of feed water flow is accomplished by adjusting the air-operated diaphragm of the feed water regulator between the feed pump and the boiler.

Makeup and Excess Feed Control System

Operation of a steam-driven power plant often requires the addition or removal of water from the steam cycle. The makeup and excess feed control system is responsible for doing this and for maintaining a specified level of feed water in the deaerating feed tank.

Whenever the level in the deaerating feed tank deviates from the specified value, water is either withdrawn from or added to the deaerating feed tank. In both cases the process is facilitated by two standby tanks: the atmospheric drain tank and the distillate tank. When the feed water level in the deaerating feed tank falls below normal, the makeup feed regulator is adjusted by the control system to increase flow from the standby tanks. Increased flow into the deaerating feed tank compensates for the loss in the feed water level. Similarly, a deaerating dump regulator is activated by the control system to withdraw excess feed water from the deaerating feed tank when the level in the tank rises above the normal value.

In addition to the automatic boiler control system, a power plant has several other controls which are not discussed here because they are not relevant to the scope of problems covered by **Turbinia**. A brief description of the failures in the power plant and the nature of the troubleshooting task are provided next.

Troubleshooting Task

Many failures in marine power plants, particularly those simulated by **Turbinia**, are not catastrophic in nature. Most failures involve a single malfunctioning component resulting in progressive deterioration of performance. However, if these failures are left unattended for a long time, catastrophic situations may be created due to cascading of failures. Thus, troubleshooting for a failure involves identifying a single malfunctioning component, and investigations to identify the cause of the failure must begin as soon as symptoms of abnormal behavior are noticed.

Since a single failure by itself is usually not catastrophic, the troubleshooting task of identifying the failed component is not severely constrained by time. Although prompt identification of failed component is cost effective, well defined procedures do not exist for identifying faults quickly. Therefore, unlike catastrophic situations where operators are

taught to follow mandatory procedures, troubleshooting involves forming a set of several concurrent hypotheses concerning failed components and repeatedly refining the hypotheses based on the outcome of diagnostic tests. The refinement of hypotheses continues until the suspicion set is reduced to a single component.

Diagnostic problem solving task in a dynamic domain is not easy due to three main reasons. First, the effects of a failure, in the form of abnormal system states, propagate due to system dynamics. Therefore, abnormal system states usually cannot be uniquely associated with specific failures. Second, even when such a unique relationship can be defined it can only be done for steady state values. However, the system seldom attains a steady state. Third, even if the system attains a steady state it may take a long time to do so. Troubleshooting, on the other hand, must begin immediately, or else it may have catastrophic consequences.

The diagnostic problem solving task in a real system is further complicated by the operator's inability to observe all abnormal system behaviors. This is due to the limited number of available gauges. This limitation prevents the operator from accessing pressures, temperatures, and flow measurements across every component. Thus, the operator has to effectively utilize the available diagnostic information to identify the malfunctioning component.

Training for the troubleshooting task in a marine power plant can be at several levels depending upon the educational backgrounds and experience of the student operators. A comprehensive computer-based instructional system that can provide training at all levels, although ideal, is a "more long term" goal. Instead, a more pragmatic approach is to design a training program that can satisfy the diverse needs of a homogeneous population with respect to their educational background and experience.

Turbinia-Vyasa, the instructional system developed as a part of this research effort has a limited but pragmatic pedagogical goal. It aims at improving the troubleshooting skills of marine engineers and naval personnel who learn to operate the power plant as a part of their curriculum. A brief description of the knowledge, skills and experience of a typical student who will be trained on **Turbinia-Vyasa** is provided next.

Student Operator

A novice operator in training, although unfamiliar with the faults in the power plant, has a basic understanding of the theory of power generation. The student is usually also familiar with the names and functions of the individual components but does not completely understand the integration of the components into the system, their role in achieving the higher level system goals, their interaction with other components and the dynamics of the system. Very few students are knowledgeable about the detailed structural layout of the power plant even though they may have a general idea about the locations of individual components. Therefore, the students who will use **Turbinia-Vyasa**, in addition to practical experience and exposure to failure situations, need to consolidate their knowledge of the structure, function and behavior of the power plant and its components.

This concludes the description of the instructional system's domain. The next section provides a description of **Turbinia**, the marine power plant simulator, used in the instructional system.

Turbinia: The Simulator

Turbinia, the marine power plant simulator, was designed using qualitative approximation to represent system dynamics. It can simulate a large number of failures in a marine power plant and provides a good environment for teaching troubleshooting. It is an enhanced version of QSTEAM (Govindaraj, 1987) and PEQUOD (Fath, 1987) that is more robust, modular, and suitable for application in training programs. However, it retains the basic notion of hierarchical representation of components used in the earlier versions.

The qualitative approximation methodology is based on a hierarchical description of the system. The primitives that form the basic units of the hierarchy 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. In **Turbinia**, approximately 100 components have been modeled to achieve fairly high degrees of structural and dynamic fidelity even though the physical fidelity of the simulator is low.

In modeling the system hierarchy, the components in the power plant have been categorized into two basic primitive types: simple and composite. The simple primitives have a single fluid flowing in and out of them. The fluid may, however, exist in different phases within the simple primitive and may flow to and from multiple components. The composite primitives, on the other hand, have two fluids flowing through them.

Simple as well as composite primitives are further subdivided depending upon the function they perform. In **Turbinia**, there are twelve simple primitives and one composite primitive. The simple primitives are: capacitor, conduit, controller, convertor, gain, double-gain, phase-changer, reactor, sink, source, source-sink, and transducer. The heat-exchanger is the only composite primitive. All components in the simulated power plant are instances of simple and composite primitives. A summary of system decomposition into simple and composite primitives is shown in Figure 4.5.

The knowledge concerning the components of the power plant, represented in **Turbinia**, is of two types. One is structural knowledge that includes information such as connections to other components, gauges attached, and fluids flowing through the component. The other is knowledge needed to compute the evolution of system states during simulation. This knowledge consists of equations that approximate the functions of the primitive, including parameter values necessary to maintain temporal fidelity of state evolution during normal and failed states of the components. A complete description of the details of implementation are provided later in a separate section.

Turbinia uses pressure, temperature and flow or level to describe the states of the simulated marine power plant. Abnormal system states are described in terms of deviations of pressure, temperature and flow or level from their nominal value. Therefore state value corresponding to normal operation is zero in all cases. State values are computed from functionally approximate system equations and are expressed in a qualitative manner. There are five qualitative values used to describe the system states: low, slightly low, normal, slightly high, and high.

Evolution of system states during simulation is performed in two steps. First, the states of the individual components are updated, one at a time, using the approximate system equations. The computation involves solving the equations for each component using parameter values appropriate for the current status of the component. The current status of the component can have four values: normal and unaffected, normal but indirectly

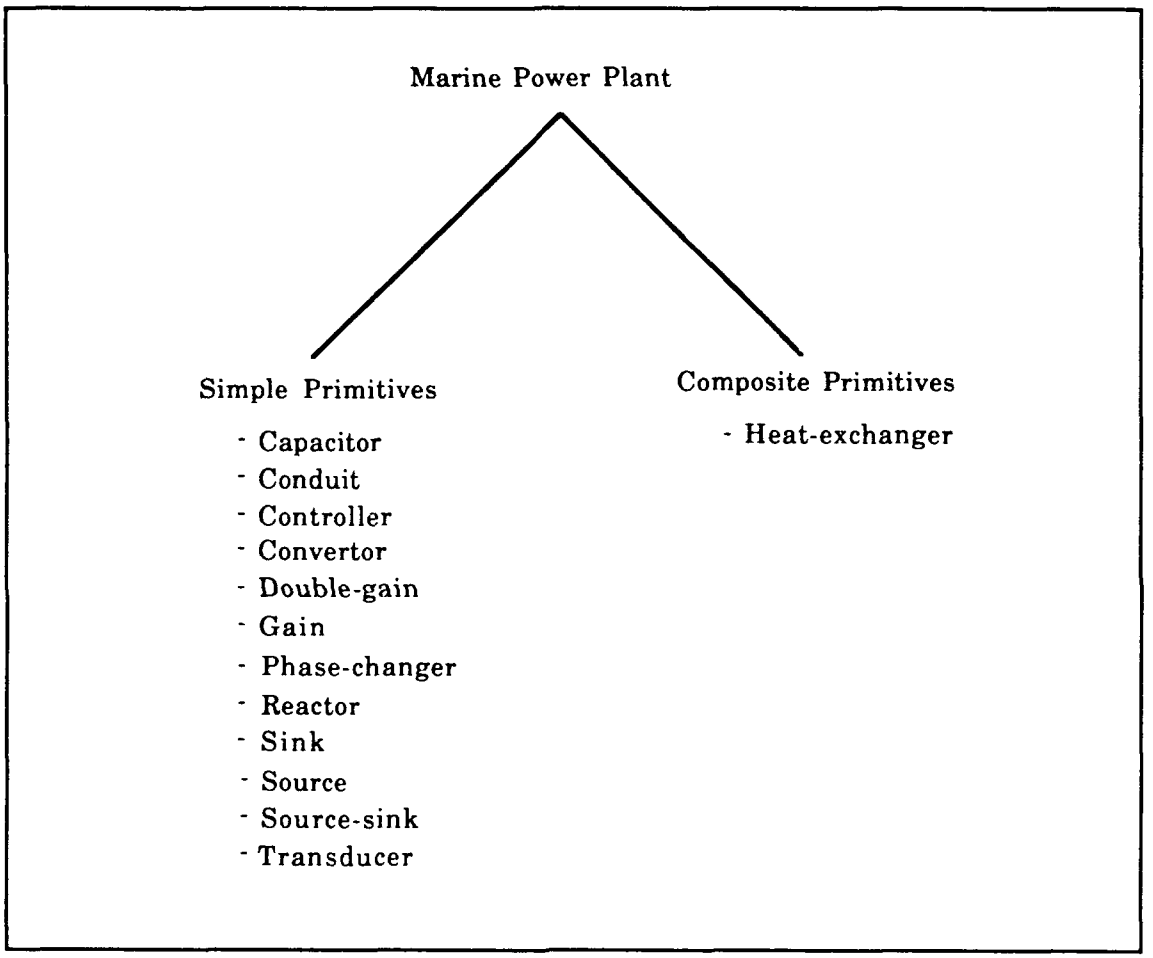


Figure 4.5 System Hierarchy

affected, normal but directly affected, and abnormal or failed. While all components that have abnormal state values across them are considered affected, only those that experience the effects of failure due to interaction with the failed component are said to be directly affected. Others that experience the effects of failure due to propagation are considered indirectly affected.

For normal and unaffected components as well as normal but indirectly affected components the computed states are zero (normal) till the input states of the component become abnormal (i.e., non-zero). When the input states become abnormal, the parameter values determine the effect of failure that gets propagated across the component.

For normal but directly affected components as well as the failed component, the parameter values determine the extent of deviation from the normal. While the parameter values for the failed component are used to compute the deviation from the normal from the very beginning of the simulation, the parameter values for normal but directly affected components are used after some pre-determined time interval to maintain temporal fidelity of state evolution.

After computing the states of the individual components, including the failed component, the next step in state evolution involves propagating the updated states to successor components. Propagation involves transmitting the output states of all components to input states of the successor components. The simulation proceeds by repeating the cycle of alternately updating and propagating the system states. For the purpose of maintaining the temporal fidelity of state evolution, the time interval between cycles is adjusted for individual faults.

In addition to the primitives discussed, the simulator has knowledge of the gauges in the system. There are three types of gauges: pressure, temperature and flow or level. The flow and the level gauges are identical except that the level gauges are mounted on tanks and flow gauges appear between components. Wherever available, these gauges display the qualitative values of system state. These values are updated whenever the states of the component adjacent to the gauge on either side is updated.

Finally, the simulator also has knowledge of the graphical representations of components, gauges and icons used by its interactive user interface. This knowledge includes responses to user actions at the interface.

This completes a description of the simulator **Turbinia**. Further implementation details of the simulator and its user interface are provided later. The next section describes the computer-based tutor **Vyasa**.

Vyasa: The Computer-Based Tutor

Vyasa is a computer-based intelligent tutor that trains operators to troubleshoot the marine power plant simulated by **Turbinia**. **Vyasa** operates in two modes: passive and active.

In the passive mode the student is solely responsible for initiating the communications. When the passive tutor is invoked, the simulation is temporarily brought to a halt and the student can access various segments of knowledge in the expert module.

In the active mode, the tutor takes the initiative to provide instructions when it evaluates a possible misconception based on the student's actions. The instructions may be provided by the active tutor with or without intervention. The capabilities of active tutor include all the capabilities of the passive tutor as well.

This section describes the knowledge organization in **Vyasa**. Complete details of the implementation and a description of the student's interaction with **Vyasa** are provided later in a separate section.

Vyasa uses the framework for knowledge organization proposed in the preceding chapter. Knowledge in the tutor is comprised of:

(1) system knowledge,
(2) failure knowledge,
(3) knowledge of student actions,
(4) knowledge to update the student model,
(5) knowledge to evaluate misconceptions, and
(6) instructional knowledge.

The expert module of Vyasa contains the system and failure knowledge. The rest of the knowledge is contained in the instructional module of Vyasa. A complete discussion of Vyasa's knowledge appears next.

System Knowledge

System knowledge in Vyasa is comprised of fluid paths, functional subsystems, and schematics (Figure 4.6). Each fluid path is made up of components sharing a common fluid in the power plant. Each subsystem is a collection of components responsible for an important system function. Each schematic is a pictorial representation of a section of the power plant. These three constituents of system knowledge are interrelated. For instance, a fluid path may appear in multiple subsystems and schematics. Similarly, a subsystem may contain several fluids and may span over multiple schematics. Also, a single schematic may contain several fluid paths and subsystems. Each constituent of system knowledge is now described in further detail.

Fluid paths

Representation of a power plant as a collection of fluids is done by decomposition of the system into thirteen fluid paths. They are: combustion air, fuel oil, flue gas, feed water, superheated steam, desuperheated steam. steam, main condenser hot fluid, condensate, main condenser cold fluid, saltwater, control air, and lube oil. Among these thirteen paths, six represent different segments of a single continuous closed loop water path. Water that flows through this closed loop is called feed water prior to entering the boiler, steam at the boiler exit, superheated steam past the superheater, desuperheated steam at the desuperheater exit, main condenser hot fluid in paths leading to the condenser and condensate in the paths feeding to the deaerating feed tank.

The fluid path knowledge in Vyasa includes information such as the name of the fluid, and the name of the subsystems and schematics in which the fluid is found. Also included is a list of connectors and components in each schematic that contains the fluid.

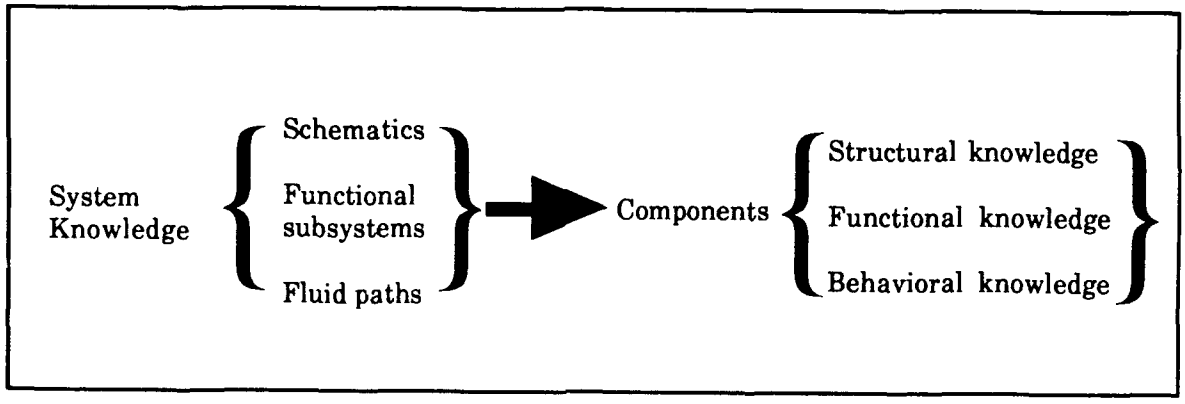


Figure 4.6 System Knowledge Decomposition

Functional subsystems

Functional decomposition of the power plant is done via nine subsystems. They are: combustion, steam generation, auxiliary steam use, power generation, steam condensation, feed water preheating, lubrication, control air, and saltwater service.

Steam generation, power generation, steam condensation and feed water preheating subsystems are responsible for the four major functions performed in the power plant during the four phases of the steam cycle described earlier.

The combustion subsystem is responsible for burning the fuel-air mixture to release thermal energy for heating water in the boiler. The auxiliary steam use subsystem is responsible for operating the auxiliary units of the power plant. This subsystem uses low pressure desuperheated steam to run auxiliary equipment such as the feed water pump, fuel pump, saltwater service pump and the forced draft fan. It also uses the desuperheated steam to preheat the fuel oil and the feed water and to prevent leakage of air into and out of the turbine casings. The interaction between all these subsystems to produce power is summarized in Figure 4.7.

The remaining subsystems perform other functions necessary for the safe operation of the power plant. The control air subsystem is responsible for distributing control air to many valves and regulators operated by control air. The lubrication subsystem lubricates moving parts and removes the heat produced by friction. The saltwater service subsystem distributes the cold sea water to remove heat from units dissipating heat.

The subsystem level knowledge represented in Vyasa consists of the name of the subsystem, its primary function, the names of fluids present in the subsystem and the names of the schematics in which the whole or part of the subsystem can be viewed. In addition, a list of connectors and components that constitute the subsystem along each fluid path in each of the relevant schematic is also included.

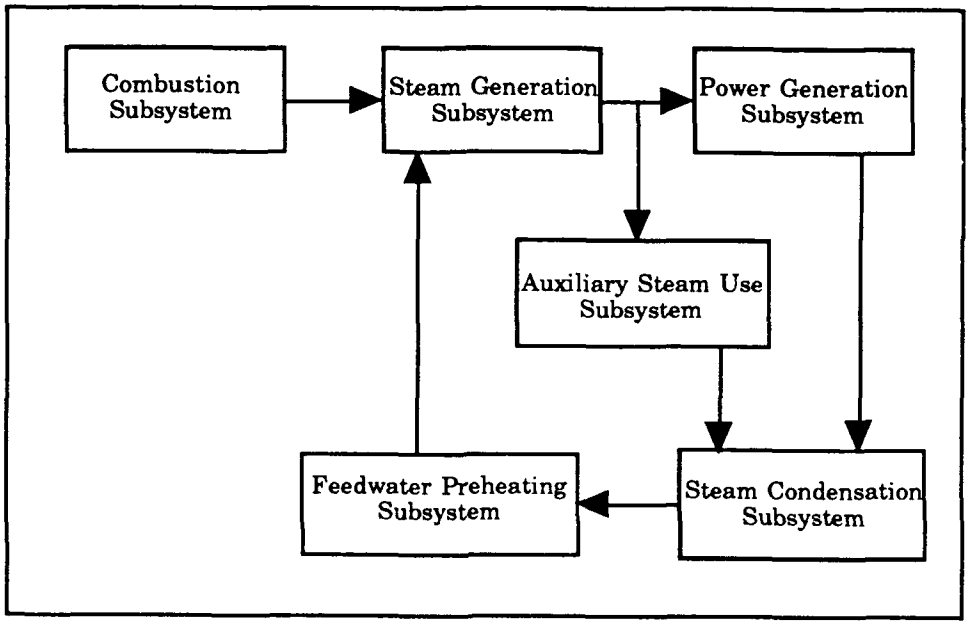


Figure 4.7 Interacting Subsystems of Marine Power Plant

Schematics

In Vyasa, the structural view of the power plant is provided by seven schematics: boiler, steam, feed water, fuel oil, control air, saltwater, and lube oil. Each schematic contains one or more subsystems and fluid paths.

The boiler, steam, feed water, and fuel oil schematics contain the main subsystems responsible for the production of power (shown in Figure 4.7) and all the six segments of the closed loop water path. A complete description of the interactive schematic interface is provided later in this chapter.

The schematic knowledge represented in Vyasa includes information such as the names of the components, subsystems and fluid paths, list of icons displayed, list of graphical objects that represent the components and connectors, and a list of gauges. In addition, the schematic knowledge of Vyasa includes information about regions of the schematic that are sensitive to mouse clicks, records of regions picked by the user, and instructions for highlighting or lowlighting the picked region.

Components

System knowledge at the component level concerns a component's structure, function and behavior. In Vyasa, the structural, functional and behavioral knowledge of components is organized at the level of detail necessary for the troubleshooting task. A component's structural knowledge refers to its input and output connections to other components, fluids carried by it, gauges attached to it and its association with a functional subsystem and schematic. The functional knowledge is a description of the purpose of the component in the system and its contribution to the higher level system function. The component's behavioral knowledge describes the manner in which the system state values are affected by the presence of the component in both the normal and failed states.

In addition, the system knowledge at the component level includes the modes in which the component can fail, the parameters that are used to compute the component's behavior in the failed state, and the component's link to the graphical object that represents it on the schematic interface.

Failure Knowledge

The failure knowledge in **Vyasa** is organized in terms of modes of failure and specific failures. Specific failures are faults simulated by **Turbinia**. Knowledge of failure modes helps the tutor to teach the student about typical abnormal behaviors associated with each mode. Knowledge of specific instances of failure helps the tutor in evaluating student's actions. Both components of failure knowledge are described below.

Modes of Failure

Vyasa has knowledge of four most common modes of failure for components of **Turbinia**: (a) blocked-shut, (b) stuck-open, (c) leak-in, and (d) leak-out. Knowledge of each mode of failure includes the typical system behavior associated with it, possible reasons for deviations from expected abnormal behavior, and names of components in the system that are known to fail in that mode.

Knowledge of blocked-shut mode of failure, for instance, includes the upstream and downstream abnormal system behaviors expected in liquid and gas paths, the type of component that can curtail the propagation of abnormal behavior, and the names of the component in the power plant that are commonly known to fail in the blocked-shut mode. A summary of typical system behavior and the conditions that curtail the propagation of abnormal behavior for each of the four modes is shown in Table 4.1.

Specific Failures

Knowledge of specific failures includes initial symptoms, cause and mode of failure. In addition, the subsystems, fluid paths, schematics, components and gauges affected by each failure are included in the tutor's knowledge of specific failures.

Vyasa also has access to "pre-defined" explanations of cause-effect associations that describe the propagation of abnormal gauge readings under each failure condition. These explanations, along with diagnostic tests that serve as evidence for or against the specific failures, form an essential part of the tutor's failure knowledge.

Table 4.1 Typical Abnormal System Behavior

Failure Mode	Fluid	State	Abnormal Behavior		Propagation Limited By	
			Upstream	Downstream	Upstream	Downstream
Blocked-Shut	Liquid	Level	High	Low	Infinite Sink	Infinite Source
	Gas	Pressure	High	Low	Safety Valve	Infinite Source
Stuck-Open	Liquid	Level	Low	High	Infinite Source	Infinite Sink
	Gas	Pressure	Low	High	Infinite Source	Safety Valve
Leak-In	Liquid	Level	High	High	Infinite Sink	Infinite Sink
	Gas	Pressure	High	High	Safety Valve	Safety Valve
Leak-Out	Liquid	Level	Low	Low	Infinite Source	
	Gas	Pressure	Low	Low		

Knowledge of Student Actions

A computer-based tutor must have some means of evaluating the purpose of student's actions. This capability enhances the tutor's ability to infer misconceptions. **Vyasa** infers student's misconceptions based on student's actions using its knowledge of the valid forms of interactions at the student-tutor interface.

A student interacts with **Turbinia-Vyasa** in three modes: troubleshooting mode, tutor dialog mode, and diagnose mode. In the troubleshooting mode the student interacts with the simulator only. In the tutor dialog mode, the interaction is with **Vyasa**. In the diagnose mode, the student attempts to identify the failed component.

In all the three modes of interaction, **Vyasa** recognizes nine types of valid actions: *call-for-schematic-action, investigative-action, informative-action, diagnose-request-action, diagnostic-action, help-request-action, resume-request-action, tutor-dialog-action* and *modal-dialog-action*. A *call-for-schematic-action* is performed to call a new schematic or switch between schematics. An *investigative-action* is performed to view the gauges attached to a component. An *informative-action* usually follows the investigative-action and is taken to display the gauge readings. A *diagnose-request-action* is taken to switch to the diagnostic mode. The *diagnostic-action* is the action of identifying the failed component. The *help-request-action* and the *resume-request-action* switch the student to the tutor dialog and the troubleshooting modes respectively. *Tutor-dialog-actions* are all actions taken in the tutor dialog mode. Finally, *modal-dialog-actions* terminate interactions with dialogs.

Vyasa keeps a record of all recognized actions in a student model. This record is updated and used to determine student's misconceptions after every action.

Knowledge to Update the Student Model

The student model in **Vyasa** is a dynamic data structure that maintains a record of actions taken by the student. Each student action, if recognized as one of the nine types of valid actions, is time stamped and information relevant to the action is stored. In this manner, the student model keeps account of the schematics viewed, the order in which they were viewed, the sequence of subsystems and fluid paths explored, the components investigated, the gauges probed and their gauge readings at the time of the investigation.

After each student action, the subsystem and the fluid path most suspected by the student is determined. A count is kept of the number of investigations made in each subsystem and fluid path. The subsystem and the fluid path with the maximum number of investigative-actions are also the most suspected if at least one of the last three investigations have occured in that subsystem or fluid path. Otherwise, the most suspected subsystem or the most suspected fluid path is the one investigated last. Thus, after every action, the information concerning the most suspect subsystem and the most suspect fluid path in the student model is revised.

In addition to recording the actions, the student model maintains a list of the student's past and current hypotheses concerning the failure. This information is a list of components with their respective modes of failure, which, in the opinion of the student, explains the observed abnormal system behavior. This information on student's failure hypotheses is directly elicited from the student when **Vyasa** functions in the active mode.

Knowledge to Evaluate Misconceptions

The knowledge to evaluate misconceptions gives **Vyasa** the ability to deliver individualized instructions. This knowledge has a rule-based structure. These rules are used to identify three types of misconceptions based on observed student actions.

The first misconception concerns deficiency in student's knowledge of system structure. **Vyasa** identifies this structural misconception when the student investigates components in a schematic unaffected by the current failure. Knowledge of schematics affected by each failure, needed to evaluate the structural misconception, is obtained from the tutor's knowledge of the failures.

The second misconception concerns deficiency in student's knowledge of system functions. **Vyasa** identifies this functional misconception when the most suspected subsystem or fluid path inferred from the student's action is unrelated to the failure being investigated. Knowledge of subsystems and fluid-paths related to each failure, needed to evaluate the functional misconception, is obtained from the tutor's knowledge of the failures.

The third misconception concerns deficiency in student's knowledge of fault related system behavior. Vyasa identifies this behavioral misconception when a student continues to pursue a failure hypothesis that should have been rejected based on the diagnostic evidence available. As in the identification of the first two types of misconceptions, the additional information required to evaluate behavioral misconception is available to the tutor. For example, probable evidence against each failure in terms of diagnostic test results is stored within the tutor's knowledge of failures and actual tests conducted by the student are stored in the student model. Thus, by comparison, the tutor can determine if a diagnostic test that suggests the elimination of a hypothesis has been conducted.

After Vyasa has evaluated a student's misconceptions, it uses its knowledge to deliver instructions to rectify the misconceptions. Knowledge concerning delivery of instructions is described under instructional knowledge.

Instructional Knowledge

Instructional knowledge concerns instructional content, its form and time of presentation. The instructional content is either extracted verbatim from the tutor's knowledge base or is generated using a template. The form of instructional presentation is either textual or graphical. The time of presentation of instructional material is always issue-based which encourages intervention at particular occasions. Knowledge related to content, form and time of instructional presentation is discussed in further detail below.

Content

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

There are instances when the instructions are not extracted verbatim from the tutor's knowledge base. Instead, context-specific information relevant to the actions of the student

is generated to fill an appropriate instructional template. Such an instruction generation typically occurs when the student seeks advice concerning a failure hypothesis. For example, when advice is sought about a hypothesis which should not be pursued based on evidence gathered, the tutor generates instructions suggesting hypothesis refinement. The tutor generates this advice using an instructional template shown in Figure 4.8a. The instructional template sets the general tone of the advice but does not contain the evidence to be cited for hypothesis revision. The evidence, which is a diagnostic test already conducted by the student, is inserted in the instructional template during advice generation to provide the context-specific meaning to the advice.

The instructional template used for advice generation is the same on all occasions for the same type of advice. In all there are four such instructional templates used by the tutor (See Figures 4.8a, b, c, and d). Each template is used for a different type of advice. One is used for suggesting hypotheses revision, another to indicate lack of adequate evidence to pursue a hypothesis, a third to suggest a diagnostic test to strengthen or weaken a hypothesis; and a fourth to convey the tutor's inability to provide advice under existing conditions.

Form

Like content, the form of instructional presentation is also context-dependent. For example, answers to queries related to subsystems and fluid paths that can be visualized on the simulator interface are presented graphically. Therefore, when the student wants to know the components that constitute a particular subsystem, showing these components by highlighting them in schematics is preferred over listing the names of the components in a dialog box.

Where graphics is unlikely to enhance the understanding of the instructions or where graphical aid is computationally expensive the instructions are presented in the form of text. For example, the function of components can be eloquently and adequately described in text. Using graphics or supplementing text with graphics to describe component functions is tedious, computationally expensive and unlikely to provide additional information.

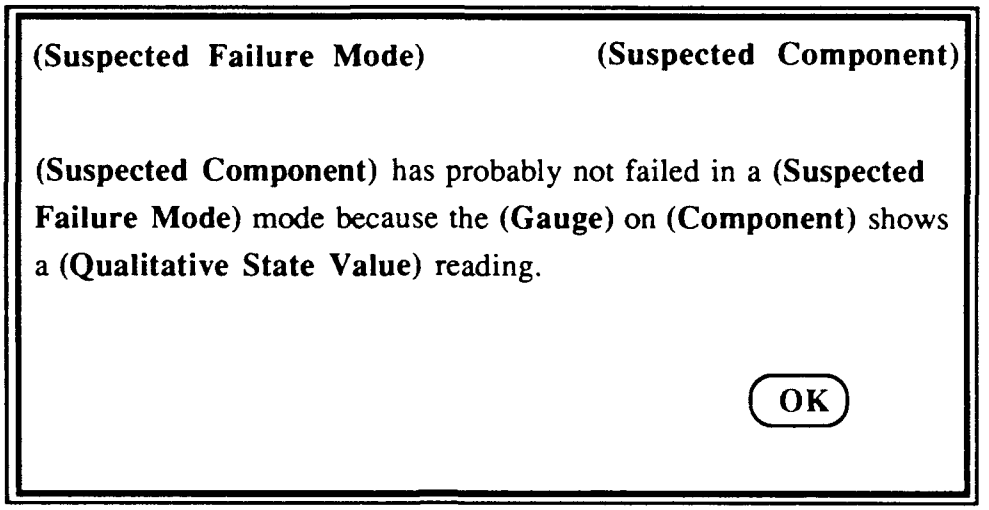


Figure 4.8a Instructional Template to Suggest Hypothesis Revision

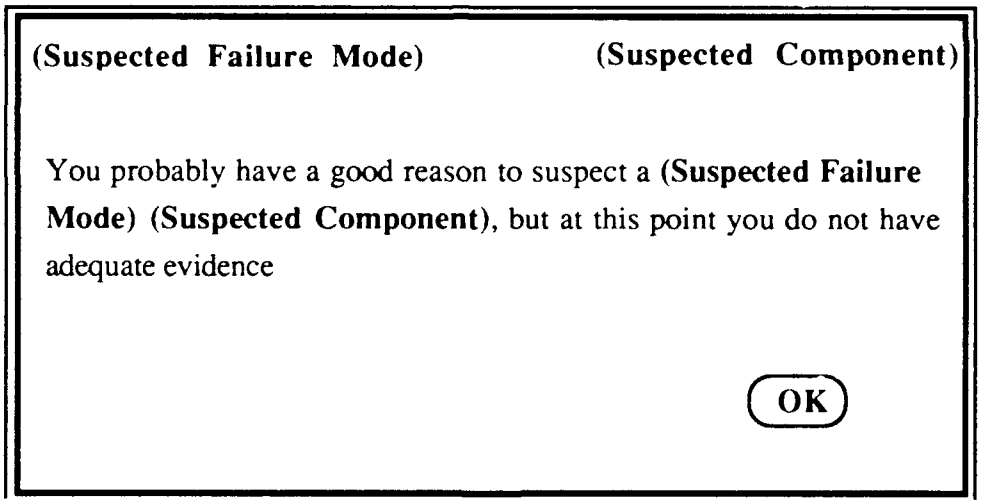


Figure 4.8b Instructional Template to Indicate Lack of Evidence

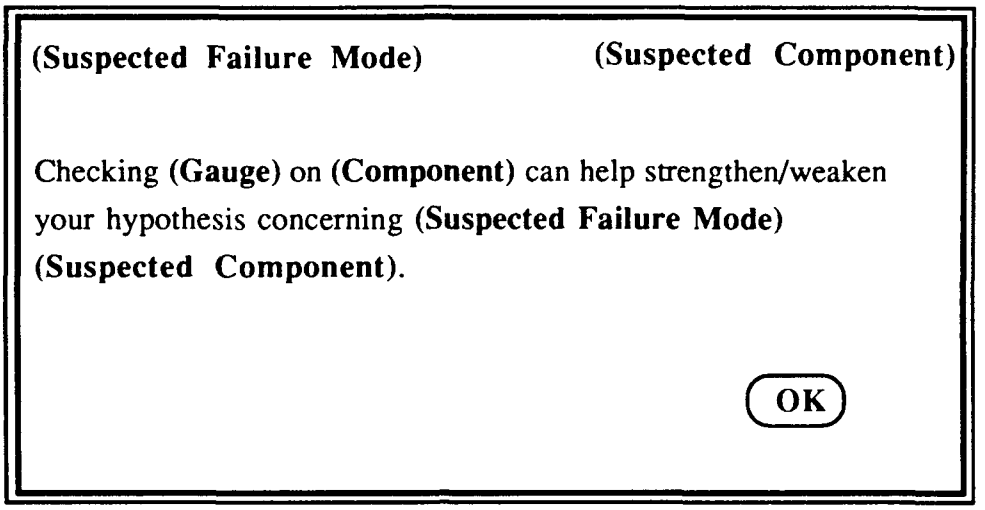


Figure 4.8c Instructional Template to Suggest a Diagnostic Test to Strengthen or Weaken a Hypothesis

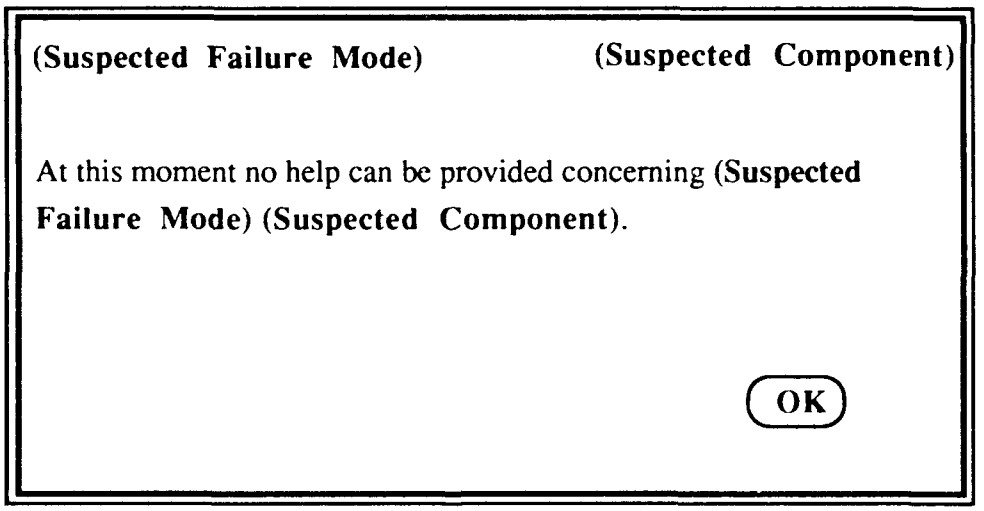


Figure 4.8d Instructional Template to Express Inability to Provide Help

However, where graphics supplemented with text may help students retain vital information or reduce errors, a dual mode of presentation is adopted. For example, there are four icons that are commonly used to represent the four modes of failure. These icons indicate the structural changes in the failed component associated with the mode of failure they represent. Although clearly distinguishable, these icons are accompanied by text in dialog boxes used by the students to indicate the suspected mode of failure. Thus, an icon representing a blocked-shut mode of failure also has "blocked-shut" written under it. This dual mode of presenting information reduces the inadvertent error of incorrect selection of a failure mode while indicating the suspected mode of failure.

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

Time and Duration

Apart from instructional content and its form of presentation, **Vyasa** has knowledge about time and duration of instructional presentation. Instructions that are given on request from the student are provided immediately in a dialog box or in the special tutor communication window. These instructions are displayed for an unspecified duration of time until the student makes a new request. In some cases, however, the student is required to acknowledge the receipt of instructions before making a new request.

In addition to the instructions presented on request, **Vyasa** delivers instructions on its own. **Vyasa** delivers these instructions with and without intervention depending on the context.

Instructions with Intervention

When the tutor identifies a misconception, instructions to rectify the misconceptions are presented immediately with a beep. The tutor has different sets of instructions for each of the three types of misconception it identifies. The instructional sets for structural and functional misconceptions inform the student that a region unaffected by the current failure is being investigated unnecessarily. The idea is to suggest that the student only investigate portions of the system relevant to the failure. Similarly, when behavioral

misconceptions are identified, the student is instructed to refine the failure hypotheses based on the evidence available.

Since the instructions to rectify misconceptions are highly context-sensitive and may change with every new action taken by the student, they need to be presented only as long as they retain their context-sensitivity. For example, when the student investigates an unaffected schematic the tutor informs the student that the schematic being investigated is unaffected by the failure. However, it makes sense for the tutor to display such an instruction only while the unaffected schematic is being viewed. But, if the student quickly switches to an affected schematic while the instructions have not been displayed long enough for the student to read them, the instructions are not erased from the tutor window. Instead, they continue to be displayed for a specified minimum duration of time necessary to read them.

Sometimes the student ignores the instructions given by the tutor. Whenever it is obvious from new actions that the student has ignored the instructions, the same instructions must be presented again. In such cases, **Vyasa** delivers the same instructions again with a beep instead of extending the duration of display of the previous instruction. This increases the likelihood of drawing the attention of the student to the instructions.

Instructions without Intervention

Instructions are also provided without intervention at the end of the training session. These instructions include explanations of abnormal gauge readings for the failure investigated during the session. These are causal explanations that animate the effects of fault propagation and are presented in a chronological order. For each failure, these explanations are stored as a pre-defined set in the failure knowledge of **Vyasa**.

This completes the description of the components of knowledge in **Vyasa**. The implementation details such as the method of knowledge representation and controls in **Turbinia-Vyasa** are discussed next.

Knowledge Representation and Control Structure

Knowledge Representation

Vyasa uses objects to encapsulate the components of knowledge described above. Most of the knowledge is represented in a declarative form and is amenable to changes. Most of the procedures that manipulate data are not stored separately but are also encapsulated within the objects. Object-oriented programming features such as inheritance and polymorphism have also been effectively employed. Objects that are instances of the same class have similar representations and share methods that create and manipulate data. A detailed description of the abstract data types used for representing knowledge is provided below. Throughout this section, the object classes are shown in a sans serif (Helvetica) font.

Fluid paths

Knowledge concerning the thirteen fluid paths is represented in instances of an object class called FluidPath (Table 4.2). FluidPath has six variables: *fluid-path-name, in-schematic, in-subsystem, components, connectors, and fluid-path-schematic-association*. *Fluid-path-name* stores the name of the fluid path. Variables *in-schematic* and *in-subsystem* store the names of the schematic and subsystems in which the fluid path is found. A list of component names that lie along the fluid path is stored in the variable *components* and a list of instances of the type connector that represent connections between these components is stored in the variable *connectors*.

Fluid-path-schematic-association contains information that separates the components and connectors in the fluid path by schematic. This information is organized in a list of paired associations. Each paired association consists of a schematic name and an instance of FluidPathStructureWithinSchematic (Table 4.3) which stores structural information for the segment of the fluid path that lies within the paired schematic. This structural information consists of a list of component names and connector objects that lie in the fluid path within the paired schematic. Since each paired association contains structural information for the segment of the fluid path in one schematic, the total number of paired associations in the whole list is the same as the number of schematics in which the fluid path is found.

Table 4.2 Description of Object Class FluidPath

Class: FluidPath	
Class Variables	**Description**
fluid-path-name	name of the fluid path
in-schematic	schematics in which the fluid path is found
in-subsystem	subsystems in which the fluid path is found
components	component s that lie in the fluid path
connectors	Connector objects that lie in the fluid path
fluid-path-schematic-association	list of paired associations. Each pair stores a name of schematic and an instance of FluidPathStructureWithinSchematic

Table 4.3 Description of Object Class FluidPathStructureWithinSchematic

Class: FluidPathStructureWithinSchematic	
Class Variables	**Description**
components	component s that lie in the fluid path within a single schematic
connectors	Connector objects that lie in the fluid path within a single schematic

An example of a fluid path in the simulated marine power plant is the fuel oil path. The fuel oil path is found in fuel oil and boiler schematics. Components that lie along the fuel oil path are fuel oil settling tank, fuel oil service pump, fuel oil hp regulator, fuel oil heater, fuel oil discharge strainer, fuel oil control valve, master fuel oil valve and the burner. All except the burner in the fuel oil path are displayed on the fuel oil schematic. The object ***FuelOilPath*** is shown in Table 4.4. Instances of all the thirteen fluid paths are created prior to run time and are initialized from data in input files.

Table 4.4 Description of ***FuelOilPath***

Object: *FuelOilPath* **Instance of:** FluidPath	
Instance Variables	**Value**
fluid-path-name	fuel-oil-path
in-schematic	(fuel-oil-schematic boiler-schematic)
in-subsystem	(combustion-subsystem)
components	(fuel-oil-settling-tank fuel-oil-service-pump fuel-oil-hp-regulator fuel-oil-heater fuel-oil-discharge-strainer fuel-oil-control-valve master-fuel-oil-valve burner)
connectors	((#<connector 4003344> #<connector 4003276> #<connector 4003208> ----------)
fluid-path-schematic-association	((fuel-oil-schematic #<FluidPathStructureWithinSchematic 4043316>) (boiler-schematic #<FluidPathStructureWithinSchematic 4043116>))

Object: #< 4043116> **Instance of:** FluidPathStructureWithinSchematic	
Instance Variables	**Value**
components	(burner)
connectors	(#<connector 4020708>)

Functional subsystems

Knowledge concerning the nine subsystems is represented in instances of an object class called FunctionalSubsystem (Table 4.5). FunctionalSubsystem has seven variables: *subsystem-name, in-schematic, fluid-paths, components, connectors, function and subsystem-schematic-association.*

Subsystem-name stores the name of the subsystem. Variables *in-schematic* and *fluid-paths* store the names of the schematics in which the subsystem is visible and the names of the fluid paths that pass through the subsystem. A list of component names that constitute the subsystem is stored in the variable *components* and a list of instances of the type connector that represent connections between these components is stored in the variable *connectors*. *Function* stores the description of the function performed by the subsystem.

Subsystem-schematic-association contains a list of paired associations. Each paired association consists of a schematic name and an instance of SubsystemStructureWithinSchematic (Table 4.6). As in the case of fluid paths, these pairs separate the components and connectors in the subsystem by schematic. Each instance of SubsystemStructureWithinSchematic stores structural information concerning the portion of the subsystem that lies within a single schematic. Since each paired association contains subsystem structural information for a single schematic, the total number of such paired associations in the whole list is equal to the number of schematics in which the subsystem is found.

Each instance of SubsystemStructureWithinSchematic has four variables. One contains a list of component names, the second a list of connector objects, and the third a list of fluid path names. These components, connectors and fluid paths are those that are found in the portion of the subsystem within a single schematic. The fourth variable, *subsystem-fluid-path-association*, contains paired associations of the fluid path names and instances of SubsystemStructureWithinFluidPath (Table 4.7) to further separate the subsystem's components and the connectors by fluid path in each schematic.

Table 4.5 Description of Object Class FunctionalSubsystem

Class: FuntionalSubsystem	
Class Variables	**Description**
subsystem-name	name of the subsystem
in-schematic	schematics in which the subsystem is visible
fluid-paths	fluid paths that pass through the subsystem
components	components that constitute the subsystem
connectors	Connector objects that connect the components in the subsystem
function	function performed by the subsystem
subsystem-schematic-association	list of paired associations. Each pair contains a name of schematic and an instance of SubsystemStructureWithinSchematic

Table 4.6 Description of Object Class SubsystemStructureWithinSchematic

Class: SubsystemStructureWithinSchematic	
Class Variables	**Description**
fluid-paths	fluid paths in a single schematic that pass through the subsystem
components	components that constitute the portion of the subsystem within a single schematic
connectors	Connector objects that connect the components in the portion of the subsystem within a single schematic
subsystem-fluid-path-association	list of paired associations. Each pair has the name of a fluid path that passes through the portion of the subsystem within a single schematic and an instance of SubsystemStructureWithinFluidPath

Table 4.7 Description of Object Class SubsystemStructureWithinFluidPath

Class: SubsystemStructureWithinFluidPath	
Class Variables	**Description**
components	components that lie in a fluid path that passes through the portion of the subsystem within a single schematic
connectors	Connector objects that lie in a fluid path that passes through the portion of the subsystem within a single schematic

An example of a functional subsystem in the simulated marine power plant is the combustion subsystem. The combustion subsystem spans over the fuel oil and boiler schematics. In the fuel oil schematic, the combustion subsystem has only fuel oil flowing through it. In the boiler schematic, the combustion subsystem has fuel oil as well as combustion air fluid paths in it. The combustion subsystem is comprised of fuel oil settling tank, fuel oil service pump, fuel oil hp regulator, fuel oil heater, fuel oil discharge strainer, fuel oil control valve, and the master fuel oil valve in the fuel oil schematic; and forced draft fan, air heater, windbox, air register and the burner in the boiler schematic. Of the components in boiler schematic, the burner is located in the fuel oil path and the rest lie along the combustion air path. The ***CombustionSubsystem*** object is shown in Table 4.8. Instances of all the nine functional subsystems are created prior to run time and are initialized from data in input files.

Table 4.8 Description of ***CombustionSubsystem***

Object: *CombustionSubsystem* **Instance of:** FuntionalSubsystem	
Instance Variables	**Value**
subsystem-name	combustion-subsystem
in-schematic	(boiler-schematic fuel-oil-schematic)
fluid-paths	(fuel-oil-path combustion-air-path)
components	(fuel-oil-settling-tank fuel-oil-service-pump fuel-oil-hp-regulator fuel-oil-heater fuel-oil-discharge-strainer fuel-oil-control-valve master-fuel-oil-valve forced-draft-fan air-heater windbox air-register burner)
connectors	(#<connector 4020096> #<connector 4020028> ----)
function	"To mix the combustion-air with fuel and ignite it in the burner to release thermal energy"
subsystem-schematic-association	((fuel-oil-schematic #<SubsystemStructureWithinSchematic 4042304>) (boiler-schematic #<SubsystemStructureWithinSchematic 4042504>))

Object: #<4042504> **Instance of:** SubsystemStructureWithinSchematic	
Instance Variables	**Value**
fluid-paths	(fuel-oil-path combustion-air-path)
components	(forced-draft-fan air-heater windbox air-register burner)
connectors	(#<connector 4020028> #<connector 4020640> ----)
subsystem-fluid-path-association	((combustion-air-path #<SubsystemStructureWithinFluidPath 3773792>) (fuel-oil-path #<SubsystemStructureWithinFluidPath 3773992>))

Object: #<3773992> **Instance of:** SubsystemStructureWithinFluidPath	
Instance Variables	**Value**
components	(burner)
connectors	(#<connector 4020708>)

Schematics

Knowledge concerning the seven schematics is represented in instances of an object class called Schematic (Table 4.9). Table 4.10 shows ***BoilerSchematic***, an instance of Schematic with most of its instance variables initialized from an input data file. Schematic has seventeen class variables which are described below.

The variable *ccl-window* contains an instance of color-window, a system-defined subclass of Macintosh windows. Color windows are used by the system to display information on the screen.*Title* stores the title of the schematic window. *Schematic-name* contains the name of the schematic. *Components-in-schematic*, *fluid-paths-in-schematic*, and *subsystems-in-schematic* store the names of the components, fluid paths and subsystems visible in the schematic respectively. *Picture-handle* contains a handle to the Macintosh picture resource that is displayed in the schematic. *High-light-function* and *low-light-function* are variables that point to procedures that highlight or lowlight selected regions within the schematic window. *Button-event-function* points to a default function for system response when the mouse is clicked within the schematic window.

Five other Schematic class variables, *list-of-graphic-components*, *list-of-graphic-connectors*, *list-of-graphic-icons*, *list-of-visible-graphic-gauges* and *list-of-graphic-gauge-readings* store references to objects that represent components, connectors, icons, gauges, and gauge readings in the schematic respectively. These objects are called graphical objects.

Table 4.9 Description of Object Class Schematic

Class: Schematic	
Class Variables	**Description**
ccl-window	an instance of system-defined color-window
title	name printed on the title bar of schematic
schematic-name	name of the schematic
components-in-schematic	components in the schematic
fluid-paths-in-schematic	fluid paths in the schematic
subsystems-in-schematic	subsystems in the schematic
list-of-graphic-components	graphical objects that represent components in the schematic; list of GeneralGraphicObject objects
list-of-graphic-connectors	graphical objects that represent connectors in the schematic; list of Connector objects
list-of-graphic-icons	graphical objects that contain schematic icons in the schematic; list of SchematicIconGraphicObject objects
list-of-visible-graphic-gauges	graphical objects that contain gauge icons in the schematic; list of GaugeIconGraphicObject objects
list-of-graphic-gauge-readings	graphical objects that contain the icons representing the qualitative state values displayed by a gauge in the schematic; list of GeneralGraphicObject objects
picture-handle	a handle to the picture displayed in the schematic
high-light-function	pointer to the function that can highlight a region in the schematic
low-light-function	pointer to the function that can lowlight a region in the schematic
button-event-function	pointer to the function that evaluates response to last operator action
list-of-active-regions	ActiveRegion objects
pick-region	last ActiveRegion selected by the operator

Table 4.10 Description of ***BoilerSchematic***

Object: *BoilerSchematic* **Instance of:** Schematic	
Instance Variables	**Value**
ccl-window	#<Object #280, "boiler-schematic", a *color-window*>
title	"boiler-schematic"
schematic-name	boiler-schematic
components-in-schematic	(ac-valve windbox tubes super-steam superheater stack feed-water-regulator furnace forced-draft-fan economizer drum desuper-steam desuperheater burner attemperator atmosphere air-register air-heater ad-drive air-damper)
fluid-paths-in-schematic	(control-air steam desuperheated-steam superheated steam flue-gas feed-water fuel-oil combustion-air)
subsystems-in-schematic	(control-air-subsystem steam-generation-subsystem combustion-subsystem)
list-of-graphic-components	(#<GeneralGraphicObject4002864> #<GeneralGraphicObject 4002656> ---------)
list-of-graphic-connectors	(#<connector 4005248> #<connector 4005180> -----)
list-of-graphic-icons	(#<SchematicIconGraphicObject 4006436> #<SchematicIconGraphicObject 4006636>--------)
list-of-visible-graphic-gauges	(#<GaugeIconGraphicObject 4005736> #<GaugeIconGraphicObject 4005936> ----)
list-of-graphic-gauge-readings	(#<GeneralGraphicObject 4005316> #<GeneralGraphicObject 4005516> -------)
picture-handle	#<Mac Handle, Unlocked, Size 2512 #x1DE5B8>
high-light-function	#<Compiled-function active-regions/default-high-light-fn>
low-light-function	#<Compiled-function active-regions/default-low-light-fn>
button-event-function	#<Compiled-function active-regions/button-event-fn>
list-of-active-regions	(#<ActiveRegion 4005392> #<ActiveRegion 4005532> -------)
pick-region	(#<ActiveRegion 4000416>)

Graphical Objects

Graphical objects are of several types. They are used to represent components and connectors in the schematic. Graphical objects are also used for icons in a schematic including those that represent gauges and gauge readings.

Components

Graphical objects that represent components in a schematic are instances of GeneralGraphicObject described in Table 4.11. Each instance of GeneralGraphicObject contains information that includes the component it represents, the schematic which it appears in, the connectors attached to it, and the text that is printed on it. The graphical object's location on the schematic, and the dimensions necessary to define its shape are also stored in its instance variables.

Connectors

Lines on the schematic that represent the physical connection between two components and show the direction of fluid flow through them are also graphical objects. They are called connectors. Connectors are made up of horizontal or vertical lines or a combination of both. The direction of fluid flow along the connectors is represented by an arrow-head. Lines with arrow-heads are called vectors. The ends of the lines that have the arrow-heads are known as leading edges of the connectors.

Connectors in the schematic are represented by instances of Connector described in Table 4.12. Each instance of Connector stores its color, the name of the schematic it is in, the names of the components at its trailing and leading edges and their graphical representations, and the name of the fluid flowing through it. The location, length and orientation of vectors and lines that make up the connector are also stored in its instance variables.

Table 4.11 Description of Object Class GeneralGraphicObject

Class: GeneralGraphicObject	
Class Variables	**Description**
parent-window	name of the schematic that contains the graphical object
parent-object	name of the object represented by the graphical object
list-of-input-connectors	Connector objects on its input side
list-of-output-connectors	Connector objects on its output side
printed-name	name printed on the graphical object
shape	shape of the graphical object
origin-x	x-coordinate of the point of origin
origin-y	y-coordinate of the point of origin
width	width of the graphical object
height	height of the graphical object
scale-value	a factor that can be used to enlarge or reduce the graphical object

Table 4.12 Description of Object Class Connector

Class: Connector	
Class Variables	**Description**
parent-window	schematic that contains the connector
connects-from	component at the trailing edge
connects-to	component at the leading edge
from-graphic-object	graphical object that represents the component at the trailing edge; an instance of GeneralGraphicObject
to-graphic-object	graphical object that represents the component at the leading edge; an instance of GeneralGraphicObject
fluid	fluid flowing through the connector
lines	list of length and location of lines that make up the connector
vectors	list of length location and orientation of lines that make up the connector
color	name of the default color for the connector

Icons

Icons in **Turbinia-Vyasa** are Macintosh resources. However, in a schematic they are represented by graphical objects. These graphical objects are also instances of GeneralGraphicObject. Graphical objects that represent schematic icons are instances of SchematicIconGraphicObject (See Table 4.13) which is a subclass of GeneralGraphicObject. Graphical objects that represent gauge icons are instances of another subclass of GeneralGraphicObject called the GaugeIconGraphicObject (See Table 4.14).

Instances of SchematicIconGraphicObject inherit all variables of GeneralGraphicObject but have an additional variable of their own. This variable, *inter-schematic-link*, stores information that establishes links between schematic icons in different schematics. Use of these links between schematic icons in schematics is required to provide smooth transition between schematics. Details of how the transition between schematics is accomplished is provided in the section under student-tutor interaction.

Instances of GaugeIconGraphicObject have two additional variables of their own. One variable, *associated-active-region*, stores an instance of ActiveRegion that must be added to or deleted from the list of active regions in a schematic whenever the gauge icon is displayed or removed from the schematic. The second variable, *associated-gauge-display-region*, stores the Region adjacent to the gauge icon that is used for displaying the gauge reading. A region object contains information about the location and dimensions of a rectangular region on a schematic.

Gauge readings are represented by five icons, one each for low, slightly low, normal, slightly high and high values. Similar to other icons in the schematic these too are represented by graphical objects. These graphical objects are instances of GeneralGraphicObject.

While graphical objects give visual representations to components, gauges, and schematic icons, they do not give users the capability to interact with these objects. Making these graphical objects active involves linking them with active regions. Active regions are instances of ActiveRegion that give the graphical objects the ability to respond to user interaction. Each manipulable graphical object is linked to an active region.

Table 4.13 Description of Object Class SchematicIconGraphicObject

Class: SchematicIconGraphicObject **Subclass of:** GeneralGraphicObject	
Class Variables	**Description**
inter-schematic-link	contains an instance of SchematicIconGraphicObject in another schematic that establishes the connection between two schematics

Table 4.14 Description of Object Class GaugeIconGraphicObject

Class: GaugeIconGraphicObject **Subclass of:** GeneralGraphicObject	
Class Variables	**Description**
associated-active-region	stores an instance of ActiveRegion that must be added or deleted from the schematic whenever the gauge icon plotted in the graphical object is displayed or removed from the schematic
associated-gauge-display-region	stores an instance of system defined Region where the reading of the gauge plotted in the graphical object is displayed

Active Regions

An instance of ActiveRegion has seven variables shown in Table 4.15. These variables store information about an active region such as the schematic window in which it appears, its dimensions on the screen, and a pointer to the graphical object linked to it. Procedures to execute when the mouse button is pressed or released with cursor inside it are also encapsulated within the object.

In an instance of Schematic (Table 4.9), active regions defined for the schematic are stored in the variable *list-of-active-regions*. This information is used by the schematic's button-event-function procedure to determine if a mouse click has occured in an active region defined for the schematic. In addition, an instance of Schematic also keeps a record of the last active region selected by the user. This information is retained in the variable *pick-region* and is updated every time a new active region is selected.

Responses to user interaction for active graphical objects in a schematic are derived from the properties of the object they represent. Components, gauges and icons are the three main types of objects represented by active graphical objects in a schematic. These objects are discussed next.

Table 4.15 Description of Object Class ActiveRegion

Class: ActiveRegion	
Class Variables	**Description**
parent-window	name of the schematic that contains the active region
region	an instance of Region that specifies the location of active region
button-down-function	pointer to a function that executes the response to mouse button down activity
button-up-function	pointer to a function that executes the response to mouse button up activity
high-light-function	pointer to a function that highlights the active region
low-light-function	pointer to a function that lowlights the active region
data	stores any context-specific information needed to execute the button-up and button-down functions

Components

All components are instances of classes of objects defined by the primitive class hierarchy (Figure 4.9). These components are instantiated prior to run time using a data file that contains structural, functional, behavioral information about components along with knowledge of failures in the component.

At the top of the primitive class hierarchy is an object class called Primitive. The nine class variables of Primitive are shown in Table 4.16. Values attached to these variables define the structure, function and behavior of the instantiated object. Each variable of Primitive is described in detail below.

Linked-graphic-objects

Linked-graphic-objects stores a list of paired associations. Each paired association consists of a schematic name and an instance of a graphical object that represents the primitive in that schematic. Thus, the number of paired associations in the list is the same as the number of representations of the primitive in various schematics.

General-structure

General-structure stores an instance of PrimitiveGeneralStructure (Table 4.17). The class variables of PrimitiveGeneralStructure store structural information that is relevant to all primitives. It stores the names of the subsystems, fluid path and schematics that contain the primitive, and the gauges attached to it on the input and output sides.

Specific-structure

Specific-structure stores an instance of PrimitiveSpecificStructure. The class PrimitiveSpecificStructure has two subclasses. Instances of primitives store one of the two subclasses of PrimitiveSpecificStructure in their *specific-structure* instance variable. Depending upon the subclass, the primitives are categorized as **simple** or **composite** primitives. Thus, SimplePrimitive and CompositePrimitive are subclasses of Primitive which

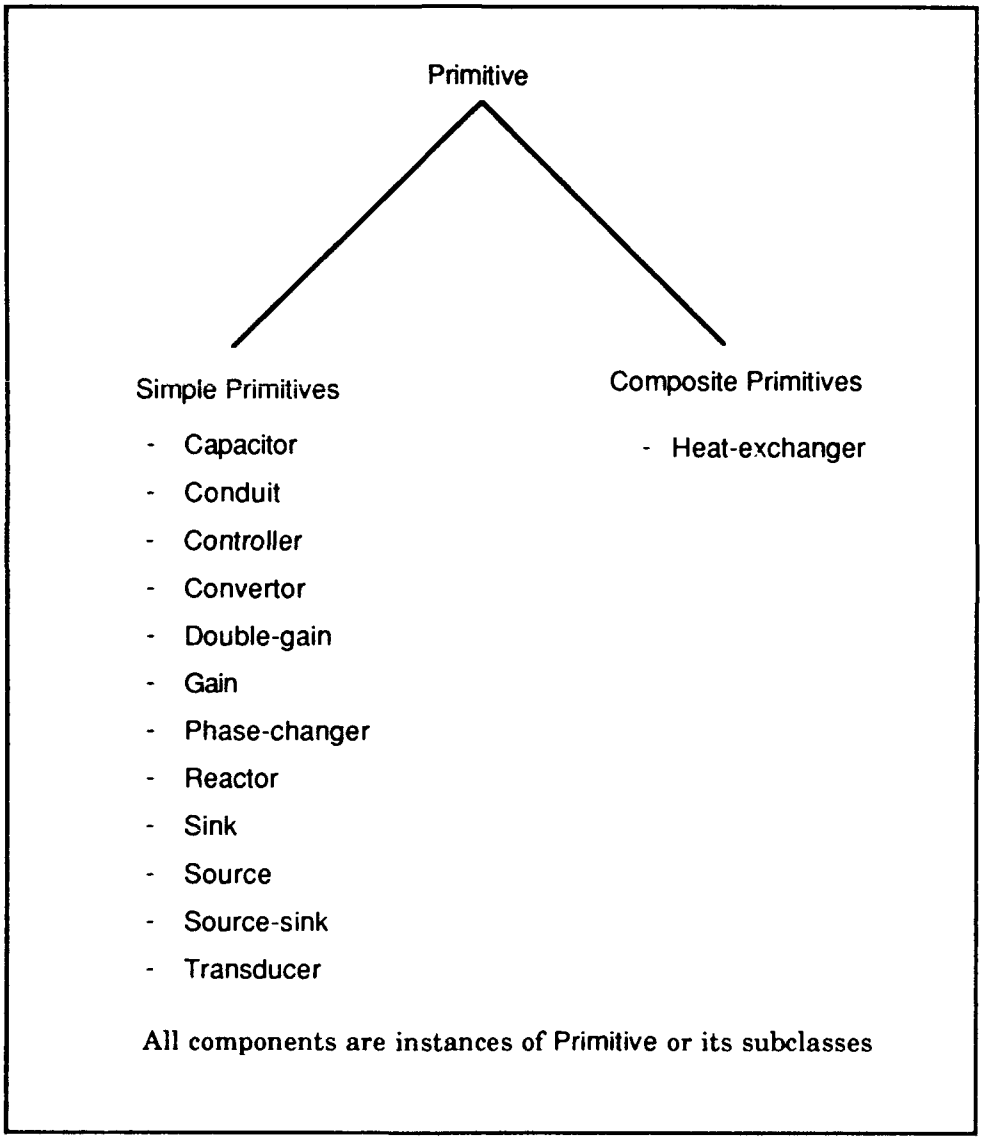


Figure 4.9 Primitive Class Hierarchy

Table 4.16 Description of Object Class Primitive

Class: Primitive All components are instances of Primitive or its subclasses	
Class Variables	**Description**
linked-graphic-objects	list of paired associations. Each pair contains the name of a schematic and an instance of GeneralGraphicObject that represents the primitive in the schematic
general-structure	contains an instance of PrimitiveGeneralStructure
specific-structure	contains an instance of PrimitiveSpecificStructure
function	describes the function of the primitive
propagation-behavior	pointer to an appropriate propagate-states method
update-states-behavior	pointer to an appropriate update-states method
normal-behavior	instance of NormalBehaviorProperties
abnormal-behavior	list of instances of AbnormalBehavior
failures	pointers to instances of SpecificFailureCase

Table 4.17 Description of Object Class PrimitiveGeneralStructure

Class: PrimitiveGeneralStructure	
Class Variables	**Description**
in-subsystem	subsystems that contain the primitive
in-fluid-path	fluid paths that contain the primitive
in-schematic	schematics that contain the primitive
input-gauge-list	list of instances of Gauge on the input side of the primitive
output-gauge-list	list of instances of Gauge on the output side of the primitive

have a common general structure but a different specific structure (Tables 4.18 and 4.20).

Simple primitives have a single fluid flowing in and out of them. They use SimplePrimitiveSpecificStructure, a subclass of PrimitiveSpecificStructure, to define their structure. Descriptions of five class variables of SimplePrimitiveSpecificStructure are provided in Table 4.19. Three of these variables define the structure of a simple primitive in terms of the names of the components on its input and output side, and the name of the fluid flowing through it. In addition, the state information is stored in variables *input-from* and *output-to* as lists of instances of type StateVariable. Each instance of StateVariable stores state information of a single connection on either side of the primitive.

Composite primitives have two fluids flowing through them. They use CompositePrimitiveSpecificStructure, the second subclass of PrimitiveSpecificStructure to define their structure. Descriptions of nine class variables of CompositePrimitiveSpecificStructure are provided in Table 4.21. As in the case of simple primitives, these variables define the structure of a composite primitive separately along the two fluid paths in terms of the names of the components on input and output sides, and the names of the fluids flowing through it. The state information for a composite primitive along the two fluid paths is stored in variables *input-from-fluid-1, input-from-fluid-2, output-to-fluid-1* and *output-to-fluid-2* as lists of instances of type StateVariable.

StateVariable class of objects store state information along each connection between components. They are used to describe the state information by both simple and composite primitives. A description of the six class variables used to store this information is provided in Table 4.22. For each connection, these variables store the name of a connected component, the weighted distribution of fluid by mass, and the three state values of pressure, temperature and flow or level. These state values change as the system states evolve with time during simulation. In addition, the variable *gauge* stores instances of Gauge as pointers to gauges that lie along the connection.

Table 4.18 Description of Object Class SimplePrimitive

Class: SimplePrimitive **Subclass of:** Primitive	
Class Variables	**Description**
specific-structure	an instance of SimplePrimitiveSpecificStructure
propagation-behavior	pointer to propagate-states method for simple primitives

Table 4.19 Description of Object Class SimplePrimitiveSpecificStructure

Class: SimplePrimitiveSpecificStructure **Subclass of:** PrimitiveSpecificStructure	
Class Variables	**Description**
input-components	components attached to the primitive on the input side
output-components	components attached to the primitive on the output side
input-from	list of instances of StateVariable. Each instance stores state information concerning a single input connection of the primitive
output-to	list of instances of StateVariable. Each instance stores state information concerning a single output connection of the primitive
fluid	fluid in input and output connectors of the primitive

Table 4.20 Description of Object Class CompositePrimitive

Class: CompositePrimitive **Subclass of:** Primitive	
Class Variables	**Description**
specific-structure	an instance of CompositePrimitiveSpecificStructure
propagation-behavior	pointer to propagate-states method for composite primitives

Table 4.21 Description of Object Class CompositePrimitiveSpecificStructure

Class: CompositePrimitiveSpecificStructure **Subclass of:** PrimitiveSpecificStructure	
Class Variables	**Description**
input-components-fluid-1	components attached to the primitive on the input side along fluid-1
input-components-fluid-2	components attached to the primitive on the input side along fluid-2
output-components-fluid-1	components attached to the primitive on the output side along fluid-1
output-components-fluid-2	components attached to the primitive on the output side along fluid-2
input-from-fluid-1	list of instances of StateVariable
input-from-fluid-2	list of instances of StateVariable
output-to-fluid-1	list of instances of StateVariable
output-to-fluid-2	list of instances of StateVariable
fluid-1	fluid-1
fluid-2	fluid-2

Table 4.22 Description of Object Class StateVariable

Class: StateVariable	
Class Variables	**Description**
connected-component	name of connected component
flow-or-level	flow or level (state value)
pressure	pressure (state value)
temperature	temperature (state value)
weight	weighted distribution of fluid by mass along the connection
gauges	list of instances of Gauge along the connection

Function

Function contains a description of the role played by the primitive in achieving the overall system goal. This information is extracted from a data file and stored as text.

Propagation-behavior

Propagation-behavior stores a pointer to an appropriate **propagate-states** method. Simple and composite primitives have a different propagate-states method that are used for simulating state evolution in **Turbinia**.

Normal-behavior

Normal-behavior stores an instance of a subclass of NormalBehaviorProperties (Table 4.23). NormalBehaviorProperties has several subclasses and depending upon the subclass the simple and composite primitives are further categorized as **capacitor, conduit, controller, transducer, gain, phase-changer, sink, source, source-sink, reactor, double-gain, converter** or **heat-exchanger**. Apart from heat-exchanger which is a composite primitive, the rest are simple primitives. Thus, Capacitor, Conduit, Controller, Transducer, Gain, PhaseChanger, Sink, Source, SourceSink, Reactor, DoubleGain, and Converter are subclasses of SimplePrimitive, and HeatExchanger is a subclass of CompositePrimitive. Some subclasses of NormalBehaviorProperties used for simple and composite primitives are described in Tables 4.24 through 4.27.

Update-states-behavior

Update-states-behavior stores a pointer to an appropriate **update-states** method. Each subclass of SimplePrimitive and CompositePrimitive is associated with a different update-states method that is used to compute the state value during each iteration of the simulation.

Table 4.23 Description of Object Class NormalBehaviorProperties

Class: NormalBehaviorProperties	
Class Variables	**Description**
state-type	stores the state type, i.e., pressure, temperature or flow which gets affected by the primitive

Table 4.24

(a) Description of Object Class Capacitor

Class: Capacitor **Subclass of:** SimplePrimitive	
Class Variables	**Description**
normal-behavior	contains an instance of a subclass of CapacitorBehaviorProperties
update-states-behavior	pointer to update-states method for capacitors

(b) Description of Object Class CapacitorBehaviorProperties

Class: CapacitorBehaviorproperties **Subclass of:** NormalBehaviorProperties	
Class Variables	**Description**
capacitance	value of capacitance

Table 4.25

(a) Description of Object Class Controller

Class: Controller **Subclass of:** SimplePrimitive	
Class Variables	**Description**
normal-behavior	contains an instance of a subclass of ControllerBehaviorProperties
update-states-behavior	pointer to update-states method for controllers

(b) Description of Object Class ControllerBehaviorProperties

Class: ControllerBehaviorProperties **Subclass of:** NormalBehaviorProperties	
Class Variables	**Description**
gain-value	value of gain

Table 4.26

(a) Description of Object Class Reactor

Class: Reactor **Subclass of:** SimplePrimitive	
Class Variables	**Description**
normal-behavior	contains an instance of a subclass of ReactorBehaviorProperties
update-states-behavior	pointer to update-states method for reactors

(b) Description of Object Class ReactorBehaviorProperties

Class: ReactorBehaviorProperties **Subclass of:** NormalBehaviorProperties	
Class Variables	**Description**
specific-energy	value of specific energy

Table 4.27

(a) Description of Object Class HeatExchanger

Class: HeatExchanger **Subclass of:** CompositePrimitive	
Class Variables	**Description**
normal-behavior	contains an instance of a subclass of HeatExchangerBehaviorProperties
update-states-behavior	pointer to update-states method for heat exchangers

(b) Description of Object Class HeatExchangerBehaviorProperties

Class: HeatExchangerBehaviorProperties **Subclass of:** NormalBehaviorProperties	
Class Variables	**Description**
heat-ratio	value of heat ratio

Abnormal-behavior

Abnormal-behavior stores knowledge of component's behavior under specific failure situations. This knowledge is stored in instances of AbnormalBehavior (Table 4.28 a and b). Each of these instances store the information concerning a single failure such as symptom, cause, names of affected components and parameter values needed by **Turbinia** to simulate abnormal system behavior. The number of such instances stored for each component is the same as the number of different failures concerning the component known to the instructional system. When a new failure concerning a component is to be added, a new instance of AbnormalBehavior is created with data encapsulated to simulate the system behavior associated with the failure.

Failures

Failures stores knowledge of specific cases of failure in the component known to the instructional system. This knowledge is stored in instances of SpecificFailureCase discussed later. **Vyasa** uses this knowledge to perform pedagogical functions. When new cases of failure are to be incorporated in the instructional system, a new instance of SpecificFailureCase is created and added to the list of failures for the component.

Table 4.28

(a) Description of Object Class AbnormalBehavior

Class: AbnormalBehavior	
Class Variables	**Description**
failure-number	a number given to failure. Each failure is identified by this number
initial-condition	description of the initial conditions such as speed of ship just prior to failure
initial-symptom	symptoms initially observed
cause	name of the failed component and the mode of failure
affected-components	list of paired associations. Each pair consists of a name of affected component and an instance of FailureEffect

(b) Description of Object Class FailureEffect

Class: FailureEffect	
Class Variables	**Description**
affected-fluid-path	name of the affected fluid path
input-or-output	input or output depending upon whether input or output states are affected
component	name of the component directly affected by failure
state-type	the state type affected by the failure
abnormal-state-parameter	affected state value
delay-parameter	number of simulation iterations before the component is affected

Gauges

The three types of gauges in **Turbinia**: pressure, temperature and level, are instances of Gauge. The seven variables of Gauge are described in Table 4.29. Three of these variables store information related to a gauge such as its type, number, and reading. The names of components on either side of the gauge, and the resource ID of the gauge-icon that represents it are also stored in the instance variables of a gauge.

The variable *linked-graphic-objects* stores a list of paired associations to link the gauge with the graphical objects that contain the icon representing the gauge in various schematics. Each paired association consists of a schematic name and an instance of a graphical object that contains the gauge icon in the paired schematic.

Icons

There are several icons used by **Turbinia-Vyasa**. Information concerning these icons such as their resource ID, the text that is printed on them, the graphical objects that represent these icons in schematics, and their response to user interaction is contained in objects of the type IconData (Table 4.30).

Representation of failure knowledge in **Vyasa** is discussed next. There are two types of representations used for failure knowledge: one to represent general abnormal system behavior associated with each failure mode and the other to represent specific failures.

Table 4.29 Description of Object Class Gauge

Class: Gauge	
Class Variables	**Description**
type	pressure, temperature, flow or level
gauge-number	gauge number
on-input-side-of-components	components that have the gauge on their input side
on-output-side-of-components	components that have the gauge on their output side
gauge-reading	gauge reading
gauge-icon-id	resource ID of icon that represents the gauge
linked-graphic-objects	list of paired associations. Each pair contains the name of a schematic and an instance of GaugeGraphicObject that represents the gauge in the schematic

Table 4.30 Description of Object Class IconData

Class: IconData	
Class Variables	**Description**
icon-id	resource ID of icon
printed-text	text printed on icon
linked-graphic-object	list of paired associations. Each pair contains the name of a schematic and an instance of GeneralGraphicObject that represents the icon in the schematic
icon-click-response	pointer to function to respond to a mouse click

Failure Modes

The system behavior associated with component failure modes is described in instances of FailureMode class of objects (Table 4.31). 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 represented in these objects includes information about the upstream and downstream system behavior along the gas and liquid paths, the name of the failure mode and the resource ID of the icon used to represent the failure mode.

Each of the two failure-mode variables, *gas* and *liquid*, contain an instance of ExpectedAbnormalBehavior (Table 4.32). This instance describes the expected abnormal behavior for the failure mode and the type of components that curtail the propagation of abnormal behavior along the affected fluid paths.

Specific Failures

Knowledge of the individual faults in the component is stored in objects of class SpecificFailureCase. Variables of SpecificFailureCase are described in Table 4.33.

Each instance of SpecificFailureCase contains information relevant to the failure such as the symptom, the name of the failed component and the mode of failure. The expected upstream and downstream abnormal system behavior due to the fault, the gauges that show this behavior, and the names of the components that curtail the propagation of this behavior away from the malfunctioning component are also stored within the object. In addition, the encapsulated knowledge of specific failures includes information concerning affected schematics, subsystems, fluid paths, and gauges along with explanations of cause-effect associations. The diagnostic tests that strengthen or weaken the likelihood of this specific failure are also stored along with the rest of the information.

FailureTwo shown in Table 4.34 is an instance of SpecificFailureCase. This failure concerns a blocked shut feed water regulator. A blocked shut feed water regulator causes the level in the boiler drum to fall, which is observed as the initial symptom and is the first indication of the existence of a problem. Since the blocked shut feed water regulator is in the liquid path, the feed water level upstream in the deaerating feed tank is expected to rise as effects of failure propagate away from the regulator. Similarly, with time, the level downstream in the boiler drum is expected to fall further. These expected abnormalities in

Table 4.31 Description of Object Class FailureMode

Class: FailureMode	
Class Variables	**Description**
name	name of failure mode
icon	resource ID of icon that represents the failure mode
gas	an instance of ExpectedAbnormalBehavior
liquid	an instance of ExpectedAbnormalBehavior

Table 4.32 Description of Object Class ExpectedAbnormalBehavior

Class: ExpectedAbnormalBehavior	
Class Variables	**Description**
upstream-behavior	list of affected state upstream and its qualitative value
downstream-behavior	list of affected state downstream and its qualitative value
upstream-behavior-limited-by	primitives upstream that curtail propagation of abnormal behavior
downstream-behavior-limited-by	primitives downstream that curtail propagation of abnormal behavior

Table 4.33 Description of Object Class SpecificFailureCase

Class: SpecificFailureCase	
Class Variables	**Description**
failure-number	a number given to failure. Each failure is identified by this number
symptom	symptoms initially observed
cause	name of the failed component
failure-mode	mode of failure
upstream-behavior	upstream affected state and its qualitative value
downstream-behavior	downstream affected state and its qualitative value
upstream-behavior-limited-by	component upstream that curtails propagation of abnormal behavior
downstream-behavior-limited-by	component downstream that curtails propagation of abnormal behavior
upstream-behavior-gauge	upstream gauge number and its qualitative value that shows the abnormal system behavior
downstream-behavior-gauge	downstream gauge number and its qualitative value that shows the abnormal system behavior
affected-subsystems	names of affected subsystems
affected-fluid-paths	names of affected fluid paths
affected-schematics	names of affected schematics
affected-components	names of affected components
affected-gauges	affected gauges along with cause-effect explanations
hypothesis-strengthening-tests	diagnostic tests that strengthen the belief in this failure
hypothesis-weakening-tests	diagnostic tests that strengthen the belief in this failure

Table 4.34 Description of ***FailureTwo***

Object: *FailureTwo* **Instance of:** SpecificFailureCase	
Instance Variables	**Value**
failure-number	2
symptom	"When speeding up the ship, boiler level drops low"
cause	"Feed-water regulator is stuck closed"
failure-mode	"blocked-shut"
upstream-behavior	((flow-or-level high))
downstream-behavior	((flow-or-level low))
upstream-behavior-limited-by	(deaerating-feed-tank)
downstream-behavior-limited-by	(deaerating-feed-tank)
upstream-behavior-gauge	(#<flow-or-level-gauge 3955028> high)
downstream-behavior-gauge	(#<flow-or-level-gauge 3939780> low)
affected-subsystems	(feed-water-preheating-subsystem steam-generation-subsystem combustion-subsystem power-generation-subsystem steam-condensation-subsystem)
affected-fluid-paths	(feed-water flue-gas combustion-air superheated-steam desuperheated-steam steam main-condenser-hot-fluid main-condenser-cold-fluid condensate)
affected-schematics	(steam-schematic boiler-schematic feed-water-schematic)
affected-components	(drum tubes economizer superheater)
affected-gauges	affected gauges along with cause-effect explanations is shown on the next page
hypothesis-strengthening-tests	((56 low) (58 high))
hypothesis-weakening-tests	((56 normal) (56 high) (58 normal) (58 low))

List of affected gauges along with cause-effect explanations for ***FailureTwo***:

((58 (slightly-high high) "The speed of the ship is increased by increasing the mass flow rate of steam to the turbines. When the steam demand from the boiler increases, the steam pressure in the drum decreases. This sets the boiler combustion control mechanism into operation. The job of the combustion control mechanism is to increase the quantity of combustion air and fuel to the boiler. Also, when the mass flow rate of steam from the boiler is increased, the boiler feed water control mechanism has to adjust the flow rate of feed water to maintain a mass balance of flow into and out of the boiler. When the feed water regualtor is stuck, the feed water control mechanism is unable to increase the feed water flow rate. Such a failure may be regarded as an example of a blocked shut valve. Because the feed-water-regulator does not permit an increase in the feed water flow rate, the upstream water level in the deaerating-feed-tank rises above the normal value.")

(56 (low) "The blocked-shut feed-water-regulator causes the water level in the steam drum, downstream, to fall below the normal level.") (7 (slightly-low) "The steam pressure in the boiler drum also decreases when the steam demand is increased. The steam pressure decreases further as the level of water in the drum continues to fall. The combustion control mechanism tries to increase the steam generation rate to build up the steam pressure. Even then the steam saturation pressure in the drum may remain below normal.")

(64 (slightly-high) "Since the water level in the drum continues to fall, the saturation pressure of steam in the drum keeps on decreasing. The combustion control mechanism tries to increase the steam generation rate to build up the steam pressure. Therefore, the fuel oil flows into the burner at a rate which is higher than the normal rate for this operating condition.")

(1 (slightly-high) "The combustion control mechanism is also responsible for pumping more airinto the burner to support combustion of excess fuel.")

(66 (slightly-low) "As the fuel flow rate is increased, the flow level in the settling tank falls below normal.")

(2 (slightly-high) "The higher than normal combustion air pressure propagates towards the burner.")

(3 (slightly-high) "The higher than normal combustion air pressure propagates past the burner.")

(46 (high) "Since the available heat energy is now used to heat less feed water, more heat is consumed in superheating steam at constant pressure in the superheater.")

(44 (slightly-high) "Since the available heat energy is now used to heat less feed-water in the economizer, the feed-water temperature at drum input is higher than normal")

(42 (slightly-high) "With less feed-water to heat in the economizer, the flue gas temperature at the air-heater outlet rises")

(48 (slightly-high) "Higher flue-gas temperature causes an increase in combustion-air temperature during preheating in the air-heater")

(45 (slightly-high high) "Higher superheated-steam temperature propagates to desuperheater as higher desuperheated-steam temperature")

(61 (fluctuating) "Level fluctuates in the deaerating-feed-tank--distillate-tank--atmos-drain-tank feed-back loop to compensate for level variations in the deaerating-feed-tank")

(67 (fluctuating) "Level fluctuates in the deaerating-feed-tank--distillate-tank--atmos-drain-tank feed-back loop to compensate for level variations in the deaerating-feed-tank")

(55 (slightly-high high) "The higher superheated-steam temperature also causes the steam temperature at lp-turbine exit to be higher than normal"))

system behavior can be observed on the level gauges mounted on the deaerating feed tank and the boiler drum. Over time, the effects of the failure can propagate to several subsystems, fluid paths, schematics and components. When this happens, more gauges begin to show abnormal readings. These readings, when observed, can strengthen a troubleshooter's belief that a blocked shut feed water regulator is in fact responsible for the abnormal system behavior. On the other hand, if the level gauges on the deaerating feed tank and the steam drum do not reflect the expected behavior, it should weaken a troubleshooter's belief that the feed water regulator has failed in a blocked shut mode. This detailed knowledge concerning the blocked shut feed water regulator is encapsulated in ***FailureTwo***.

Finally, in **Vyasa**, the knowledge concerning evaluation and rectification of student's misconceptions is represented as rules. These rules are organized in the instructional module of the tutor in several units called *knowledge sources*. There are knowledge sources for recognizing and understanding student actions, updating the student model, evaluating misconceptions and presenting instructions to rectify the misconceptions. The overall tutoring objective of **Vyasa** when operating in the active mode is achieved by the coordinated efforts of these individual knowledge sources. A blackboard-like control architecture which captures the dynamic evolution of tutor and student behavior is responsible for coordinating the efforts of these knowledge sources. A description of this control architecture is provided next.

Control Structure

Several blackboard models have been developed in the past to represent complex control structure of problem solving activity (Hayes-Roth, 1985, Cohen et al., 1982). Each of these models consists of a global data structure called the blackboard and a number of specialist programs. These programs have access to the information on the blackboard and are at liberty to utilize and refine the information as well as post new information on the blackboard. In all these models, control of processing information from the blackboard is asynchronous and opportunistic. In other words, the specialist programs do not post information in a particular order and they use information from the blackboard whenever the information appears useful to them.

Some applications of blackboard architecture use static input like HEARSAY-II speech understanding program of Hayes-Roth (1985). However, Nii (1982, 1986) in the HASP

system and Rubin et al. (1987) in OFMspert have used blackboards with continuous stream of dynamic data. In OFMspert, the blackboard has been used to interpret interaction data and infer operator intents.

Turbinia-Vyasa uses a blackboard-like architecture for high level control and planning of pedagogical functions. It consists of a blackboard object and several rule-based knowledge sources that can access information posted on the blackboard and make changes to it (Figure 4.10). The blackboard object is global data structure that contains information organized at three levels of abstraction: state of the instructional system, tutor behavior and student behavior. The knowledge sources are invoked when preconditions necessary to activate them are posted on the blackboard. Together, the blackboard and the knowledge sources play an important role in helping **Vyasa** evaluate and provide help to rectify student misconceptions. Each component of this control architecture is described in detail next.

Blackboard

The blackboard is an instance of BlackBoard. The class variables of BlackBoard are shown in Table 4.35. Most of the variables of BlackBoard store information concerning the state of the instructional system. The state of the instructional system is defined in terms of the tutor mode, the state of the simulation, the current displays, time spent by the student in the different modes of interaction and the pending events. The tutor can be in two modes: active or passive. The simulation can be in three states: running, halted or end-of the-session. The current displays are the schematics and tutor dialogs that are visible to the student. The time spent by the student in the different modes of interaction includes the time elapsed since the start of the simulation and the time since the student last invoked the passive tutor. The pending events are the pedagogical functions to be performed next based on the current state of information on the blackboard (See Table 4.36).

The blackboard also stores complete information related to the student's last action. This information is stored in an instance of StudentAction. Variables of StudentAction are described in Table 4.37. These variables contain information such as the type of action, the time at which it took place, the item selected, mouse location and the schematic or the dialog box in which the interaction occured. In addition, information concerning the system response to the student action is also recorded. After every new action, information concerning the previous action is transferred to an output file.

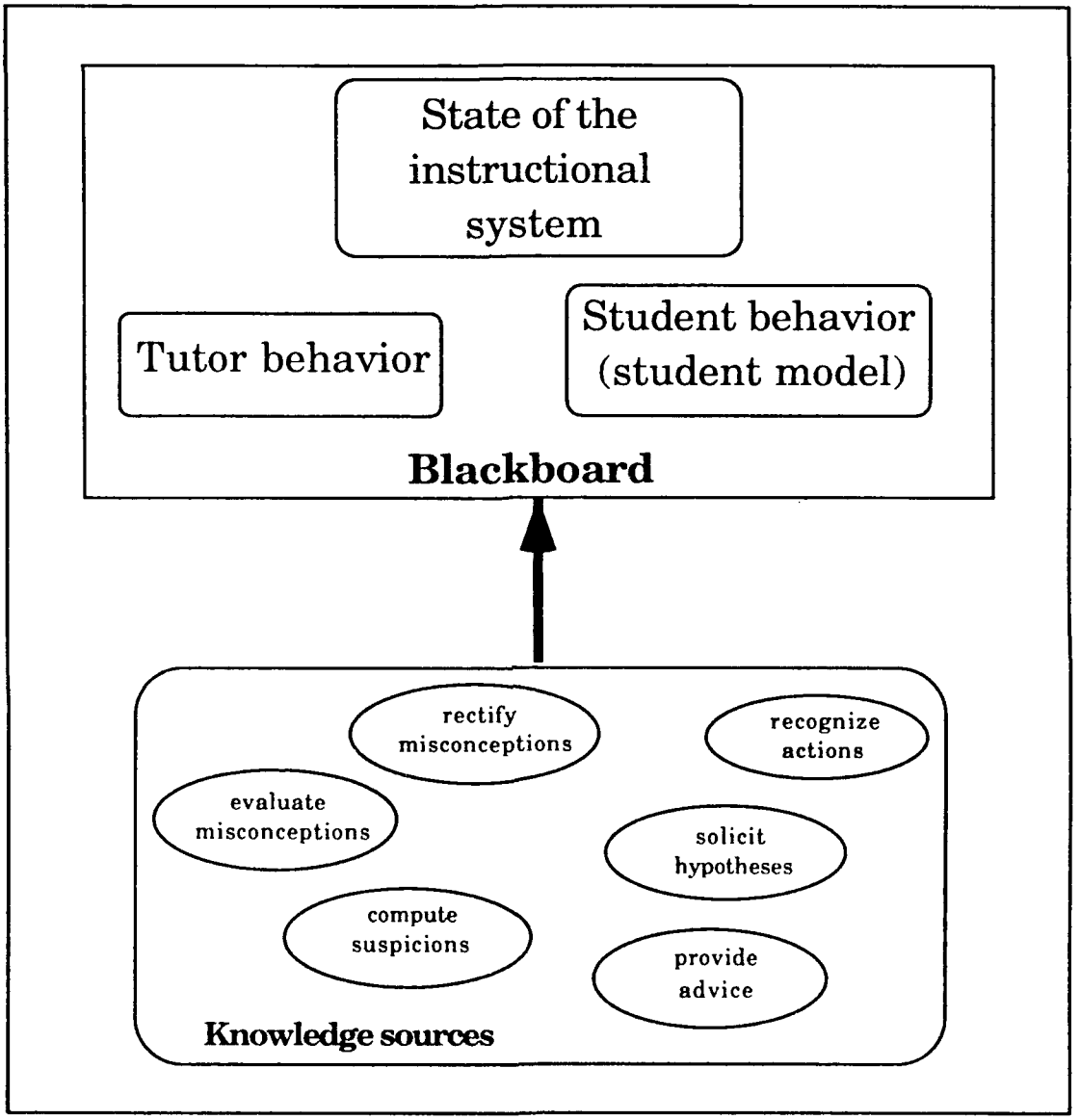


Figure 4.10 Control Architecture

Table 4.35 Description of Object Class BlackBoard

Class: BlackBoard	
Class Variables	**Description**
tutor-mode	current tutor mode: nil, passive, active
simulation-state	current state : running, halted or end-of-session
simulation-iteration-number	number of current iteration
simulation-start-time	clock time at which the simulation started
tutor-interaction-begin-time	If currently interacting with tutor, clock time at which current interaction with tutor started
total-tutor-interaction-time	total time spent by the student interacting with the tutor in the current session
active-schematic	schematic currently displayed
active-passive-tutor-dialogs	passive tutor dialogs currently displayed
state-of-student	one of: troubleshooting, requesting to diagnose, adding hypotheses on request, adding hypotheses voluntarily, deleting hypothesis or seeking advice
selected-subsystem	subsystem selected by student
selected-fluid-path	fluid path selected by student
selected-component	component selected by student
selected-mode-of-failure	failure mode selected by student
hypothesis-to-add	the hypothesis to be added to the list of hypotheses
hypothesis-to-delete	the hypothesis to be deleted from the list of hypotheses
hypothesis-to-advice	the hypothesis for which advice is sought
pending-events	list of instances of PendingEvent
student-action	an instance of StudentAction
tutor-behavior	an instance of TutorBehavior
student-behavior	an instance of StudentBehavior

Table 4.36 Description of Object Class PendingEvent

Class: PendingEvent	
Class Variables	**Description**
type	pointer to the next pedagogical function
time	time at which the event must take place
message	context-specific message for the pedagogical function

Table 4.37 Description of Object Class StudentAction

Class: StudentAction	
Class Variables	**Description**
type	one of the nine types of valid actions
schematic	schematic where the action was taken
tutor-dialog	tutor dialog where the action was taken
error-dialog	error dialog where the action was taken
time	time at which the action was taken
simulation-iteration-number	the iteration number at which the action was taken
mouse-location	location of the cursor when mouse was clicked
selected-item	name of the item selected
data	any action or context specific data that may be needed, e.g., gauge reading when the selected item is a gauge

The blackboard may, however, retain some information related to previous student actions. This is done because the context of new actions is often established from previous actions. For example, while interacting with the passive tutor, there are queries that a student poses in two steps which require the tutor to remember the information conveyed in the first step until the second step is taken. Inquiring about a subsystem's function is one such query. The formulation of this query involves selecting the subsystem name from an interactive dialog and then choosing the query concerning the subsystem's function from another interactive dialog. In order to correctly respond to all such two step queries, the tutor must remember the information provided by the student in the first step.

Furthermore, the interactive interface of passive tutor is designed to give the student the flexibility to ask multiple questions with reference to a single context. That is, multiple questions can be asked concerning the same subsystem without having to re-select the subsystem. This, however, means that the tutor must remember the selected subsystem to answer all subsequent queries and must do so till the student decides to change the context.

Since the tutor is responsible for guiding the student to formulate queries, the knowledge of previous actions that must be retained to understand the complete query is known to the tutor. This information essential to comprehend future actions is stored on the blackboard as long as it is needed. Thus, for the two step query concerning subsystem function, the name of the selected subsystem is retained in an instance variable of the blackboard till the student has no further queries about the selected subsystem. There are numerous such instance variables of the blackboard that are used to provide proper context to subsequent student actions.

In addition, the blackboard captures the dynamic evolution of the tutor and student behavior. This information is stored in instances of TutorBehavior and StudentBehavior shown in Tables 4.38 and 4.39. While most of the student behavior evolves dynamically, some of the tutor behavior is derived from the tutor's knowledge of the failures.

Table 4.38 Description of Object Class TutorBehavior

Class: TutorBehavior	
Class Variables	**Description**
failure-diagnosed	true or false; true when student is successful
current-failure	pointer to an instance of SpecificFailureCase that contains information related to current failure
structural-misconceptions-rectified	list of structural misconceptions rectified; i.e., all unaffected schematics investigated
functional-misconceptions-rectified	list of functional misconceptions rectified; i.e., unaffected subsystems and fluid paths investigated
behavioral-misconceptions-rectified	list of behavioral misconceptions rectified; i.e., hypotheses and the diagnostic tests used as evidence against the hypotheses
hypotheses-aided	hypotheses on which help was sought and the advice given
investigative-actions-since-lsm	number of investigative actions since structural misconception was identified
investigative-actions-since-lfm	number of investigative actions since functional misconception was identified
call-for-schematic-actions-since-lsm	number of call for schematic actions since structural misconception was identified
summary-of-hypotheses-refinement	hypotheses provided by the student in the order conveyed.
current-hypotheses	current student hypotheses. Each hypothesis consists of a component name and the suspected mode of failure
most-suspected-subsystem	name of most suspected subsystem
most-suspected-fluid-path	name of the most suspected fluid path
current-misconceptions	current hypotheses being pursued inspite of evidence available to reject them
evidence-against-current-hypotheses	all possible diagnostic tests that serve as evidence against current hypotheses

Table 4.39 Description of Object Class StudentBehavior

Class: StudentBehavior	
Class Variables	**Description**
schematics-viewed	schematics in the order viewed
subsystems-investigated	subsystems in the order investigated
fluid-paths-investigated	fluid paths in the order investigated
gauges-investigated	gauge numbers investigated and gauge value observed at the time of investigation
call-for-schematic-actions	all call for schematic actions
investigative-actions	all investigative actions
informative-actions	all informative actions
diagnose-request-action	all diagnose request actions
diagnose-actions	all diagnose actions
current-hypotheses	current student hypotheses. Each hypothesis consists of a component name and the suspected mode of failure

Tutor Behavior

During any training session, an instance of TutorBehavior consists of information obtained either from the tutor's knowledge of the failures or from the knowledge sources in the instructional module.

The information obtained from tutor's knowledge of the failures remains unchanged for the problem solving session. This information concerns current failure and is used by the tutor to evaluate student performance. It includes the mode of failure, affected subsystems, fluid paths, schematics, components and gauges, and the upstream and downstream abnormal behavior associated with the failure.

On the other hand, the information posted by the knowledge sources is dynamic and concerns the student. It includes information solicited by the knowledge sources from the student such as the student's initial hypotheses and current hypotheses. It also includes summary of hypotheses refinement, evidence against current hypotheses, and the most suspected subsystem and fluid path as inferred from student actions.

In addition, the existing misconceptions of the student and those rectified during the current session are posted by knowledge sources that evaluate and rectify misconceptions. A record of the actions taken since the various types of misconceptions were last identified is also maintained in the instance of TutorBehavior.

Student Behavior

Additional information concerning actions taken by the student is stored in an instance of StudentBehavior which is updated dynamically by knowledge sources after every student action. This information consists of the schematics, subsystems, fluid paths, components and gauges explored or investigated by the student and the order in which it happened along with a detailed description of each student action. The detailed information includes the type of the action and where and when it took place.

Knowledge Sources

There are several knowledge sources that use the blackboard as a globally shared database and often compete with each other to modify information on the blackboard. These

knowledge sources that also determine and execute the appropriate pedagogical functions of the tutor based on the current status of the instructional system are described below.

Knowledge sources for recognizing actions

When a student takes an action, nine knowledge sources, each capable of recognizing one of the nine types of valid interactions described earlier, compete with each other to recognize the new action. The knowledge source that succeeds in recognizing the action updates the student behavior on the blackboard by posting information relevant to the new action. A complete description of the knowledge contained in these knowledge sources was provided under "Knowledge of Student Actions" (page 67).

Knowledge sources for computing suspicions

After the new action is posted on the blackboard, there are knowledge sources that compute the subsystem and fluid path most suspected by the student. This information concerning the most suspected subsystem and fluid path is then updated on the blackboard. A complete description of the knowledge contained in these knowledge sources and the method of computing the most suspected subsystem and fluid path was given under "Knowledge to Update the Student Model" (pages 67- 68).

Knowledge sources for evaluating misconceptions

There are three knowledge sources, each capable of evaluating one of the three types of misconceptions known to Vyasa. These knowledge sources analyze the new information posted on the blackboard after each student action and compete with each other to identify misconceptions. If a misconception is identified, it is posted on the blackboard. A complete description of the knowledge contained in these knowledge sources was provided under "Knowledge to Evaluate Misconceptions" (pages 68-69).

Knowledge sources for rectifying misconceptions

After the identified misconception is posted on the blackboard, knowledge sources that rectify misconceptions compete with each other until the one that can rectify the posted misconception is activated. This knowledge source delivers the relevant instructional material and keeps a record of its activities posted on the blackboard to be used to structure

new instructions. A complete description of the knowledge contained in these knowledge sources was provided under "Instructional Knowledge" (pages 69-74).

Knowledge source for soliciting hypotheses

From time to time, based on the investigative actions conducted by the student, this knowledge source solicits information concerning failure hypotheses that the student is pursuing. After soliciting this information, this knowledge source updates the student's current hypotheses on the blackboard.

Knowledge sources for aiding hypothesis

In addition to the knowledge sources described above, there are two knowledge sources that help the students with their failure hypothesis. One uses tutor's knowledge of specific cases of failure and the other uses tutor's knowledge of failure modes.

The first knowledge source is invoked when help is sought for failure hypothesis that matches one of the specific cases of failure known to Vyasa. For such hypothesized failures, knowledge concerning affected gauges is available to the tutor. This knowledge is used to determine whether the hypothesized failure is probable based on the observed symptoms. If so, the tutor suggests more tests to strengthen the student's belief in the hypothesis. If not, the tutor suggests tests to weaken the student's belief in the hypothesis. Also, since the knowledge source has access to the information displayed on the blackboard, it can determine if evidence has already been gathered to eliminate suspicion from the hypothesis. If so, the student is advised to drop the hypothesis from the list of suspected components.

The second knowledge source is invoked when the knowledge of the hypothesized failure is unavailable to the tutor. Instead, knowledge of the general modes of failure is used by the knowledge source to generate advice. For example, if the hypothesized failure involves blocked shut mode of failure, the knowledge source first determines the fluid path through the suspected component. Then, using its knowledge of modes of failure it determines the expected abnormal behavior associated with the fault. Next, using the knowledge of available gauges, the knowledge source determines if the expected abnormal behavior is observable under the current situation. If so, it suggests checking the relevant gauges to strengthen or weaken the student's belief in the hypothesis.

This concludes the section on the blackboard architecture used by **Vyasa** to perform its functions. Apart from this control architecture, organization and representation of knowledge in Turbinia-Vyasa were also described in this chapter. However, knowledge organization that captures system structure, function, and behavior under failed and normal states, pedagogical knowledge that concerns evaluation and rectification of misconceptions, instructional knowledge that concerns content, form and time of presentation of instructions, and a control architecture for planning the pedagogical functions of the tutor are insufficient for the success of a tutoring system. Properly designed interactive interfaces also play a major role in imparting knowledge about the system and its operation during normal and abnormal situations. In the next chapter, the interface of **Turbinia-Vyasa** and details of student interaction with the instructional system are described.

Summary

In this chapter, **Turbinia-Vyasa**, an implementation of the ITS architecture proposed in Chapter III was described. First, the marine power plant domain of the instructional system was described. The discussion of the domain included a description of the various subsystems in the power plant and the role they play in accomplishing the goal of power production. Components of the automatic boiler control system that regulate the operating conditions of the control devices were also described. In addition, salient features of the troubleshooting task in this domain and the educational background of the student trainees were discussed. Next, organization of knowledge in **Turbinia-Vyasa** was described. The discussion of knowledge organization included a description of the various components of knowledge represented in the instructional system. This was followed by a discussion of implementation details that focused on methods of knowledge representation employed in **Turbinia-Vyasa**. Finally, a control architecture was described that is used by the instructional system to plan the pedagogical functions of the tutor.

CHAPTER V

STUDENT-TUTOR INTERFACE OF TURBINIA-VYASA

The student-tutor interface of **Turbinia-Vyasa** and the valid forms of interactions at this interface are described in this chapter. A brief description of the various elements of the interface is provided, followed by a discussion of the operator interactions with the simulator and the tutor in both the passive and active modes.

The student-tutor interface of **Turbinia-Vyasa** has been developed on a dual screen Apple Macintosh II workstation. The dual screen configuration consists of one 19" color monitor and a 13" color monitor. In this set up, the larger monitor is the left screen and the smaller monitor is the right screen. A single button computer mouse that can point to all locations on both screens is the only input device. The mouse is used to interact with the direct manipulation interface to both **Turbinia** and **Vyasa**. All actions at the interface involve moving the mouse cursor to a desired location and clicking on the mouse button. In all cases, a single click is adequate. All valid user actions have appropriate response while invalid actions are ignored by the system.

The Interface

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

The seven *schematics* display the physical connections between the components of the power plant. They are used to investigate components and probe gauges attached to these components.

The *schematic menu* displays seven icons each representing one of the seven schematics. These icons are used to display the schematics on the left screen.

The *requests menu* has three icons. The first icon is used to request for an opportunity to diagnose the fault, the second to temporarily halt the simulation and the third to resume the simulation. When working without **Vyasa**, the stop and resume icons are disabled.

The *clock* displays the time left to troubleshoot the current problem. This clock is updated every minute.

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

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

The *hypothesis menu* has four items. Students use the hypothesis menu to communicate information concerning their failure hypotheses to **Vyasa**. The first item "View" is used to review the failure hypotheses that the student has provided to the tutor. "Add" and "Delete" are used to modify hypotheses. "Advice" provides assistance from the tutor.

In the next section, student interaction with **Turbinia** and **Vyasa** is described. Each element of the interface is also discussed in more detail.

Interaction with Turbinia-Vyasa

At the beginning of every training session, the dual screen Apple Macintosh II workstation displays three menus and a clock on the large screen and a dialog box on the small screen (Figure 5.1). The three menus on the large screen are the *schematic menu*, the *requests*

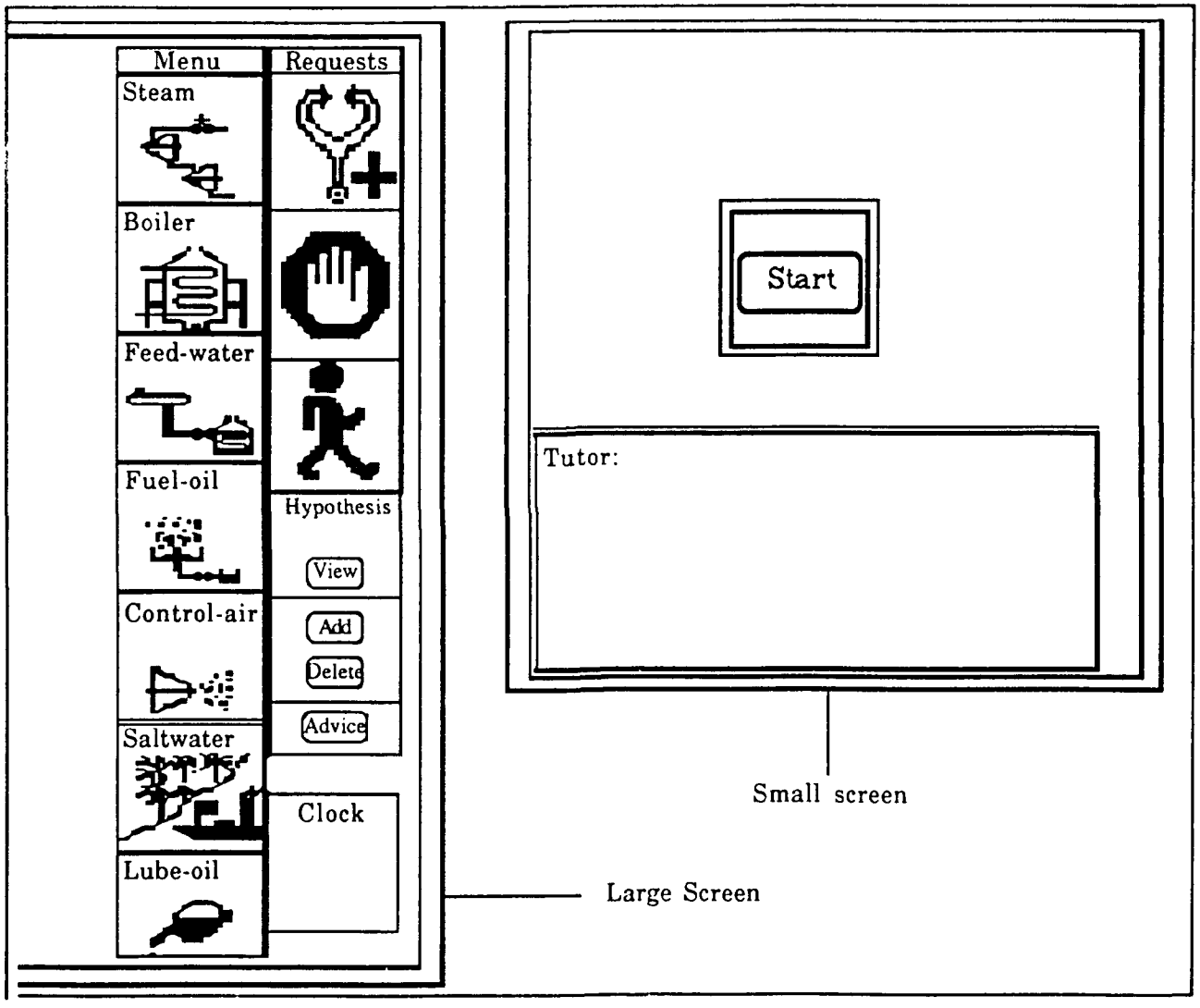


Figure 5.1 Configuration of Screens

menu and the *hypothesis menu*. A *tutor dialog* is displayed on the bottom edge of the small screen. For sessions where the active mode of the tutor is not invoked, the *hypothesis menu* under the *requests menu* is not displayed.

Schematic menu

The schematic menu on the large screen and also shown in Figure 5.2, displays seven icons. Each icon represents one of the seven schematics of the simulated power plant. The names of the seven schematics are also provided in textual form above each icon. The seven schematics are the steam, boiler, feed water, fuel oil, control air, saltwater and lube oil. Any of the seven schematics can be accessed by the student by clicking on the icon representing the schematic.

Requests menu

The requests menu adjacent to the schematic menu displays three icons (Figure 5.3). The first is the diagnose icon. The student must click on the diagnose icon before identifying the failure. By clicking on the diagnose icon, the student indicates to the instructional system an intention to identify the component responsible for the observed abnormal behavior.

The second icon on the requests menu is the stop icon. A click on the stop icon can halt the simulation and put the student in a mode to interact with the passive tutor. In a session without the tutor, a click on the stop icon produces an error message.

The third icon on the requests menu is the resume icon. The resume icon is used to restart simulation after it has been halted to communicate with the passive tutor. In a session without the tutor or when the student is not interacting with the tutor, the resume icon is shown disabled. All disabled icons on the **Turbinia-Vyasa** interface have a yellow-brown colored background as compared to enabled icons that are shown in gray.

Clock

The time to diagnose a fault is limited. During a session, the clock below the hypothesis menu displays the time left to solve the current problem. This clock gets updated every

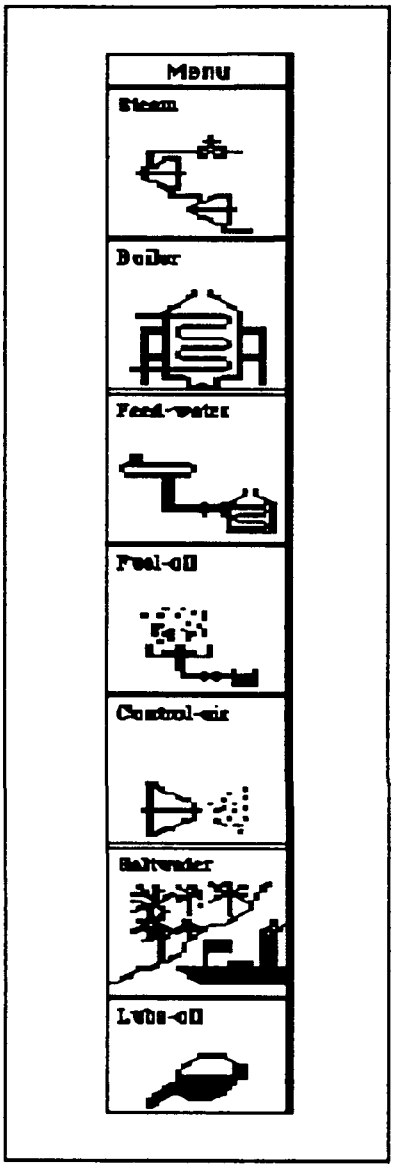


Figure 5.2 Schematic Menu

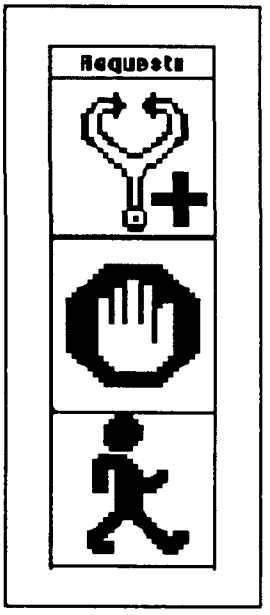


Figure 5.3 Requests Menu

minute. When the clock winds down to 0 and the fault is not yet diagnosed, the simulation stops and the student can no longer continue to diagnose.

Tutor dialog

The instructional system communicates with the student through textual messages and instructions presented on the tutor dialog. This tutor dialog is displayed on the bottom edge of the small monitor (Figure 5.4). All communications through this dialog box are accompanied by a beep.

Symptom dialog

The symptom dialog shows the simulated ship's initial operating condition and the first symptoms that indicate the existence of a problem (Figure 5.5). Troubleshooting for failure begins after the symptom dialog appears with a beep in the center of the large screen.

The rest of this chapter is divided into two major sections to describe student's interaction with the simulator and the tutor. The first section describes the interaction with **Turbinia** and the second describes the interaction with **Vyasa**. Most of the examples used to describe the interaction are related to a problem solving session that begins with the symptom dialog shown in Figure 5.5.

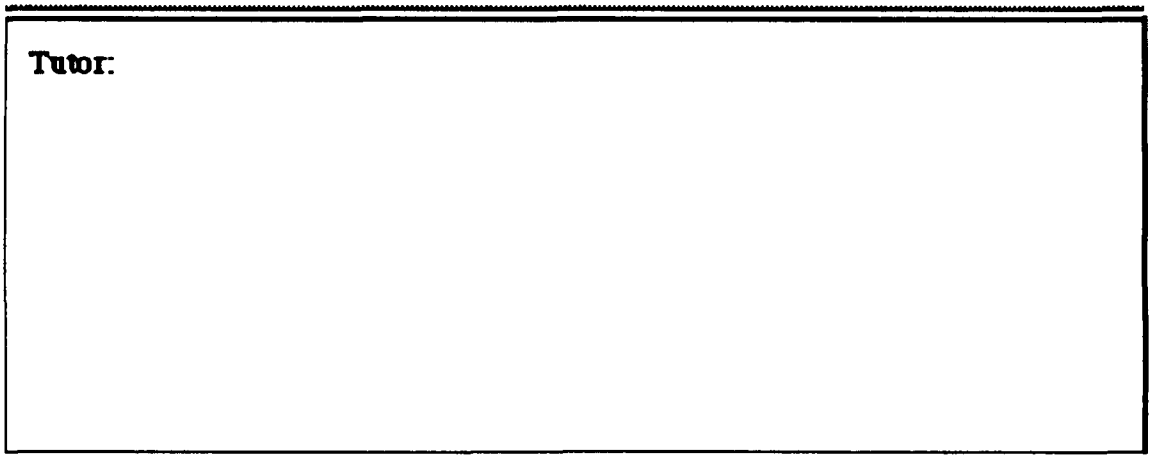


Figure 5.4 Tutor Dialog

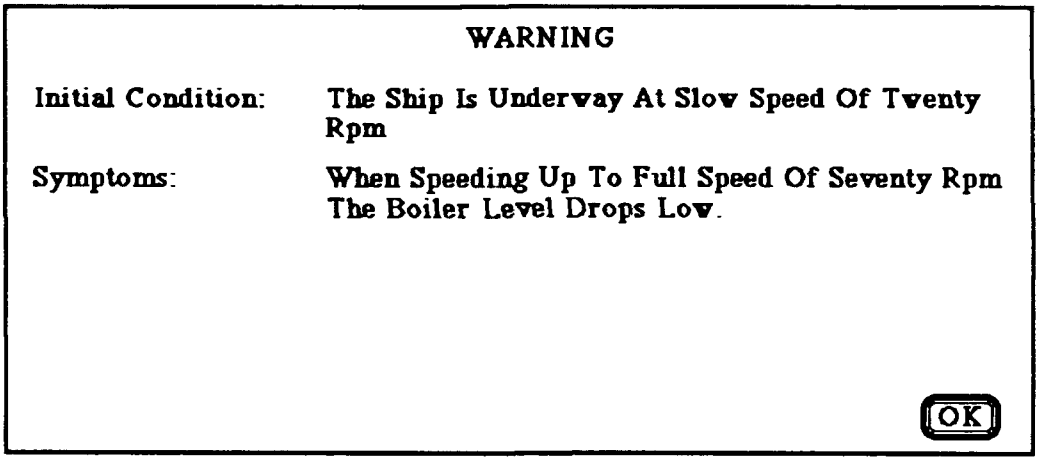


Figure 5.5 Symptom Dialog

Interaction with Turbinia

The student interacts with **Turbinia** through the seven schematics that display the physical connections between components of the power plant. The seven schematics are shown in Figures 5.6a, b, c, d, e, f, and g. These schematics can be accessed by clicking on the icons in the schematic menu. Each icon in the schematic menu represents a different schematic. The same schematic icon that is used to access the schematic also appears in the right top corner of the schematic it represents. The features of the schematic interface of **Turbinia** are discussed next, mainly with reference to boiler schematic shown in Figure 5.6b.

All the components in the schematics are represented by rectangles. The connections between components are shown by solid lines called connectors. The direction of fluid flow between components is shown by the arrow head on these connectors. For example, the economizer and the drum (shown as steam-drum in the boiler schematic) have a two way connection. The connection from the economizer to the drum represents the flow of feed water while the connection in the reverse direction represents the flow of flue gases.

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

If the student clicks on the feed water icon at the end of the input connector of feed water regulator two things happen. First, the display switches to feed water schematic. Second, the boiler icon on output connector from the hp heater is highlighted with a red band around it. The highlighted boiler icon helps establish the physical connection between the hp heater and the feed water regulator. The student can click on the highlighted boiler icon to get back to the boiler schematic. When this is done, the boiler schematic has two feed water icons highlighted, one connected to the feed water regulator and the other to the economizer. This simply means that the hp heater is connected to both the feed water regulator and the economizer in the boiler schematic.

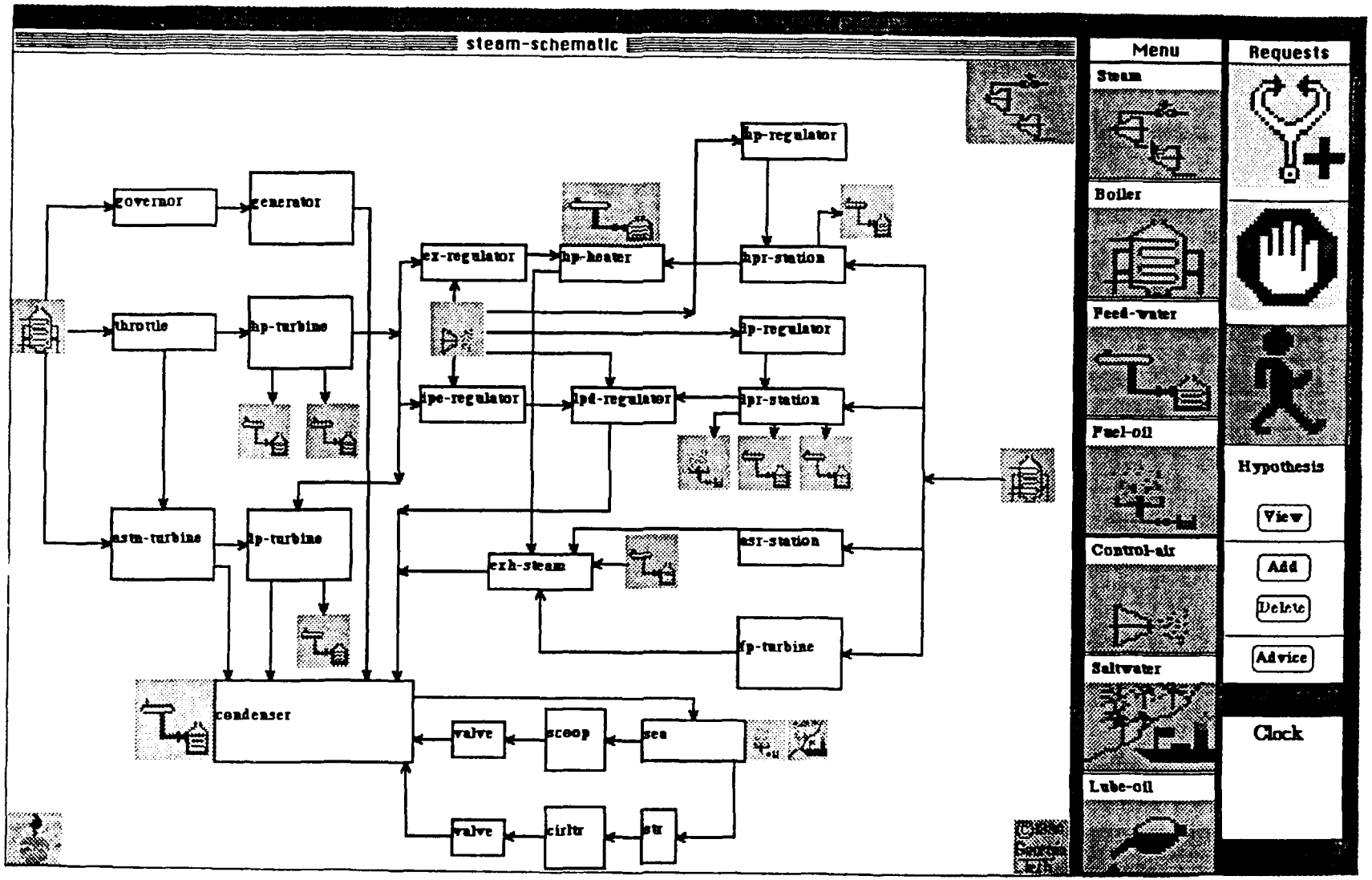


Figure 5.6a Steam Schematic

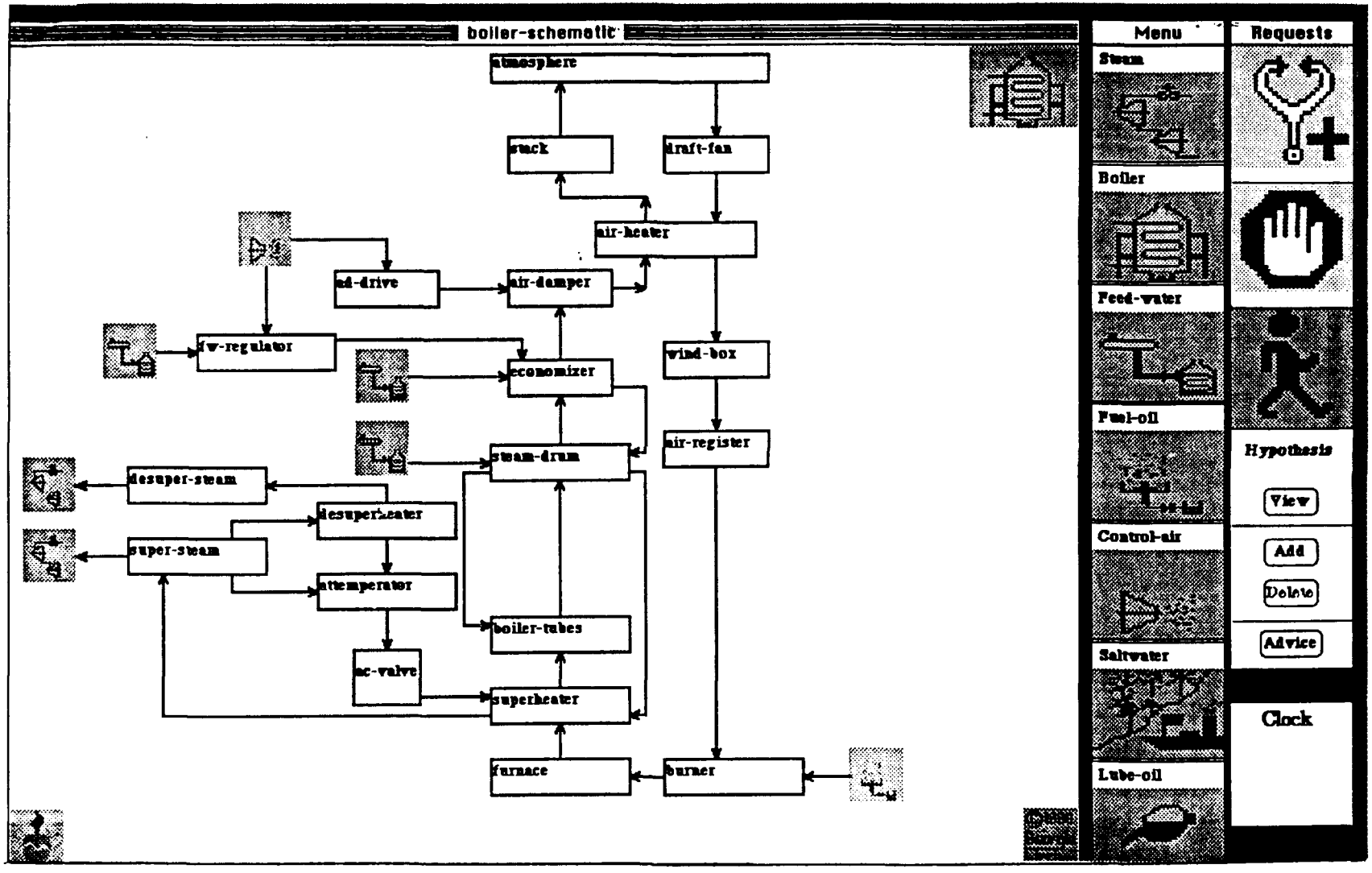


Figure 5.6b Boiler Schematic

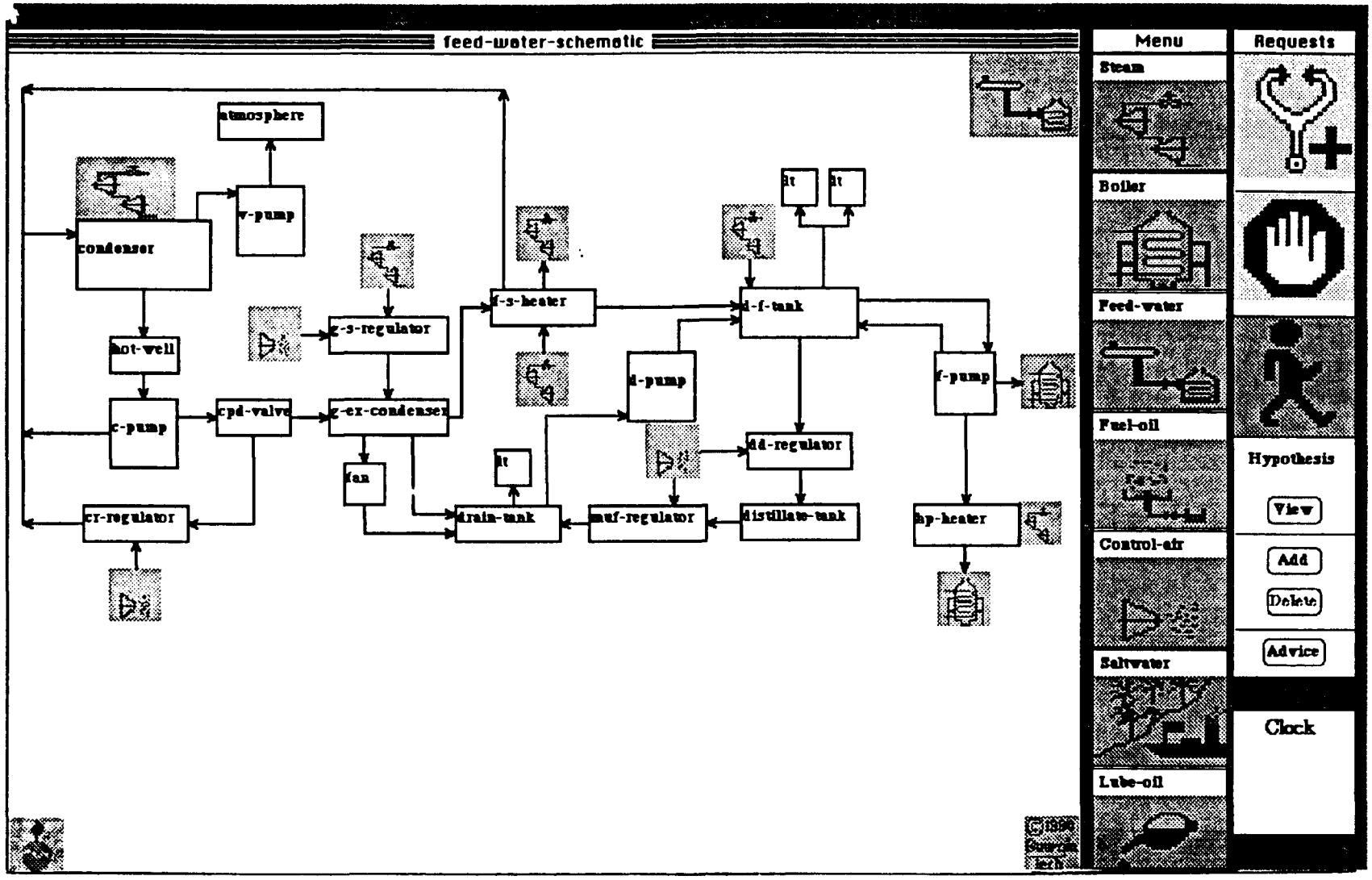


Figure 5.6c Feed Water Schematic

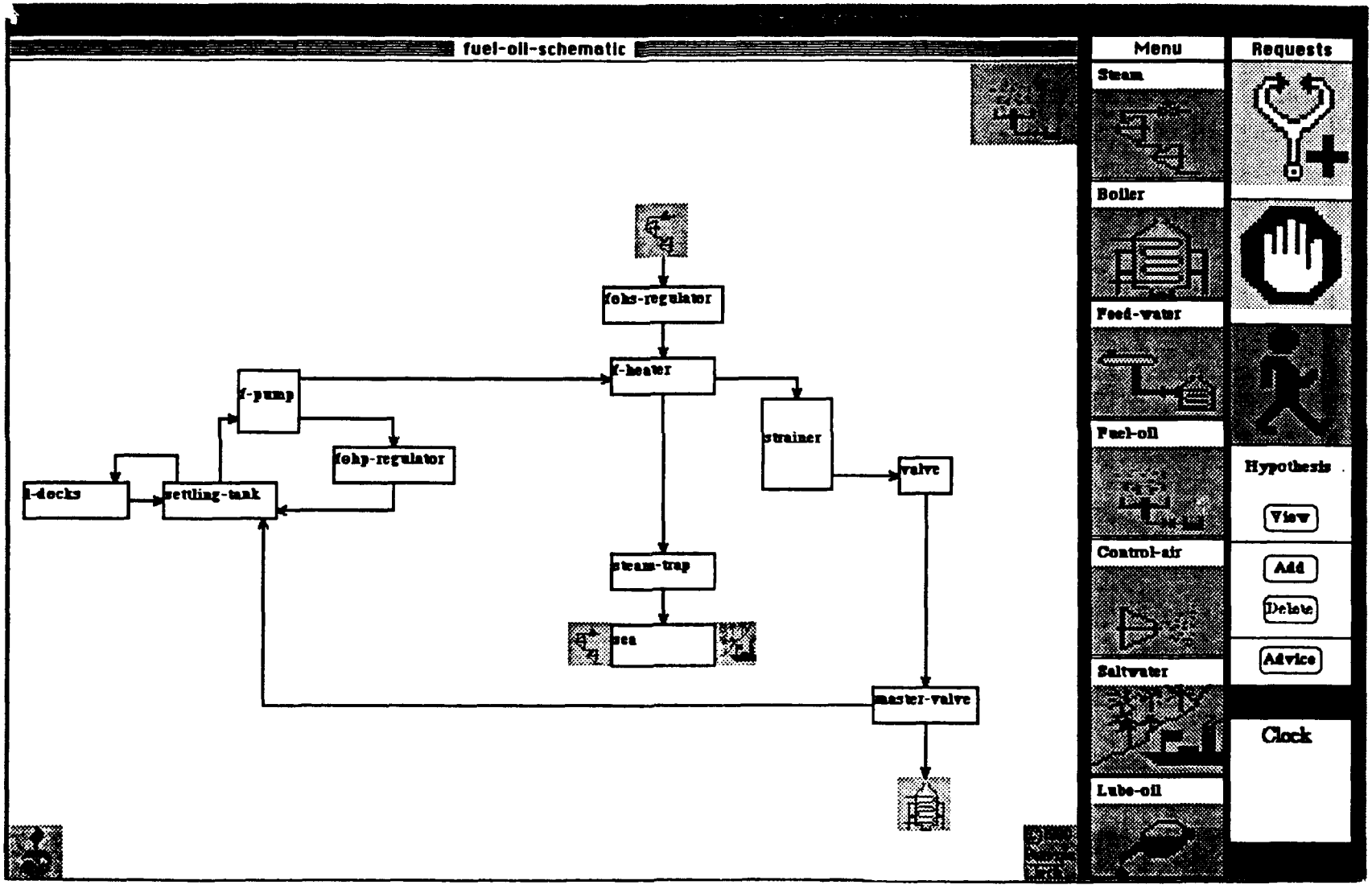


Figure 5.6d Fuel Oil Schematic

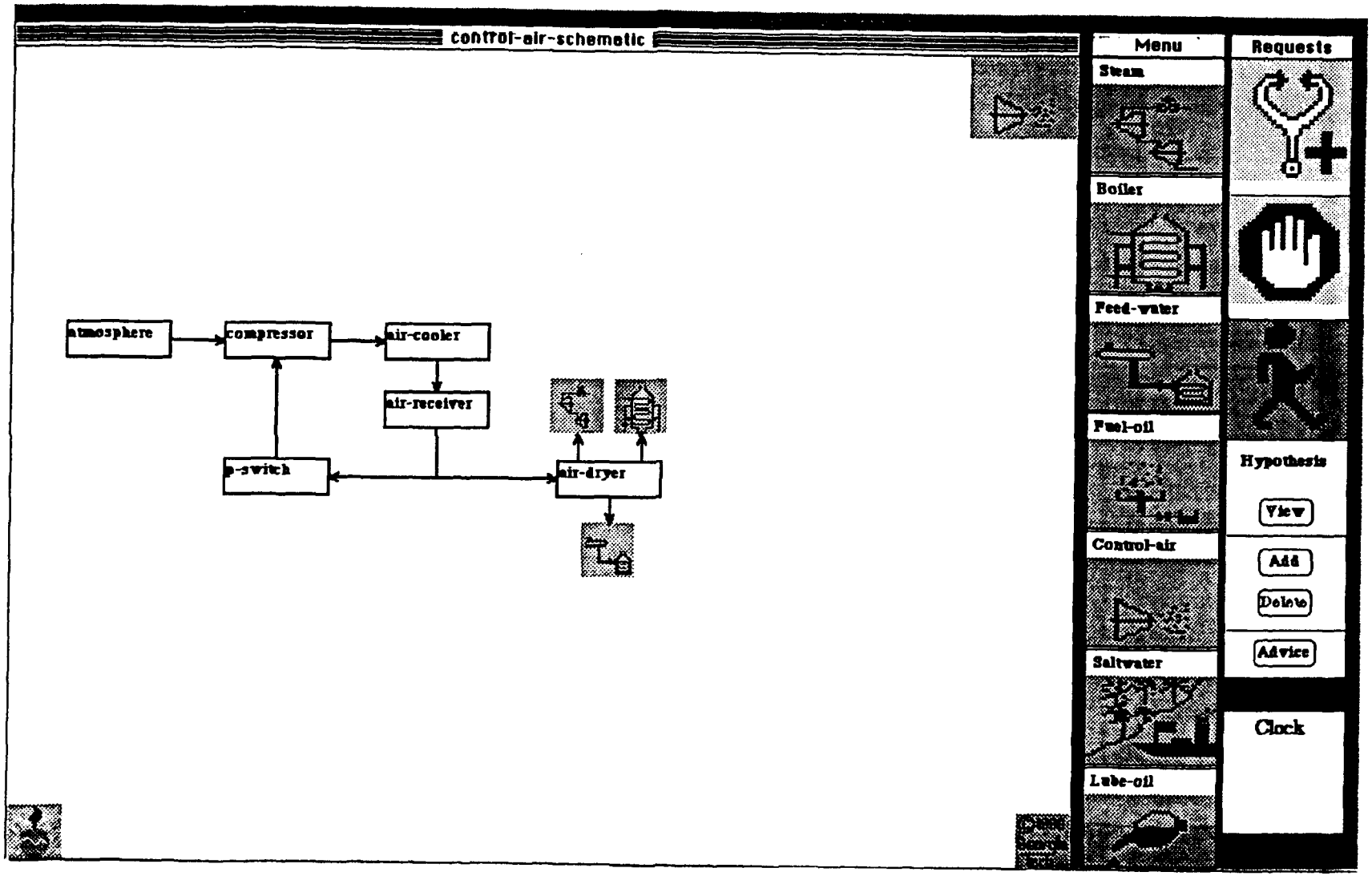


Figure 5.6e Control Air Schematic

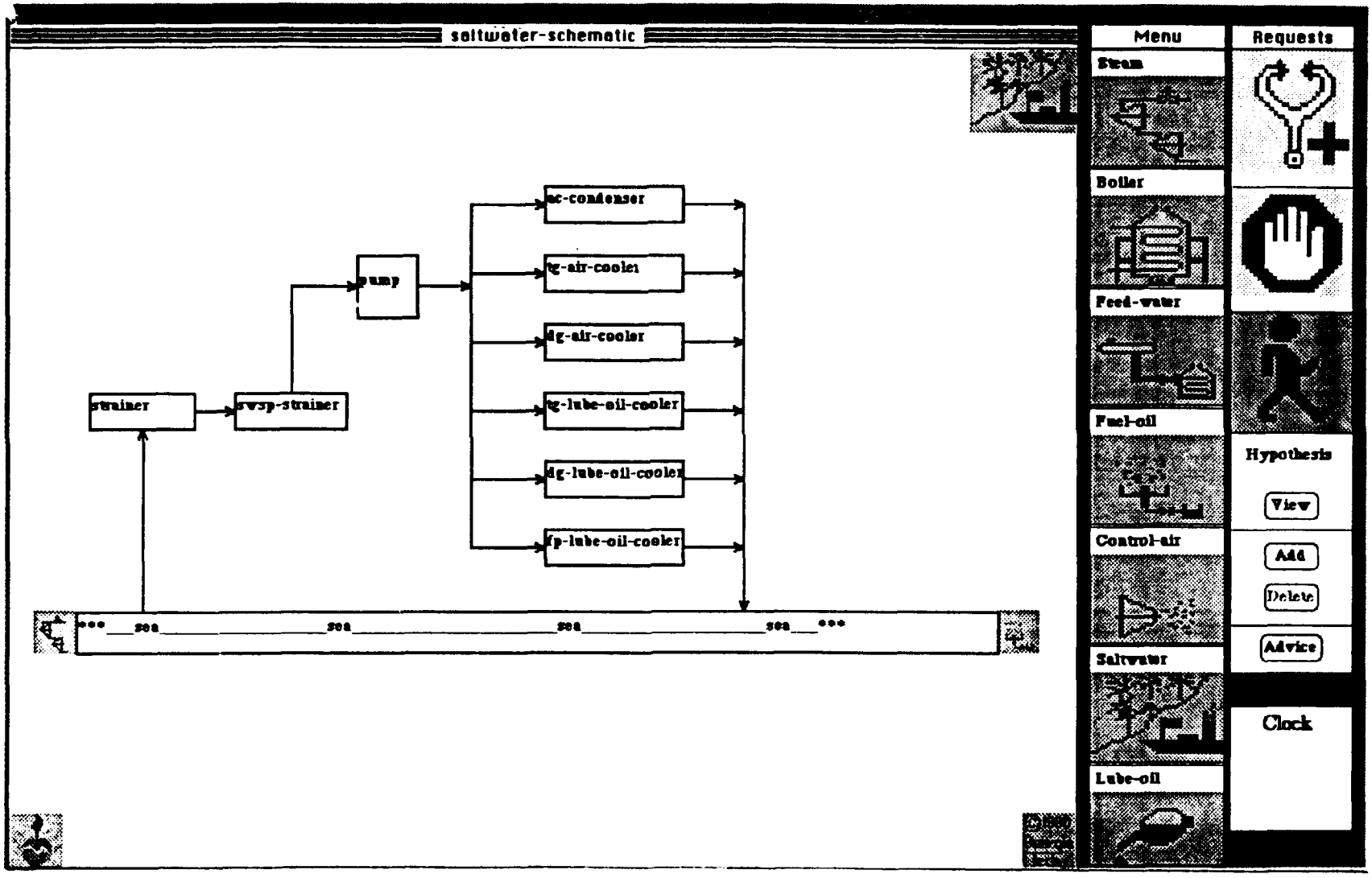


Figure 5.6f Saltwater Schematic

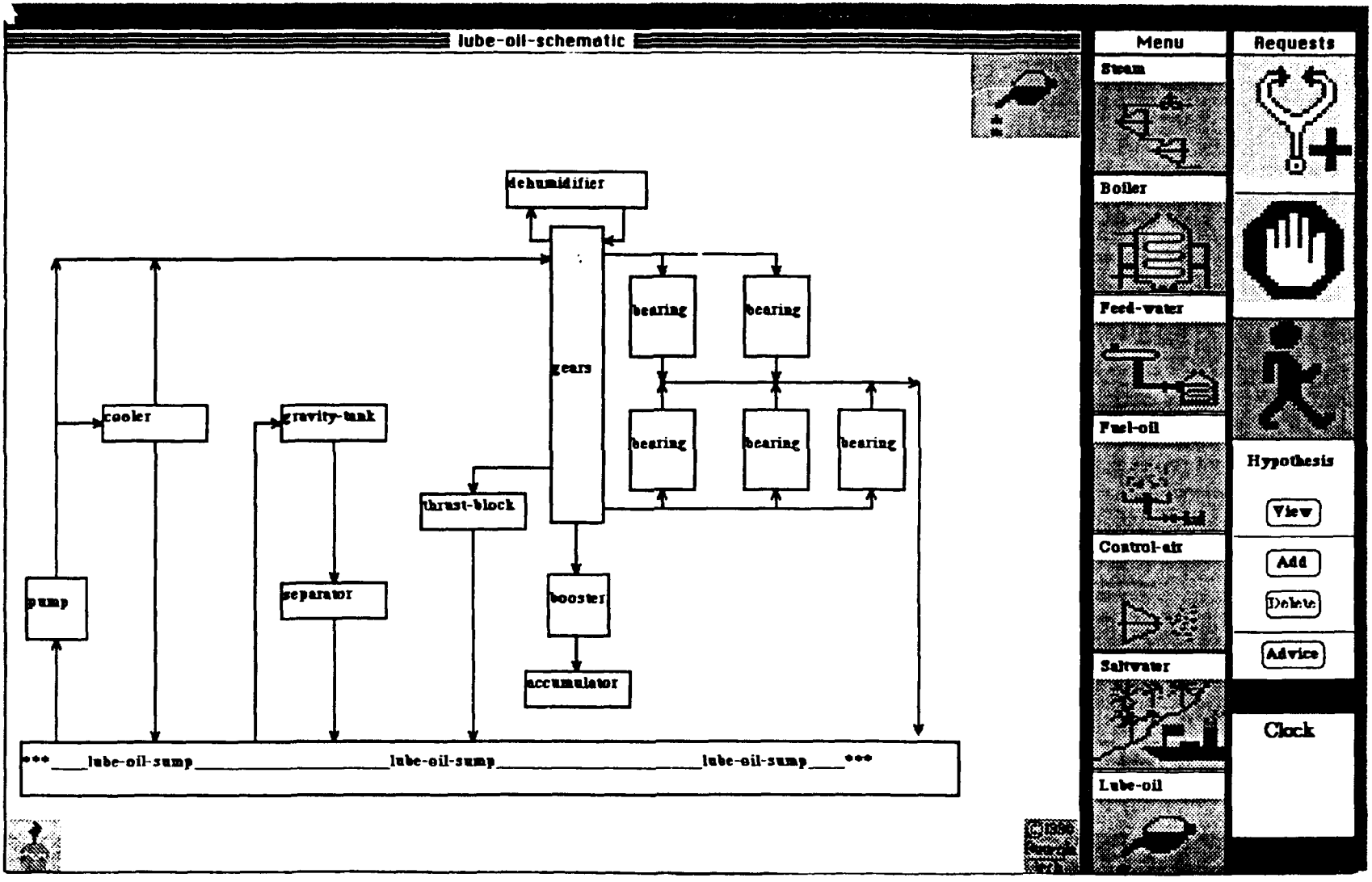


Figure 5.6g Lube Oil Schematic

Most components of **Turbinia** are uniquely represented in one of the seven schematics. However, there are a few that have multiple representations. For example, the condenser and the hp heater appear in both the steam and the feed water schematics (Figures 5.6a & c). Additional occurences of these components in multiple schematics is indicated by schematic icons attached to these components. Unlike other schematic icons, these do not have a connector attached to them. If the student clicks on these icons, the display switches to the other schematic that has the second occurence of the component. The second occurence of the component in the new schematic is indicated by a highlighted schematic icon attached to the component.

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

The three types of gauges in **Turbinia**, pressure, temperature, and flow-or-level, are represented by icons with letters P, T and L inscribed in them to indicate pressure, temperature, and level respectively. Although there is no visible distinction between the level and the flow gauges, the level gauges are mounted on the components while flow gauges appear on the connectors between components. Level gauges in **Turbinia** are attached to tanks (deaerating feed tank, fuel oil settling tank, atmospheric drain tank, distillate tank, hotwell, and drum) and the only gauge that measures flow is located in the fuel oil path across the strainer in fuel oil schematic (Figure 5.6d).

When the student clicks on a displayed gauge to probe its reading, an icon appears near the gauge. This icon is a qualitative representation of the current gauge reading. **Turbinia** uses five different qualitative representations of state values. These five are low, slightly low, normal, slightly high and high; each is represented by an icon as shown in Figure 5.7.

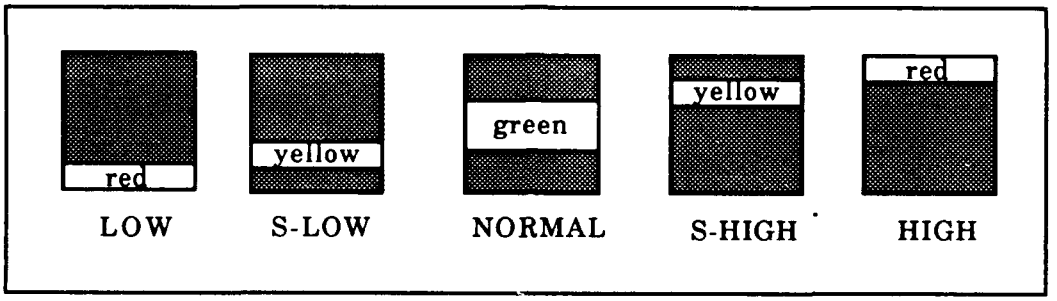


Figure 5.7 Qualitative State Representation

Thus, if the student clicks on the drum in the boiler schematic, all four gauges attached to the drum are displayed on the schematic as a result of this investigative action. There are two pressure gauges, one on the flue gas connector to the economizer and the other on the steam drum. There is a temperature gauge on the feed water connector from the economizer and a level gauge on the steam drum (See Figure 5.8). Now, if the feed water level in the boiler drum is low, the informative action of clicking on the level gauge attached to the drum will result in the appearance of a low level icon below the gauge (also shown in Figure 5.8).

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

The student can access any displayed gauge or a gauge reading only until a new investigative action is taken. When a new investigative action is taken, the gauges and the gauge readings of the previously investigated component that were visible disappear. This was done to achieve a good mapping between the operator task in the real and simulated domains. In the real domain, components of power plant are often spread over a large area, or sometimes in many rooms. Thus, multiple gauges cannot be viewed together. Even when the components are located in the same room, often their sizes are huge and it is often not possible to view gauges attached to two components at the same time.

When the student is ready to diagnose the fault, a request for conveying the diagnosis must be submitted. This is done by clicking on the diagnose icon in the requests menu and it puts **Turbinia** in a diagnose mode. After switching to the diagnose mode, **Turbinia** asks the student to select the suspected component. This message is conveyed to the student through text appearing in the tutor dialog at the bottom of the small screen (Figure 5.9). The student can now use the same action that was earlier used to pick the component for investigation to identify the failed component. If the student's diagnosis is correct, a congratulatory message appears on the tutor dialog (Figure 5.10). Otherwise, an error dialog accompanied by a beep is displayed over the schematic. This error dialog is shown in Figure 5.11. The student can close this error dialog by selecting one of the two options available. The student can either revise the diagnosis by choosing "try again" or get back to the troubleshooting mode by choosing "investigate".

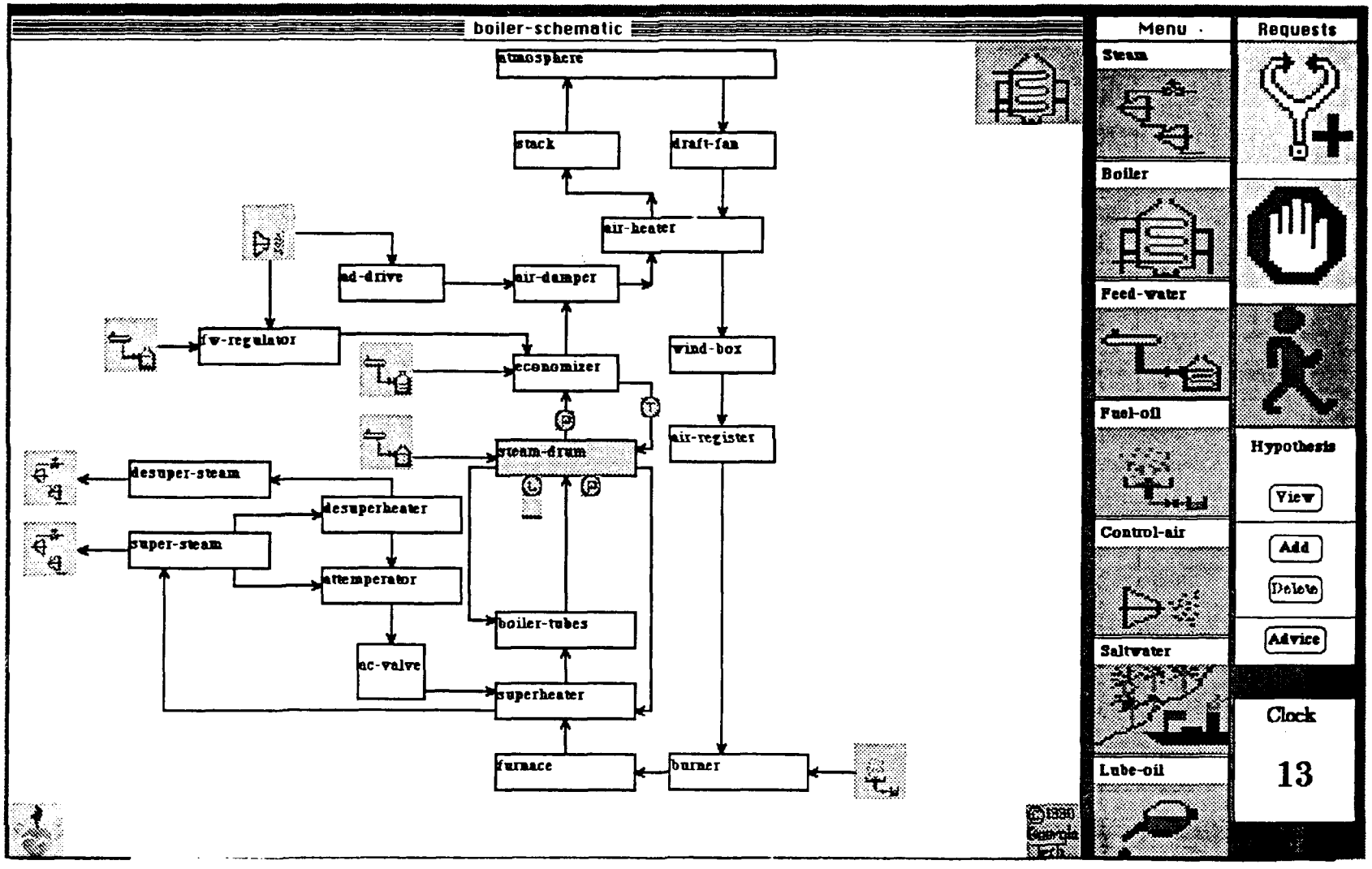


Figure 5.8 Boiler Schematic Showing Gauges on the Steam Drum

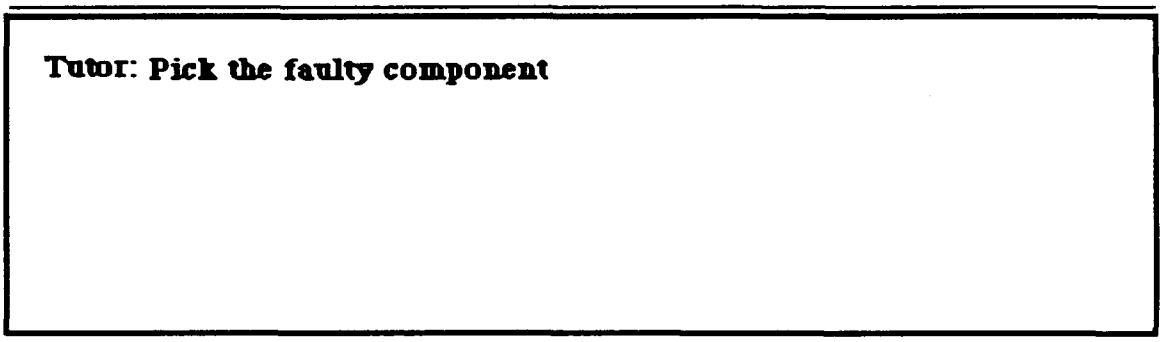


Figure 5.9 Tutor's Instructions to Select the Suspected Component

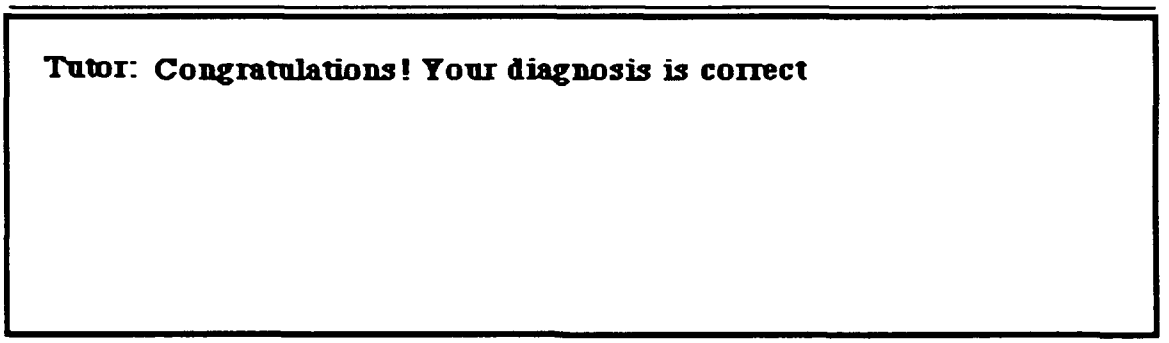


Figure 5.10 Message of Congratulations on Correct Diagnosis

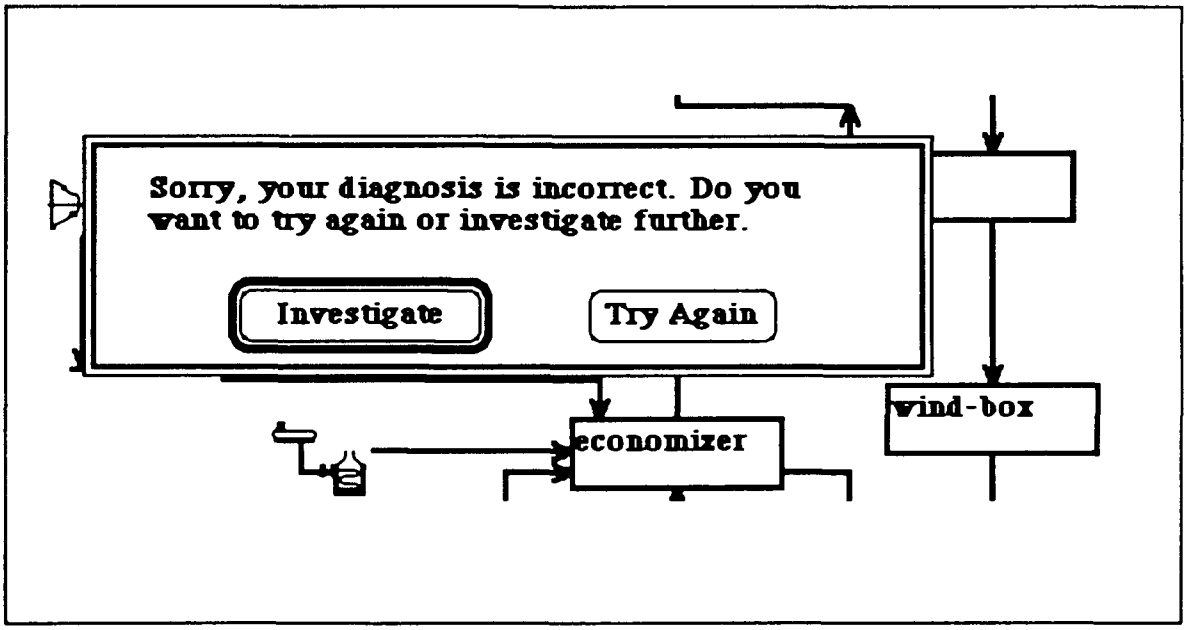


Figure 5.11 Error Dialog for Incorrect Diagnosis

In addition to the components, connectors and schematic icons, all schematics display two other icons that do not represent a schematic. One icon appears on the left bottom corner and the other on the right bottom corner of all schematics. The icon on the bottom right corner is a Georgia Tech copyright icon. This icon is disabled and has no response. The icon at bottom left corner is a symptom icon and is used to recall the initial symptoms. Thus, the student can access the ship's initial operating conditions and the initial symptoms at any time.

Interaction with Vyasa

Passive Mode

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

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

When the passive tutor is invoked using the stop icon, a *help-levels* passive tutor help dialog appears in the top left corner of the large monitor. This dialog box is also shown in Figure 5.12. This dialog has seven buttons of which two are enabled. The two highlighted buttons indicate the levels of help that the tutor can provide in the passive mode. These two levels are the failure and system knowledge help. Using these buttons the student can access knowledge concerning specific modes of failure, components, subsystems, and fluid paths. The information accessed by the student is presented in textual or graphical form.

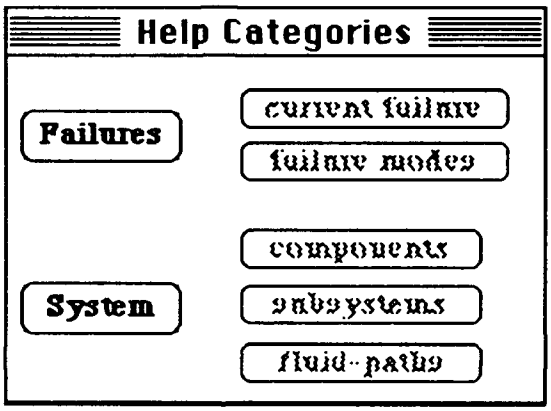


Figure 5.12 Help-Levels Dialog

To access knowledge about the system, the student must choose the "system" button in the help-levels dialog. Following the selection of the "system" button, the "components", "subsystems", and "fluid-paths" buttons are enabled (Figure 5.13). These three buttons provide the student with further options to select the type of system knowledge. When the student selects any one of these three buttons, a new passive tutor help dialog associated with the selected button appears next to the help-levels dialog. This new dialog also contains several selectable items. The student can, by selecting items in the passive tutor help dialogs, explore the entire knowledge-base of the tutor at the component, subsystem and fluid path levels.

For instance, if the student selects the "subsystems" button in the help-levels dialog, another dialog that lists all the subsystems in the power plant appears next to the help levels dialog (Figure 5.14). The student can select a subsystem from the list of subsystems to further explore the tutor's knowledge of the selected subsystem. The selection is made by clicking on the subsystem name listed in the dialog. After the student selects a subsystem, a dialog box listing the selected subsystem and the types of information concerning the selected subsystem that can be accessed by the student is displayed. Figure 5.15 is an example of such a dialog box which is displayed when the student selects the combustion subsystem. This dialog provides the student with three options: (1) view the components that make up the combustion subsystem, (2) view the fluid paths that pass through the combustion subsystem, or (3) ask for the description of the function performed by the combustion subsystem. If the student decides to view the components that constitute the combustion subsystem, the button marked "show subsystem" must be selected. A click on "show subsystem" results in the appearance of a dialog box containing the seven schematic icons (Figure 5.16, the grey colored buttons indicate the sequence of actions taken thus far). Those icons that represent schematics in which combustion subsystem can be found are highlighted in this dialog box. Since the combustion subsystem can be found in boiler and fuel oil schematic, the boiler and the fuel oil schematic icons are the ones highlighted in Figure 5.16. A click on any one of these icons displays the schematic represented by the icon and highlights in red all components that constitute the combustion subsystem in that schematic (Figure 5.17).

The set of hierarchically organized passive tutor help dialog menus that guides the student's exploration of system knowledge also maintains context-sensitivity and offers substantial flexibility to the student. For example, when the student inquires about fluid paths in Figure 5.15, the tutor offers a choice of selecting a liquid or gas path from only

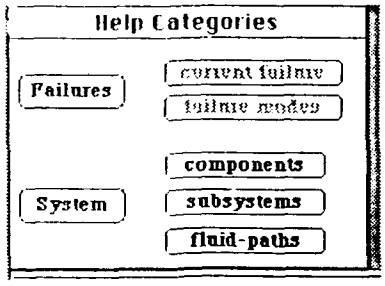


Figure 5.13 Help-Levels Dialog with "System" Buttons Enabled

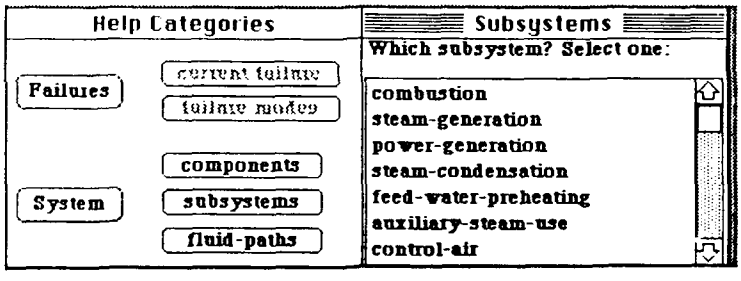


Figure 5.14 Passive Tutor Dialog to Select Subsystem

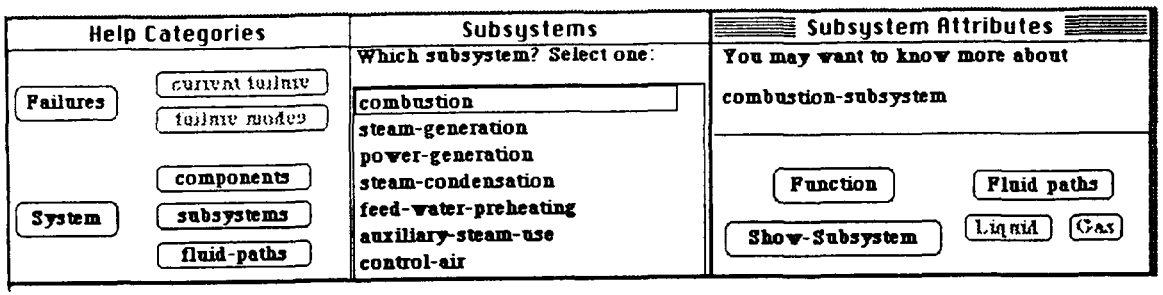


Figure 5.15 Passive Tutor Dialog to Access Subsystem Related Knowledge

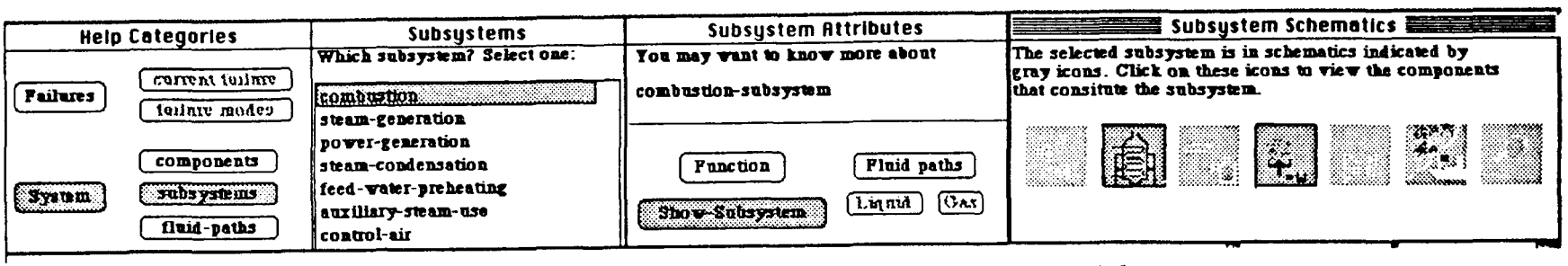


Figure 5.16 Passive Tutor Dialog to Display Selected Subsystem

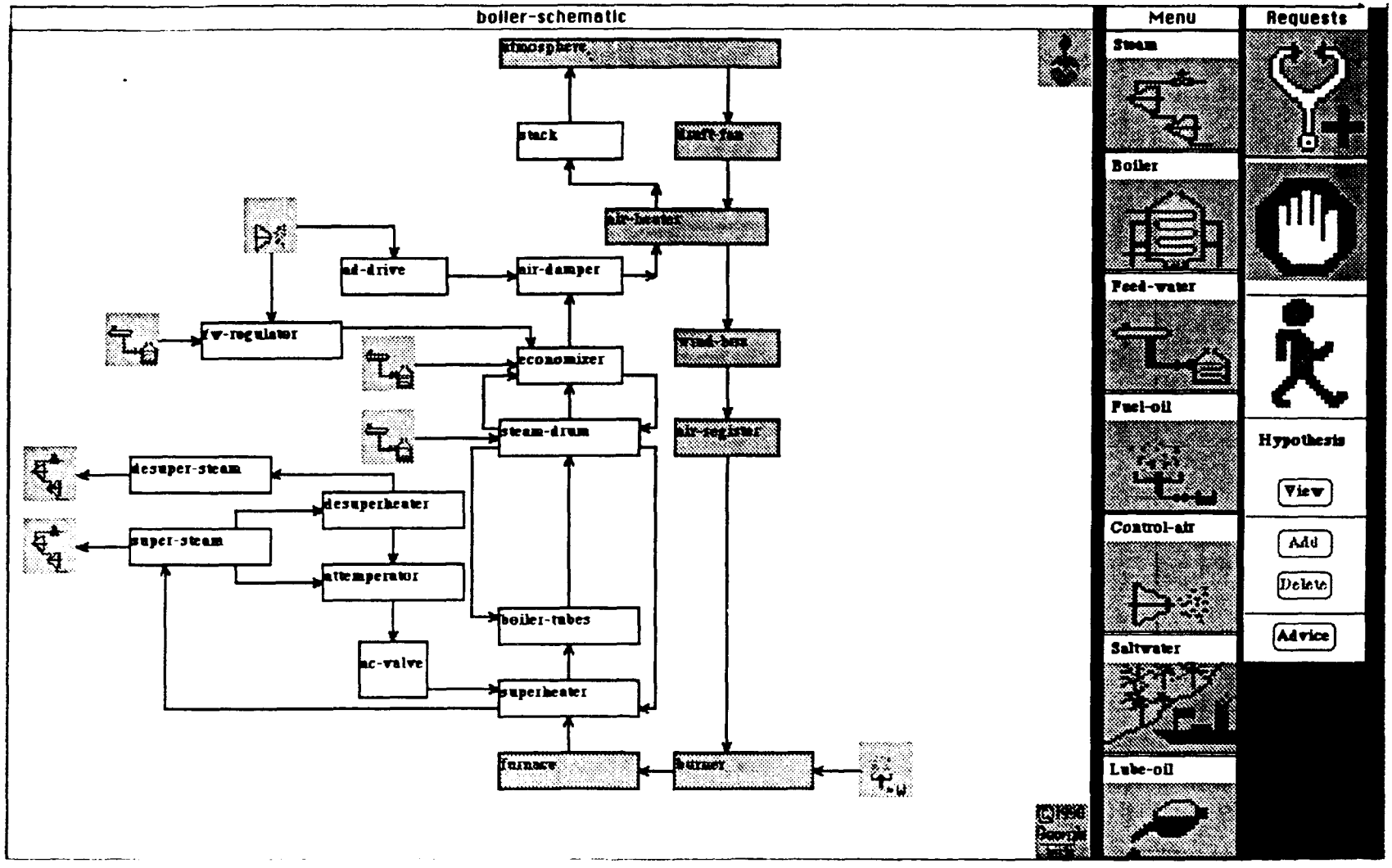


Figure 5.17 Combustion Subsystem Highlighted in Boiler Schematic

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.

In general, Vyasa responds to an interaction in passive tutor help dialogs by either highlighting the lowlighted buttons in the currently active dialog box or displaying a new dialog box with certain items highlighted. The student can select an enabled button or any highlighted item in the displayed dialogs to make the queries more specific. Sometimes, it may be necessary for the student to select an item in the schematic or use the keyboard to make the query more specific. For example, when inquiring about a component, the student must select the component by either clicking on the component in the schematic or typing its name using the keyboard. In any event, when a query is specific enough for Vyasa to comprehend, it responds with an answer. The answer is presented as text in a dialog box or as graphics in the schematics.

The flexibility in interaction with Vyasa also enables the student to alter the query at any time. This can be done by merely discontinuing the sequence of actions necessary to express the current query and switching to a new query by clicking on a highlighted item in any of the displayed dialog boxes. In such cases, only the dialog boxes relevant to the new query are kept open by the tutor while all others are closed.

Thus, the student can access tutor's knowledge of the system by following a sequence of straightforward interactions with passive tutor help dialogs. The tutor assumes the responsibility of guiding the student's interaction and provides lot of flexibility to the student to express and alter queries. Figure 5.18 provides a summary of interactions with the passive tutor help dialogs to access the different components of system knowledge.

The same help-levels dialog that is used to access knowledge of the system can also be used to obtain information concerning failures. To do so, the student must choose the "failures" button in the help-levels dialog shown earlier in Figure 5.12. Following the selection of the "failures" button, all buttons related to the "system" button are disabled and all visible dialogs concerning any earlier query related to system knowledge are closed. At the same time, the "current failures" and "failure modes" buttons are now enabled. Using the "failure-modes" button the student can access information concerning typical system behavior associated with each mode of failure in the liquid and gas paths (Figure 5.19).

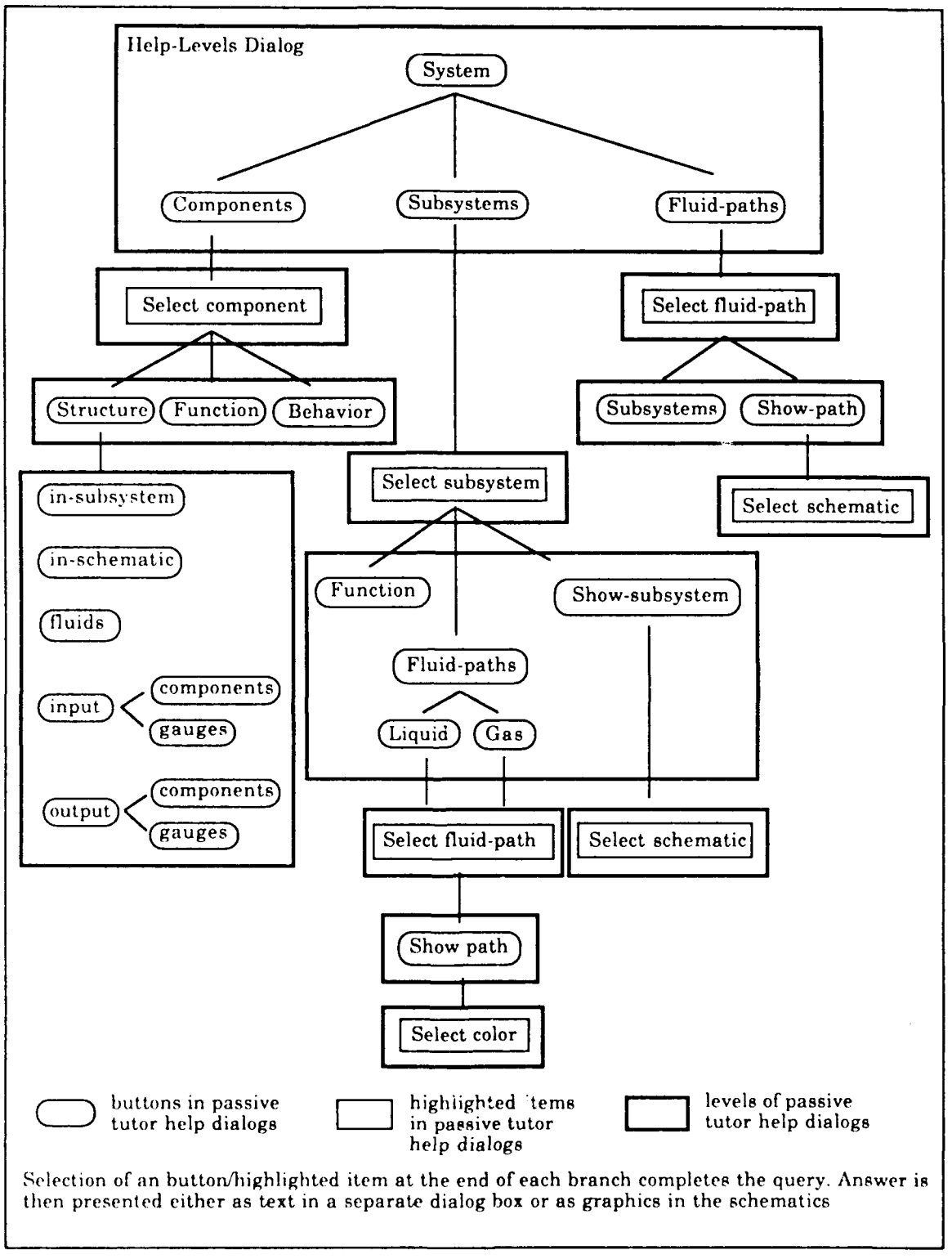


Figure 5.18 Summary of Interactions with Passive Tutor Dialogs to Access System Knowledge

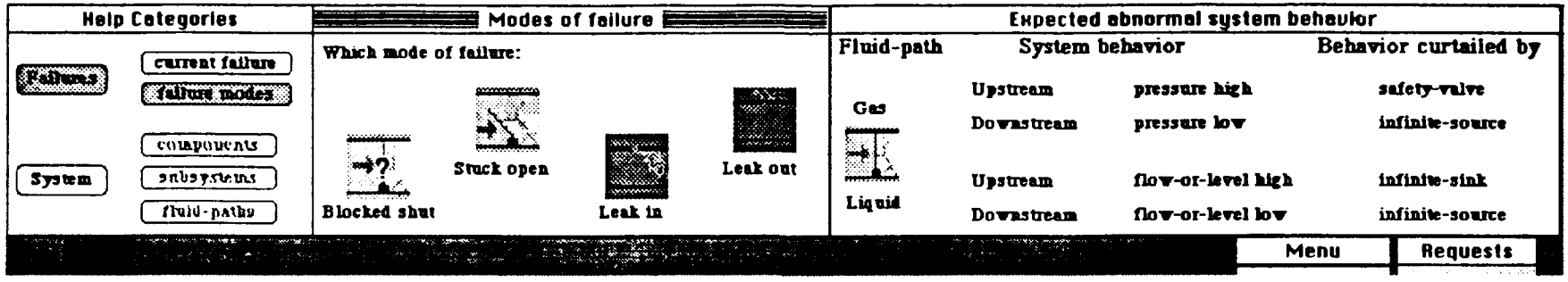


Figure 5.19 Passive Tutor Dialog to Display Failure Mode Knowledge

Using the "current failure" button the student can bring up a clipboard that extends to the smaller screen on the right. Figure 5.20 provides a summary of interactions with the passive tutor help dialogs to access tutor's failure knowledge.

The clipboard presents a summary of observed results from the student's diagnostic actions. Based on the observed gauge readings, the clipboard displays the schematics, subsystems and fluid paths that contain the affected gauges. The extended portion of the clipboard on the smaller screen displays the gauges probed along with their gauge readings. For example, if the student has only investigated the level gauge on the steam drum since the start of the troubleshooting session and found it to be low, the clipboard will show the boiler schematic, steam generation subsystem and feed water path as the affected schematic, subsystem and fluid path respectively (Figure 5.21). In addition, the extended portion of the clipboard displays the drum's level gauge investigated by the student (Figure 5.22). By displaying this information, the clipboard eliminates a student's need to use paper and pencil to keep a record of the diagnostic information gathered during troubleshooting.

The clipboard helps the student in more than one way. Apart from book-keeping, the clipboard informs the student if any gauge reading has changed since it was last viewed. It does so by displaying a blue colored marker next to the gauge reading (Figure 5.23). Once the student re-investigates the gauge, the marker disappears and the clipboard is updated to contain the latest information. When a gauge reading is not yet stable, the blue marker next to the gauge appears and disappears on its own. This is an indication for the student that the gauge reading is perhaps oscillating.

Furthermore, the clipboard also informs the student about the most likely mode of current failure, if and when it can be inferred from the tests conducted. For instance, while investigating the cause for low feed water level in the drum, if the student in addition to observing the low feed water level in the drum investigates the deaerating feed tank and finds the feed water level to be high, "blocked-shut" is posted as the most likely mode of failure (Figure 5.24). This mode of failure is inferred from the two gauge readings observed by the student: a high feed water level in the deaerating-feed-tank and a low feed water level in the steam drum.

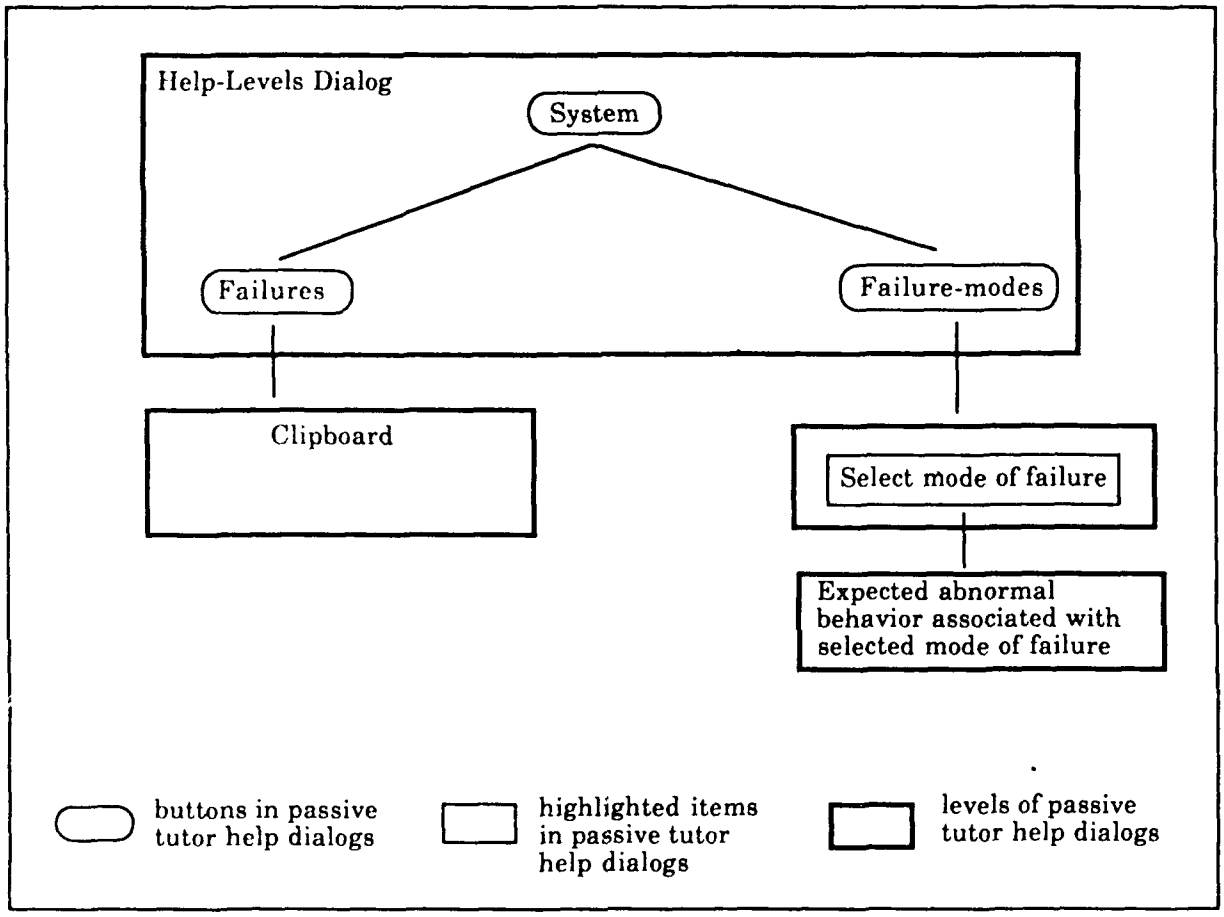


Figure 5.20 Summary of Interactions with Passive Tutor Dialogs to Access Failure Knowledge

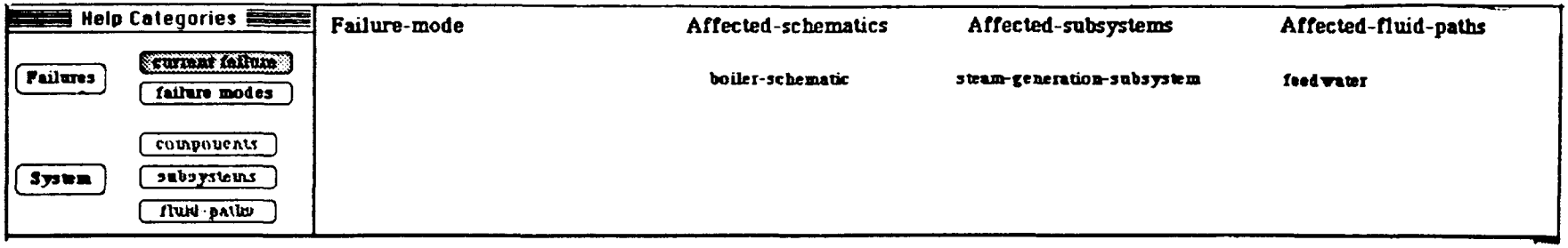


Figure 5.21 Clipboard

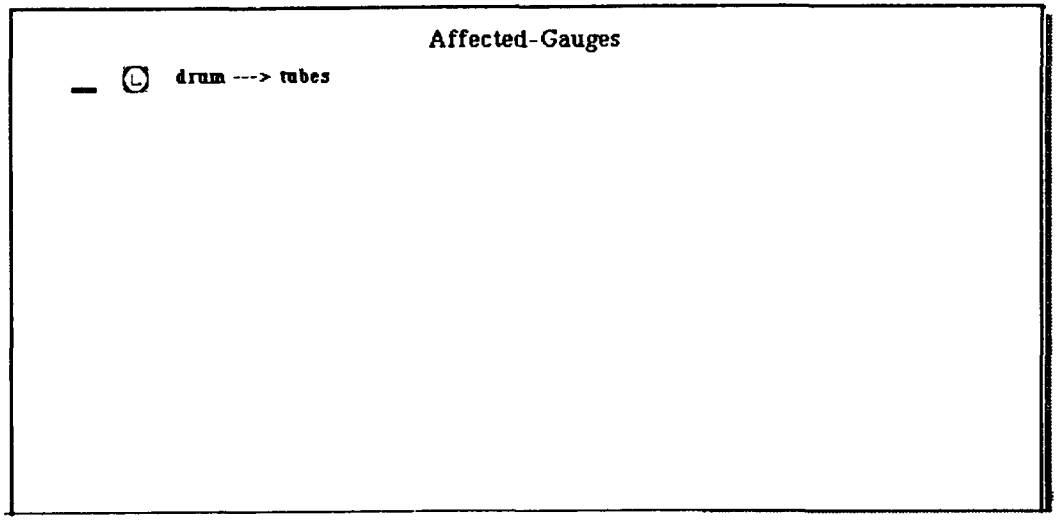


Figure 5.22 Extended Portion of the Clipboard

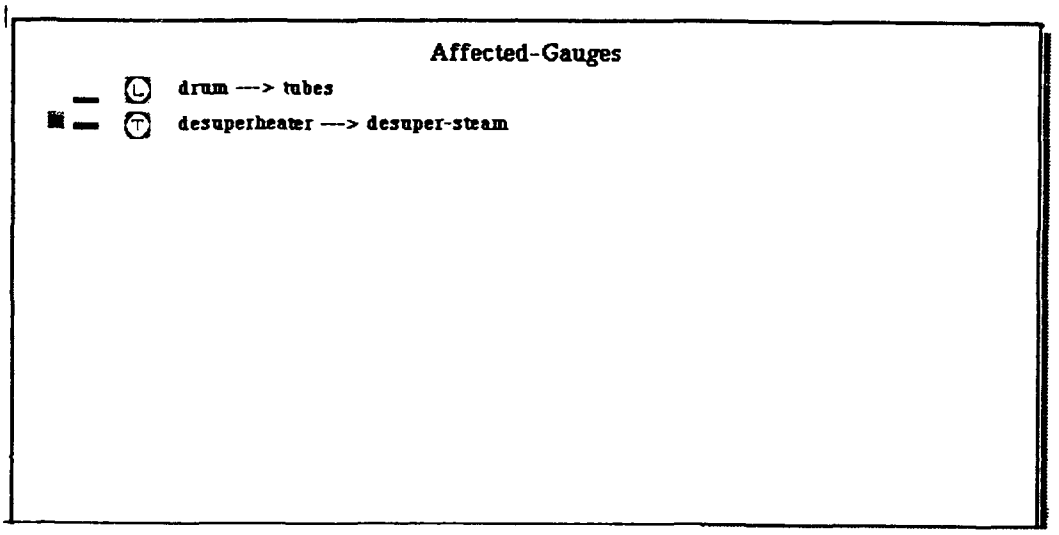


Figure 5.23 Clipboard Indicating Change in Gauge Reading

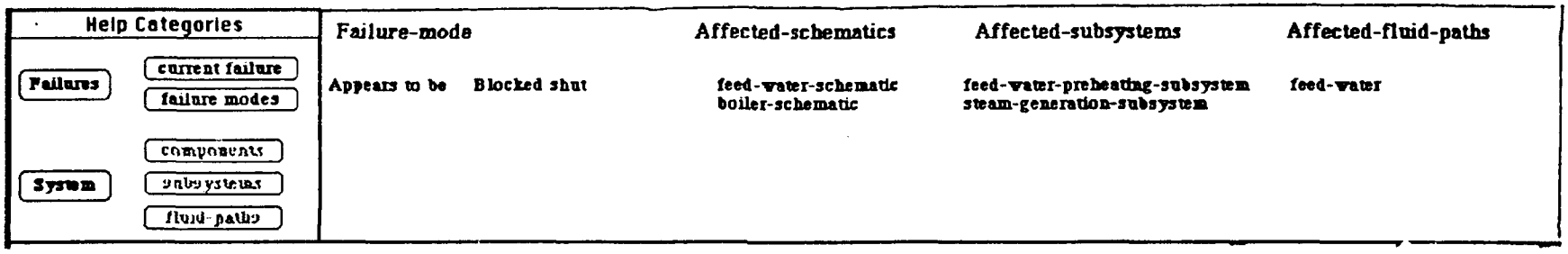


Figure 5.24 Clipboard Indicating Blocked-Shut Mode of Failure

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

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

Active Mode

In the active mode, Vyasa often intervenes to communicate with the student. It does this through instructions presented on the tutor dialog, accompanied by a beep. These instructions are delivered following the evaluation of a student's misconception.

For instance if the student investigates schematics, subsystems or fluid paths unaffected by the failure, the tutor delivers the appropriate instructions to guide the student away from unaffected portions of the power plant (Figures 5.25). Such instructions are usually displayed for a fixed, but short, period of time unless the instructions lose their context due to system dynamics.

In addition to guiding the students with instructions, the tutor in the active mode is capable of helping the students with their hypotheses. Student's hypotheses concerning failures are either solicited by the tutor or voluntarily disclosed by the student. In either case, the manner of communicating the failure hypotheses to the tutor is identical. First, the tutor asks the student to select the suspected component responsible for the current abnormal system behavior (Figure 5.26). Once the selection is made, the student is prompted to identify the failure mode. This identification is made by picking the appropriate icon in the displayed dialog box (See Figure 5.27, condensate pump is the selected component in this example).

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

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

Tutor: You seem to be investigating a schematic unaffected by the current failure

Tutor: You seem to be investigating a subsystem unaffected by the current failure

Tutor: You seem to be investigating a fluid path unaffected by the current failure

Tutor: You seem to be investigating a subsystem and a fluid path unaffected by the current failure

Figure 5.25 Examples of Instructions from Vyasa

Tutor: **Provide your hypotheses. Select the component you suspect is responsible for the current abnormal behavior. You may select more than one component.**

When finished, click on the "Done" button.

Done

Figure 5.26 Tutor Soliciting Hypotheses from the Student

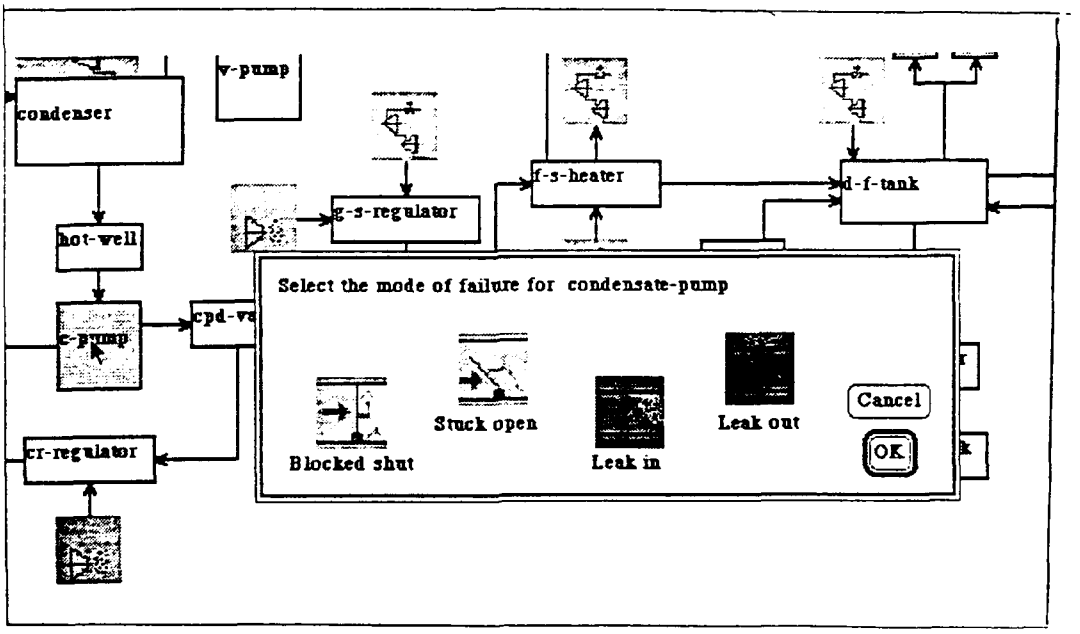


Figure 5.27 Student Communicating the Suspected Mode of Failure

the tutor attempts to rectify the student's misconception by providing instructions in the manner shown in Figure 5.28.

Vyasa also provides help with the failure hypothesis without intervention. This help is provided on request. All communications with Vyasa concerning failure hypotheses, including a request for help, are carried out via the hypothesis menu (Figure 5.29). Interaction with the hypothesis menu is described next.

Hypothesis Menu

Hypothesis menu appears below the requests menu on the large monitor. It has four buttons. The "View" button is used to view the list of hypotheses provided by the student to the tutor. The "Add" button is used when the student wants to specify a new hypothesis. The "Delete" button is used to remove a hypothesis from a list of hypotheses. Hence the delete button gets highlighted only after the student has provided a set of hypotheses. Finally, the "Advice" button is used to seek help concerning a particular hypothesis.

To view, delete, or seek advice on a failure hypothesis, the student must use the "View", "Delete" and "Advice" buttons respectively. The tutor displays the hypotheses provided by the student in a dialog box on the small screen for review (Figure 5.30). The interaction with the tutor to delete a hypothesis or to seek advice is also straightforward. The tutor prompts the student for every action through the tutor dialog. When deleting a hypothesis, the student first selects the hypothesis in the dialog box shown in Figure 5.31 and then clicks on the highlighted "Delete" button in the same dialog box. When seeking advice on a hypothesis, the student first selects the hypothesis in the dialog box shown in Figure 5.32 and then clicks on the highlighted "Help" button. Any advice from the tutor is displayed in a separate dialog box over the "Help" button (Figure 5.33).

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

condensate pump has not failed in the Blocked-Shut mode

Figure 5.28 Example of Hypothesis Aiding with Intervention

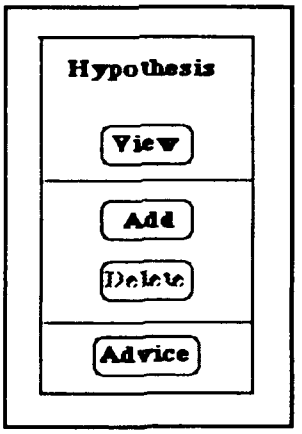


Figure 5.29 Hypothesis Menu

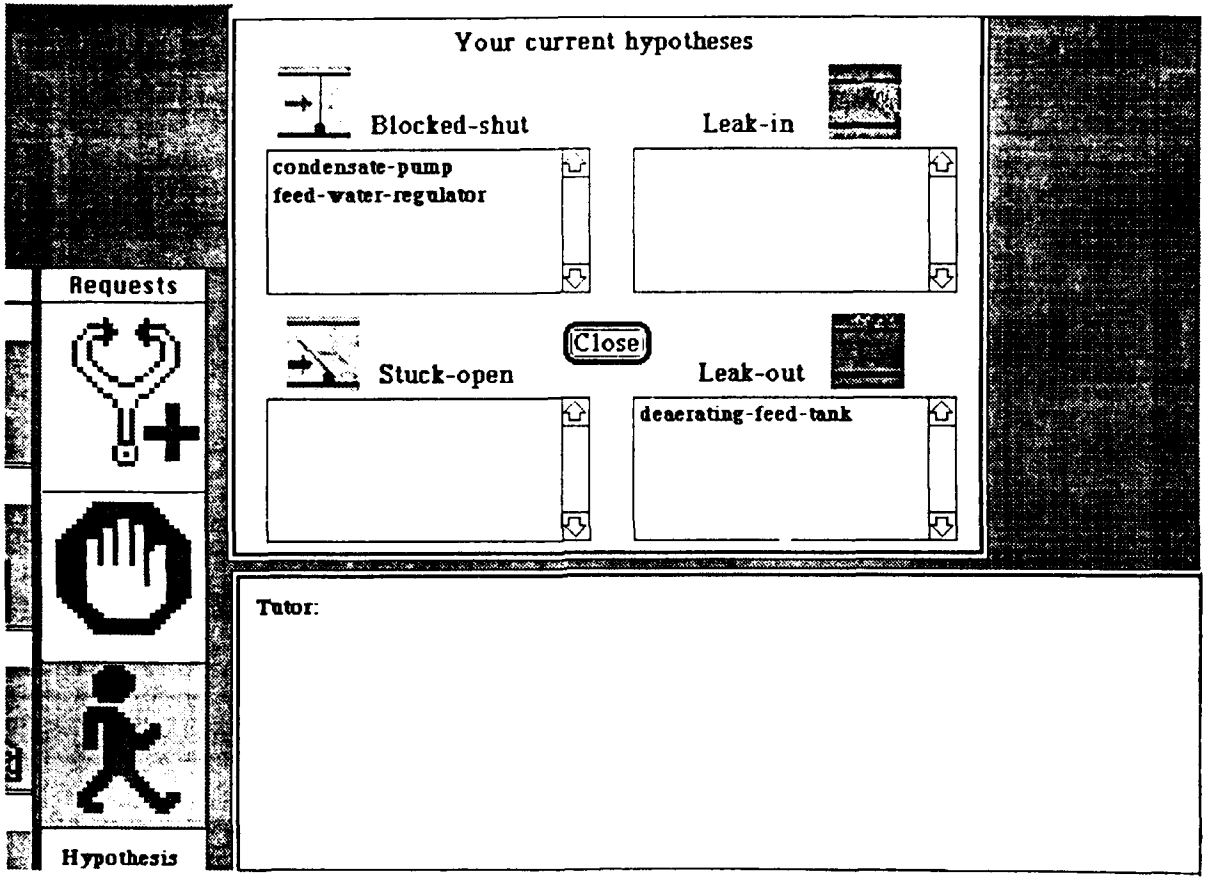


Figure 5.30 Review Hypotheses Dialog

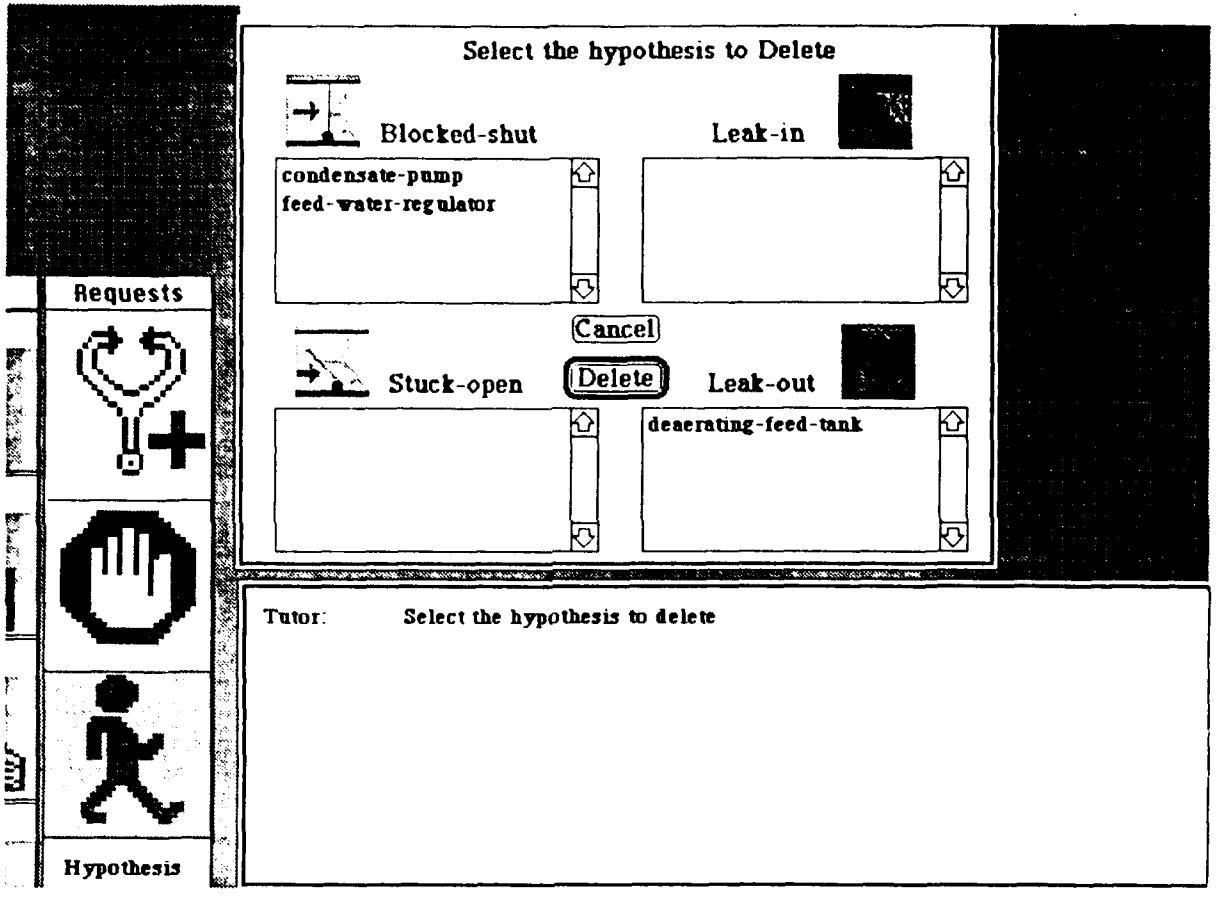


Figure 5.31 Delete Hypothesis Dialog

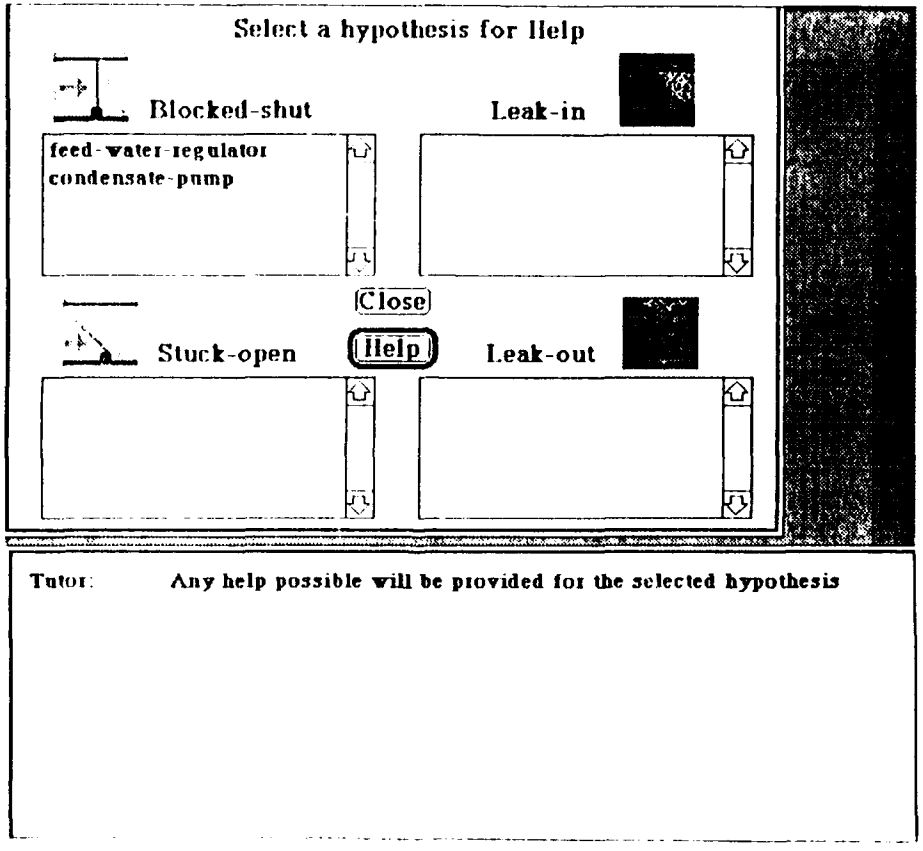


Figure 5.32 Advice Hypothesis Dialog

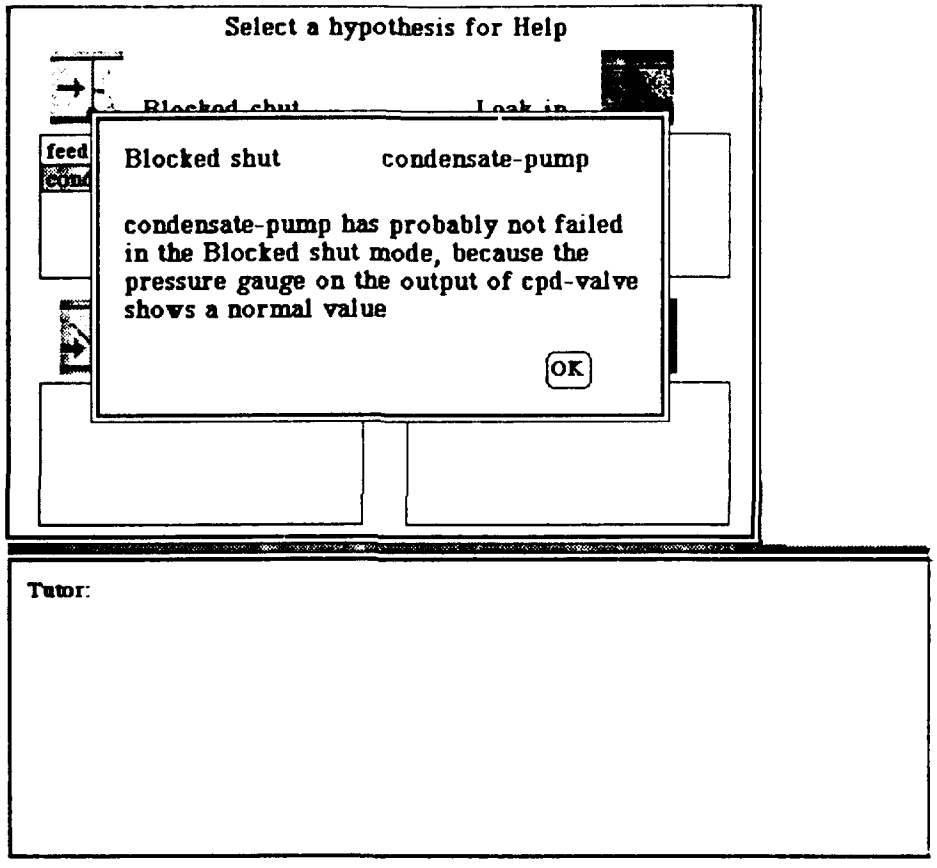


Figure 5.33 Example of Hypothesis Aiding without Intervention

In addition to the interaction described thus far in this chapter, the student interacts with the instructional system at the end of every problem solving session to view the solution to the problem last presented. The solution and the interaction with the instructional system depends upon whether the student was using only the simulator or was also aided by the tutor. In either case, the student is presented with a dialog box shown in Figure 5.34 at the end of each problem. While students using just the simulator see only the solution as shown in Figure 5.35, the students aided by the tutor have the option to view the explanation for each observed abnormal behavior (Figure 5.36). If the student decides to click on the "Explain" button in the dialog box shown in Figure 5.36, explanations are provided for all observed abnormal system behavior. These explanations containing causal reasons for each abnormal gauge reading are presented, one at a time, and in the order in which the gauges are affected by the failure. In presenting the reasons for each abnormal gauge reading, first the schematic which contains the affected gauge is displayed, then the affected gauge along with its gauge reading are made visible and finally an explanation for the abnormal reading is displayed in a dialog box on the small screen (Figure 5.37). Once the student has read the explanation and clicked on the "OK" button, the tutor proceeds to provide a similar explanation for the next affected gauge. This process continues until the tutor completes providing an explanation for each abnormal gauge reading caused by the failure.

This completes a description of the student-tutor interface of **Turbinia-Vyasa** and the valid forms of operator interactions at this interface. In the next chapter, an experimental study to evaluate the ITS architecture implemented in **Turbinia-Vyasa** is described.

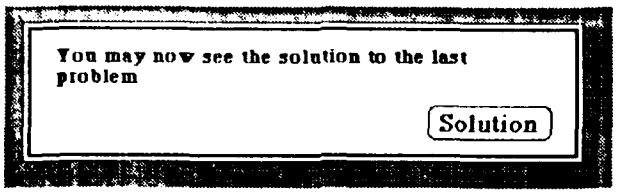


Figure 5.34 Dialog to Request Solution

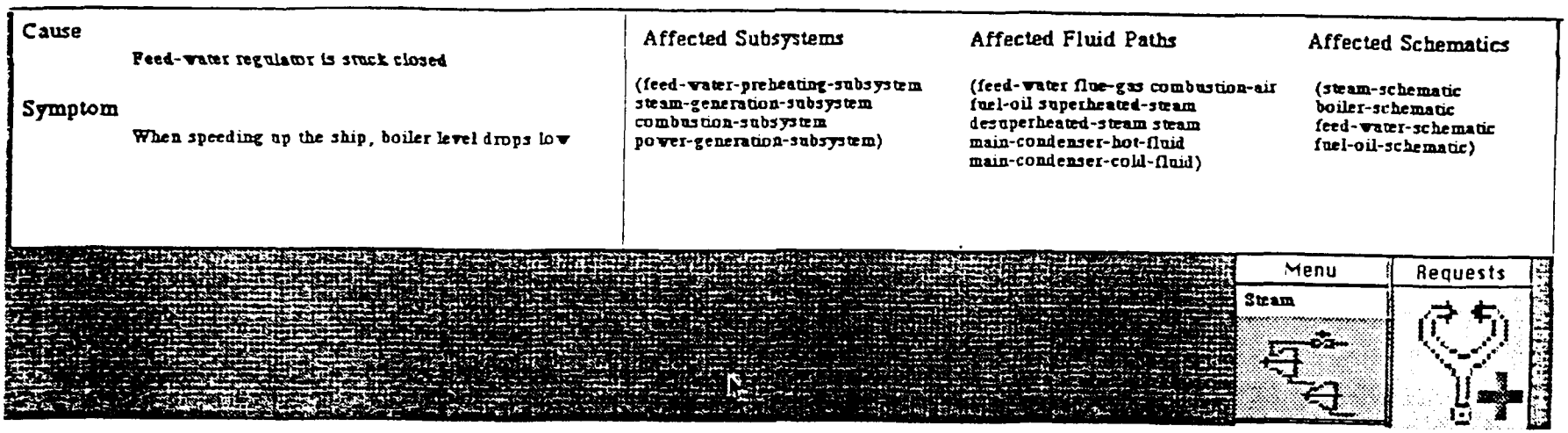


Figure 5.35 Solution for Students Trained on Simulator

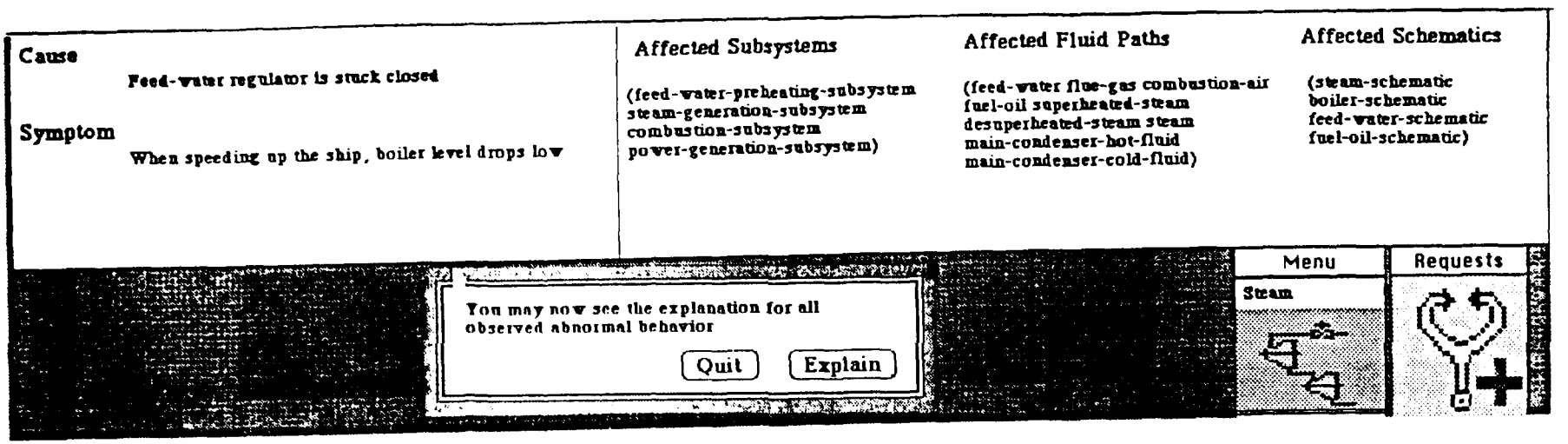


Figure 5.36 Solution for Students Aided by the Tutor

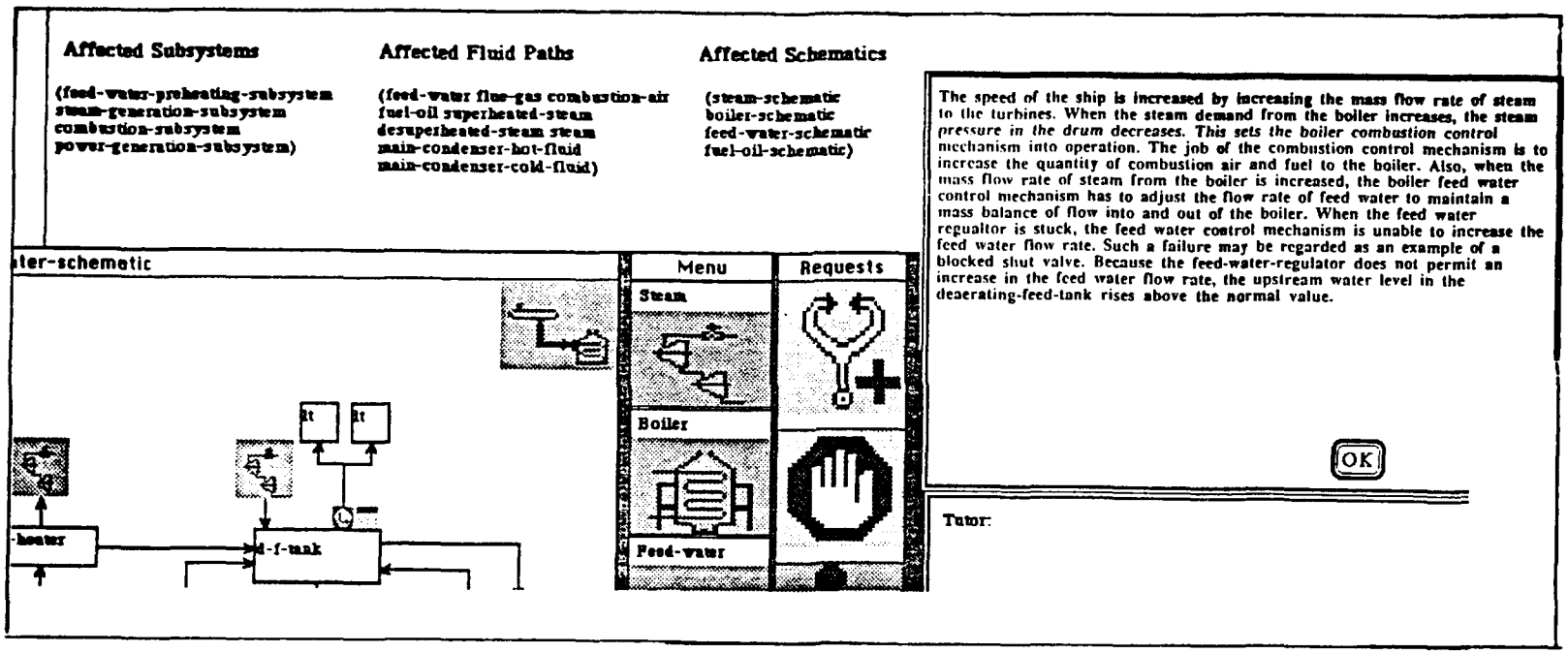


Figure 5.37 Explanations for Abnormal System Behavior

CHAPTER VI

EXPERIMENTAL EVALUATION OF TURBINIA-VYASA

The final phase of this research involved an experimental evaluation of the architecture of **Turbinia-Vyasa**. Prior to beginning formal experiments to test the effectiveness of **Turbinia-Vyasa**, 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. It was intended to evaluate the aiding material and instructional strategies. A formal experiment designed to measure the diagnostic performance of operators trained with and without the aid of **Turbinia-Vyasa** followed. The discussion in this chapter deal with the details of the evaluation.

Informal 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 **Turbinia** 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 confidence about 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. Pilot experiments began after this preliminary evaluation.

Pilot Study

A pilot experiment preceded the formal 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 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 had engineering backgrounds with several courses in thermodynamics and were well exposed to research issues in human-machine systems. However, since the pilot subjects were not drawn from the population of subjects that were to participate in the formal experiment, the experimenter could not get a good idea about the trend of the formal experimental results.

The pilot study was conducted with **Turbinia-Vyasa** operating in the active mode. Since the simulator and the passive tutor are also functional in this mode they were not subjected to a separate pilot study. 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 solved 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 software errors. These were corrected promptly prior to the start of the formal experiments. Furthermore, five problems out of the twenty-nine obtained from the Marine Safety International were eliminated from the set to be used in the formal experiment because they were either redundant or exhibited inconsistent behavior.

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.

Finally, at the conclusion of the pilot study, the experimenter was better experienced to answer questions from subjects participating in the formal experiment. This was considered important as answers to similar questions by subjects in the three experimental conditions had to be consistent and required advance preparation.

Formal Experiment

In the formal experiment, 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 to compare the effect of passive and active tutoring strategies. Thus, the formal experiment sought answers to three queries. First, it addressed the question of feasibility of building an effective computer-based tutor by implementing the proposed architecture. Next, it addressed the question of whether the training by computer-based tutor was better than the training provided by the simulator alone. Finally, it explored whether the level of aiding during the course of training affected performance.

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 **Turbinia**. The second and the third training conditions used the computer-based tutor **Vyasa**.

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.

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

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

Equipment

For the experiment, **Turbinia**, the marine power plant simulator, and **Vyasa**, the computer-based tutor were installed on a dual screen Apple Macintosh II workstation with a Daystar Digital 40MHz accelerator board. This machine was used for all data collection sessions.

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 selected were twenty-four sophomores and six

juniors. Although some of the selected subjects had additional exposure to thermodynamics through course-work or had experience operating the marine power plants, the effects of these factors were not analyzed in the experiment. Therefore, the assignment of subjects to the three experimental groups was done randomly.

For every session completed in both the training and the testing phase each subject was were paid $6 per session. In addition, an award of $25 was promised for the best troubleshooter in each of the three groups based on performance in the two data collection sessions.

All subjects were told about the performance measures that were of interest to the experimenter prior to the experiment. They were also informed that for the purpose of determining the award, the number of problems correctly diagnosed in minimum time was the only measure to be considered.

Experimental Materials

There were separate written instructional manuals for each of the three experimental conditions (Appendix A). 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 subject consent form (Appendix B) and the survey form (Appendix C) were filled by the subjects before the experiment. The consent form is required of all human subjects voluntarily participating in experiments at Georgia Institute of Technology. The survey form was designed by the experimenter to obtain a feel for the subjects' academic background, ship board experience and computer skills.

Three questionnaires (Appendix D) were also filled by the subjects 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 one and was filled by the subjects at the end of the training sessions. Answers to the two questionnaires provided the experimenter with some idea about the knowledge acquired during training. The third questionnaire was filled at the end of the last data collection session. The purpose of this exit questionnaire was to elicit subjective opinions about various aspects of the three training methods.

The subjects were also provided with pencils and a blank paper in every session to take notes, if any, pertaining to the problems and to comment on the tutor and its strategies. The purpose once again was to elicit subjective opinions about the tutor from subjects participating in the experiment.

Experimental Procedure

The experiment was conducted in two phases: a training phase and a data collection test phase. 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 then evaluated in the data collection phase where all subjects were exposed to identical problems on **Turbinia** without the aid of the tutor.

Prior to the start of the experiment, 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 a 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 (Appendix D) prior to the start of the experiment. The subjects were then handed the appropriate written instructional manual for each group. They were advised to read the instructional manual before starting the first session.

Training Phase

Training of subjects in each of the three groups lasted ten sessions. 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 thirteen minutes to solve each problem. If the subject solved the problem in less than the allotted time, the next problem was immediately presented. Thus, if the subject solved one or more of the three problems in a session within the allotted time for each problem, the session could potentially be completed in less than forty-five minutes.

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

For the first three training sessions in each group, the experimenter was present in the room to answer any questions. From session four onwards, presence of the experimenter in the room was not considered necessary although he was still available to answer questions. Rarely did the subjects seek 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.

At the end of training sessions, the subjects were ready to participate in the data collection phase of the experiment. This phase of the experiment is described next.

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. Thus, even the subjects in Groups II and III who were earlier aided by the tutor were unaided during the data collection sessions.

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

Training and Test Problems

Twenty-four problems were identified for use in the experiment. Since the effect of training methods on performance was to be studied for familiar as well as novel situations, five out of the twenty-four problems were exclusively reserved for the data collection phase.

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

In the test sessions, in addition to the five unseen problems, the subjects were given five 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: The 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. The measures used to access the performance of the subjects can be grouped into two categories: product measures and process measures (Henneman and Rouse 1984). Product measures are those measures that access the performance based on the final outcome of the task. Process measures, on the other hand, access performance based upon the process by which the final results were obtained. This section describes the product and process measures used to evaluate the troubleshooter's performance.

Product Measures

Number of problems solved

Successful fault diagnosis 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 ten minutes allocated for each problem in the test sessions.

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.

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

Process Measures

Number of informative actions

A student may take many actions at **Turbinia**'s interface. 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 being investigated. The smaller the number of these relevant informative actions taken to solve the problem, the better is the diagnostic performance. 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.

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 diagnostic 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 incorrect 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 schematics 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 considered the performance.

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 it is a seen problem, 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. In such cases when the student solves the problem, the diagnosis is termed as overdue.

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

Table 6.1 summarizes the product and process measures used to access the performance of subjects in each of the three experimental conditions. The next section describes the anticipated differences in the performance of subjects trained with and without the tutor.

Possible differences in performance

Product measures provide a good criteria to compare the performance of subjects trained with and without the tutor, but they cannot be used to compare the strategies developed by subjects in aided and unaided groups. Process measures, on the other hand, provide means to compare the troubleshooting strategies of the subjects. This section discusses the anticipated differences in the performance of subjects in terms of product and process measures.

Differences in the performance of the subjects, in the three experimental groups, in terms of number of problems solved and the average troubleshooting time can reflect the ability or inability of the tutor to teach subjects to identify the fault in a timely manner. However, since subjects in **Turbinia-Vyasa** are not penalized for wrong diagnosis and there are only a finite number of failure possibilities, it was possible for all subjects to solve the problems within the allocated time. Therefore, substantial differences were not expected in the performance of subjects with respect to the number of problems solved. Also, no definite conclusions concerning the tutor's ability to teach students to identify faults based on this performance measure alone were expected.

However, with respect to the troubleshooting strategies developed by the subjects, it was expected that those trained on the simulator alone would perhaps develop an unguided search strategy to locate the fault. Therefore, they would most likely 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. Also, since they would rarely be using abstract reasoning, they would perhaps solve the problem in less time but with many attempts of incorrect diagnosis. In other words, it was expected

Table 6.1 Summary of performance measures

Performance Measures	Description
Product Measures	
Number of problems solved	number of problems solved out of 10 presented to each subject
Average troubleshooting time	average time taken to solve a problem
Process Measures	
Number of informative actions	average number of diagnostic tests conducted per problem
Percentage of relevant informative actions	percentage of diagnostic tests conducted that were relevant to the failure being investigated
Percentage of guesses	percentage of incorrect diagnosis that are guesses
Instances of investigations in unaffected schematics/subsystems/fluid-paths	number of investigations per problem in schematics, subsystems and fluid paths unaffected by the failure
Nature of diagnosis	proportions of correct diagnoses that are premature, timely and overdue

that those trained with the tutor would perform significantly better in terms of process measures.

It was also possible that those trained on the simulator may learn to map symptoms to faults better than those trained with the tutor because they have less activities to perform and can concentrate more on symptom-cause associations. However. those trained on the simulator were not likely to be able to understand the reasons for many of the abnormal system behaviors. As such, their troubleshooting strategy based simply on associating symptoms to faults, while useful for solving familiar problems was not likely to be of much help in unfamiliar situations. On the other hand, since the subjects trained with the tutor will be taught to reason about failures, their troubleshooting strategy will consist of formulating hypotheses and conducting tests to verify or reject each hypothesis. Such a troubleshooting strategy should prepare those trained with the tutor to cope with unfamiliar situations better than those who only rely on symptom-cause associations. Therefore, it was expected that the performance of those trained with the tutor will be better for unseen problems.

Furthermore, it was anticipated that comparing the performance of those trained with the passive and active tutor might reveal some interesting individual differences in performance. These differences may help understand preferences for learning styles and may suggest something about the instructional strategy used in **Turbinia-Vyasa**.

This concludes a description of the experiment conducted to evaluate the architecture of **Turbinia-Vyasa**, including the measures that were used to compare the performance of subjects trained with and without the tutor. The next chapter discusses the analysis of the data collected from the experiment and the results of the analysis.

CHAPTER VII

EXPERIMENTAL RESULTS

In this chapter, results from the analysis of the experimental data are described. Performance of the subjects from each of the three training groups were compared. First, the effects of the factors considered in the experiment are presented using ANOVA. Next, the results of the analysis are discussed in detail. Finally, the chapter concludes with a discussion of the effectiveness of the proposed architecture and its use in computer-based tutors for diagnostic problem solving in complex dynamic domains.

Analysis of Data

The data were analyzed using the SAS General Linear Model (GLM) and Type III sum of squares. The mixed model used to describe the data consisted of seven sources of variation that included both fixed and random effects (Table 7.1). The estimated mean squares expression computed by SAS for each source of variation is also shown in Table 7.1. Individual ANOVA Tables for the performance measures appear at the end of this chapter.

Analysis of the fixed part of the mixed model involved estimating and testing hypothesis about the fixed effects. Significant effects were detected using an α value of 0.05. Least Significant Difference (LSD) and Duncan's multiple comparison tests (Milliken and Johnson, 1984) were used to compare means for significant effects of the fixed factors.

Even though the number of problems were not equally distributed in the three levels of seen status, the estimated mean square expressions in Table 7.1 show that the coefficients for the same variance term in each expression were almost equal. This indicates that the experimental design was *fairly* well-balanced in spite of unequal distribution of problems in the three levels of seen status. Therefore, the analysis of the fixed factors, which usually depends upon whether the design is balanced or not, was conducted using the method for the

Table 7.1 Expected Mean Squares Expression

Source of Variation	Type III Expected Mean Squares Expression
Cond (Fixed Factor)	Var (error) + 3.086937 Var (Subj*Seen(Cond)) + 10 Var (Cond*Prob(Seen)) + 9.19524309 Var(Subj(Cond)) + Q (Cond, Cond*Seen)
Seen (Fixed Factor)	Var (error) + 3.1 Var (Subj*Seen(Cond)) + 10 Var (Cond*Prob(Seen)) + 30 Var (Prob(Seen)) + Q (Cond, Cond*Seen)
Cond*Seen (Fixed Factor)	Var (error) + 3.1 Var (Subj*Seen(Cond)) + 10 Var (Cond*Prob(Seen)) + Q (Cond*Seen)
Subj(Cond) (Random Factor)	Var (error) + 2.9032258 Var (Subj*Seen(Cond)) + 8.709677 Var (Subj(Cond))
Prob(Seen) (Random Factor)	Var (error) + 10 Var (Cond*Prob(Seen)) + 30 Var (Prob(Seen))
Cond*Prob(Seen) (Random Factor)	Var (error) + 10 Var (Cond*Prob(Seen))
Subj*Seen(Cond) (Random Factor)	Var (error) + 3.1 Var (Subj*Seen(Cond))

balanced case (Milliken and Johnson, 1984). However, even though the experiment was fairly well-balanced, some of the sum of squares generated were not independent. For these sum of squares, test statistics were computed using Satterthwaite's approximation technique (Milliken and Johnson, 1984). Sample computations of test statistics in the analysis of the data are shown in Appendix E.

Analysis of the random part of the mixed model was done by computing the method-of-moments (Milliken and Johnson, 1984) estimates of variance components. Statistical tests were conducted to detect the significance of these variance components.

A discussion of the effects of factors considered in the experiment, beginning with the fixed effects, is provided next. This will be followed by a detailed discussion of the results.

Fixed Effects

The experiment had two fixed factors: training condition and seen status of the problem. The effects of these two factors along with their interaction effect on the performance of the subjects are described next.

Effect of Training Condition

The subjects were exposed to one of the three training conditions: training on simulator (Group I), training with Vyasa in passive mode (Group II), and training with Vyasa in active mode (Group III). The effect of training condition on the performance of the subjects is discussed below. A summary of training condition effect is presented in Table 7.2.

Table 7.2 Summary of Training Condition Effect

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

* Significant at α = 0.1

Number of problems solved

Subjects in Group II solved 95% of the problems while subjects in Groups I and III solved 93% and 88% of the problems respectively (Figure 7.1). Thus, subjects in Group II solved the maximum number of problems followed by subjects in Group I and III. However, the effect of training condition on the number of problems solved was not statistically significant (ANOVA Table 7.4a).

The relatively poor performance of subjects in Group III can be attributed to three factors. First, a single subject was responsible for five of the unsolved problems. Second, subjects in Group III were more inclined to leave a problem unsolved because they were reluctant to guess the failures. Third, the subjects in this group became somewhat dependent on the tutor to solve the problems and when the tutor was withheld from them, during the test sessions, their performance deteriorated.

Troubleshooting time

Subjects in Group I were the quickest to solve the problems with an average troubleshooting time of 2.62 minutes per problem. Those in Group III were the slowest, taking 3.69 minutes. Subjects in Group II were somewhere in between the other two groups with an average time of 3.43 minutes per problem (Figure 7.2).

The troubleshooting time data were not at all surprising considering that the unaided group did not have a guided strategy to solve the problems and relied heavily on guessing. Guessing as opposed to abstract reasoning takes less time. However, at α level of 0.05, the effect of training condition on troubleshooting time was not significant (ANOVA Table 7.4b).

Training Condition Effect

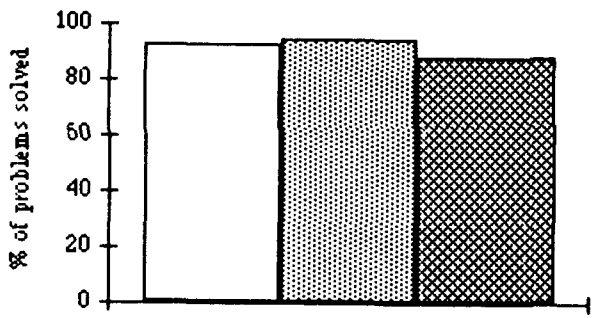


Figure 7.1 Number of problems solved

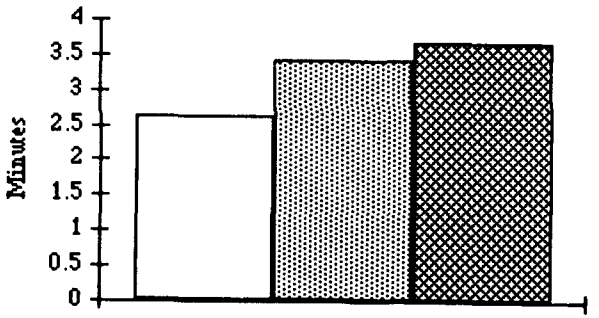
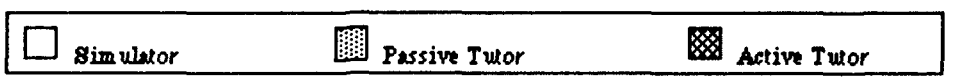


Figure 7.2 Average troubleshooting time per problem



Number of informative actions

Subjects in Group I conducted 10.72 diagnostic tests per problem as compared to 8.18 and 8.83 diagnostic tests per problem conducted by subjects in Groups II and III respectively (Figure 7.3). The difference in the number of informative actions taken by the subjects in aided and unaided groups was, however, statistically not significant (ANOVA Table 7.4c) at α level of 0.05.

Even though the results were statistically not significant the data indicate that the subjects in Group I, in comparison to the subjects in other two groups, needed more diagnostic tests to solve the problems. In other words, subjects in Groups II and III utilized the diagnostic information more effectively and required less number of diagnostic tests to solve the problems.

Percentage of relevant informative actions

Even though subjects in the unaided group conducted more diagnostic tests than subjects in the two aided groups, only 59.7% of their tests were relevant to the failures being investigated. On the other hand, for subjects trained by the passive and active tutors, the percentage of diagnostic tests relevant to the failure was much higher: 72.5% for subjects trained by the passive tutor and 71.5% for subjects trained by the active tutor (Figure 7.4). Of the three groups, the percentage of informative actions that were relevant to the failure was significantly smaller for Group I (ANOVA Table 7.4d).

Since the percentage of relevant informative actions taken by the subjects in the two aided groups was significantly higher, it indicates that those trained by the tutor were better able to identify the diagnostic tests that were useful for solving a problem.

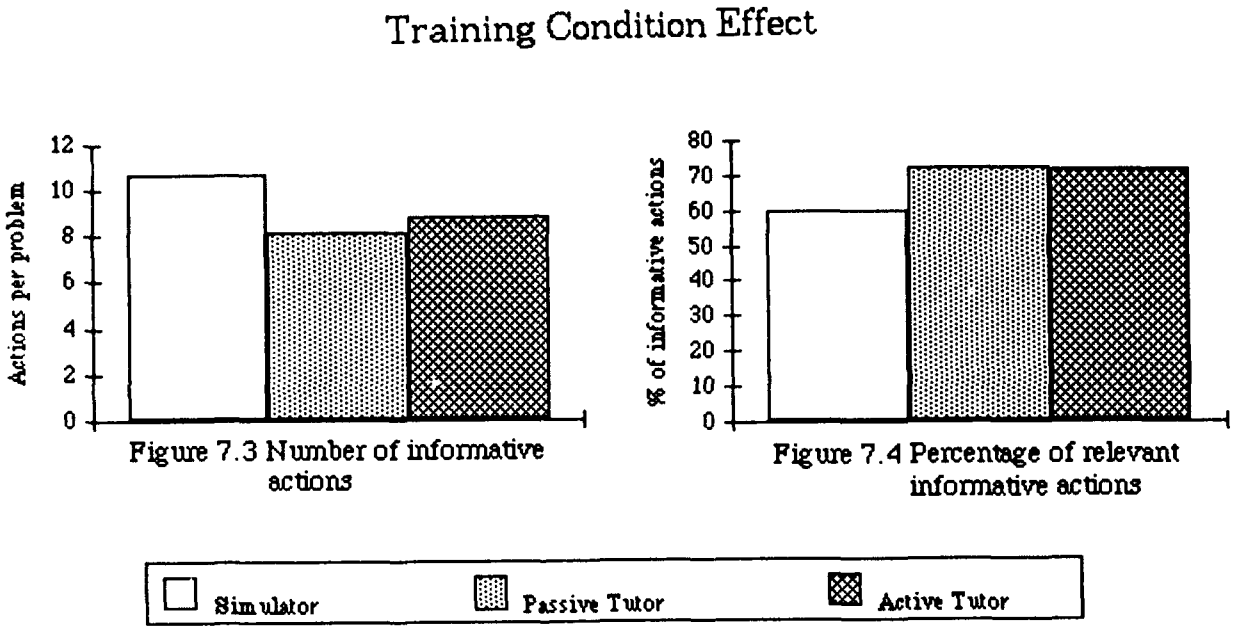


Figure 7.3 Number of informative actions

Figure 7.4 Percentage of relevant informative actions

Percentage of guesses

Subjects in Group I not only performed a large number of incorrect diagnoses but also as many as 70.5% of their incorrect diagnoses were guesses. In comparison, the percentage of incorrect diagnoses that were guesses was much lower for subjects in Groups II and III; 35.23% for subjects in Group II and 29.5% for subjects in Group III (Figure 7.5). The effect of training condition was significant with higher percentage of guesses for the unaided group (ANOVA Table 7.4e) in comparison to the two aided groups.

In addition, evidence of guessing strategy was noticed in 60% of the problems for Group I and only 39% and 30% of the problems for Groups II and III respectively. Also, the data indicate that the subjects in Group I often used guessing as a primary strategy whereas the subjects in Groups II and III started guessing only when they were running out of time.

Number of unaffected schematics/subsystems/fluid-paths investigated

Subjects in Group I provided 0.36 instances of investigations in unaffected schematics per problem as compared to 0.12 and 0.23 provided by subjects in Groups II and III (Figure 7.6). However, at α level of 0.05, the results were not significant (ANOVA Table 7.4f).

Groupwise comparisons of the number of investigations in unaffected subsystems and fluid paths are also shown in Figure 7.6. Analysis of this data shows that subjects in the two aided groups performed significantly fewer investigations in unaffected subsystems and fluid paths (ANOVA Tables 7.4g & 7.4h). In other words, subjects in the two aided groups were able to better identify the location of the fault and investigate the relevant portions of the power plant.

Nature of diagnosis

The usual SAS analysis of variance was not possible for this measure which consisted of comparing the three mutually exclusive categories of correct diagnoses (premature, timely and overdue) from each of the three training groups. Therefore, pairwise comparisons were performed to detect significant differences in the proportion of premature, timely, and overdue diagnoses across the three training conditions (Appendix F). The results of the statistical tests performed indicate that the training condition had a significant effect on the nature of the diagnosis provided by the subjects.

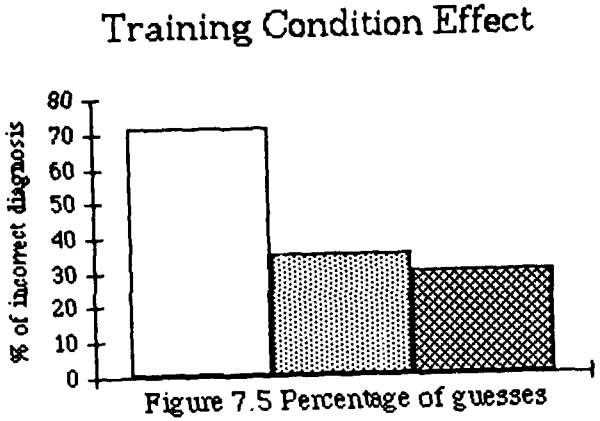


Figure 7.5 Percentage of guesses

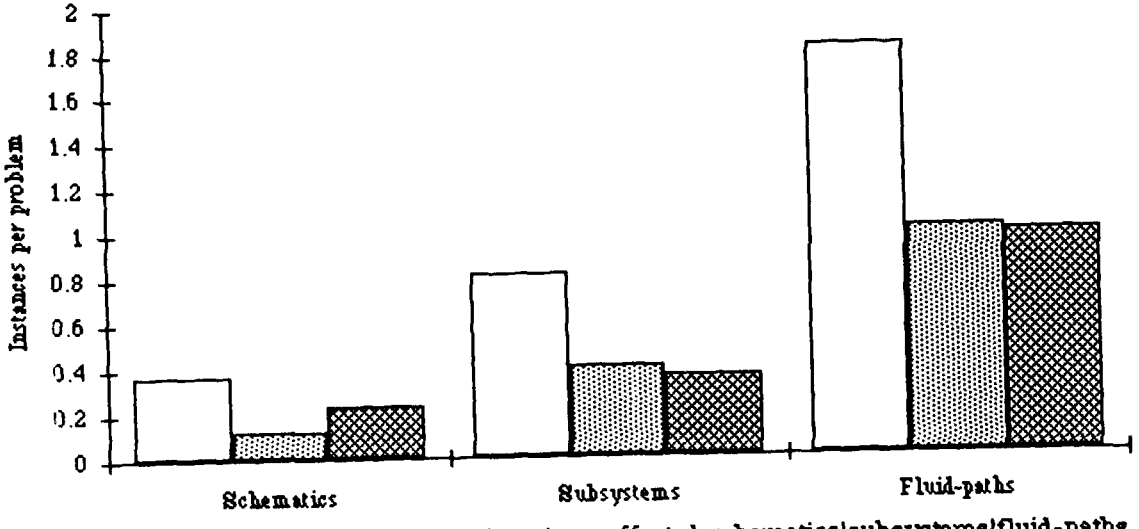
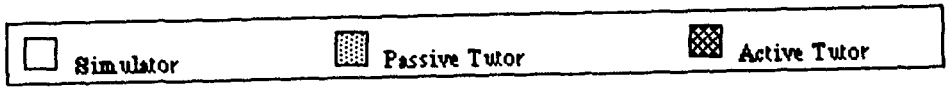


Figure 7.6 Number of investigations in unaffected schematics/subsystems/fluid-paths



Of the problems solved, subjects in Group I performed more premature diagnoses as compared to subjects in Groups II and III. In Group I, 26.8% of the diagnoses were premature. In Groups II and III, the corresponding figures were 14.73% and 9% respectively (Figure 7.7). Statistically, the proportion of premature diagnoses performed by subjects in Group I was significantly more than those performed by subjects in Groups II and III. Since the subjects in Group I relied rather heavily on guessing, it is not surprising that they got lucky more often.

Subjects in Group I solved only 53.7% of the problems in a timely manner. However, for the two aided groups, the number of problems solved in a timely manner was significantly higher: 81% for Group II and 85.22% for Group III (Figure 7.7). The results suggest that the subjects in the two aided groups either formed a better understanding of cause-effect associations or utilized it more effectively to diagnose faults.

Also, for subjects in Group I, more diagnoses were overdue as compared to the two aided groups. The number of overdue diagnoses by subjects in Group I was 19.35% compared to 4.2% and 5.6% for subjects in Groups II and III respectively. (Figure 7.7). This shows that the subjects in Group I were not as good at integrating diagnostic information as the subjects in the two aided groups.

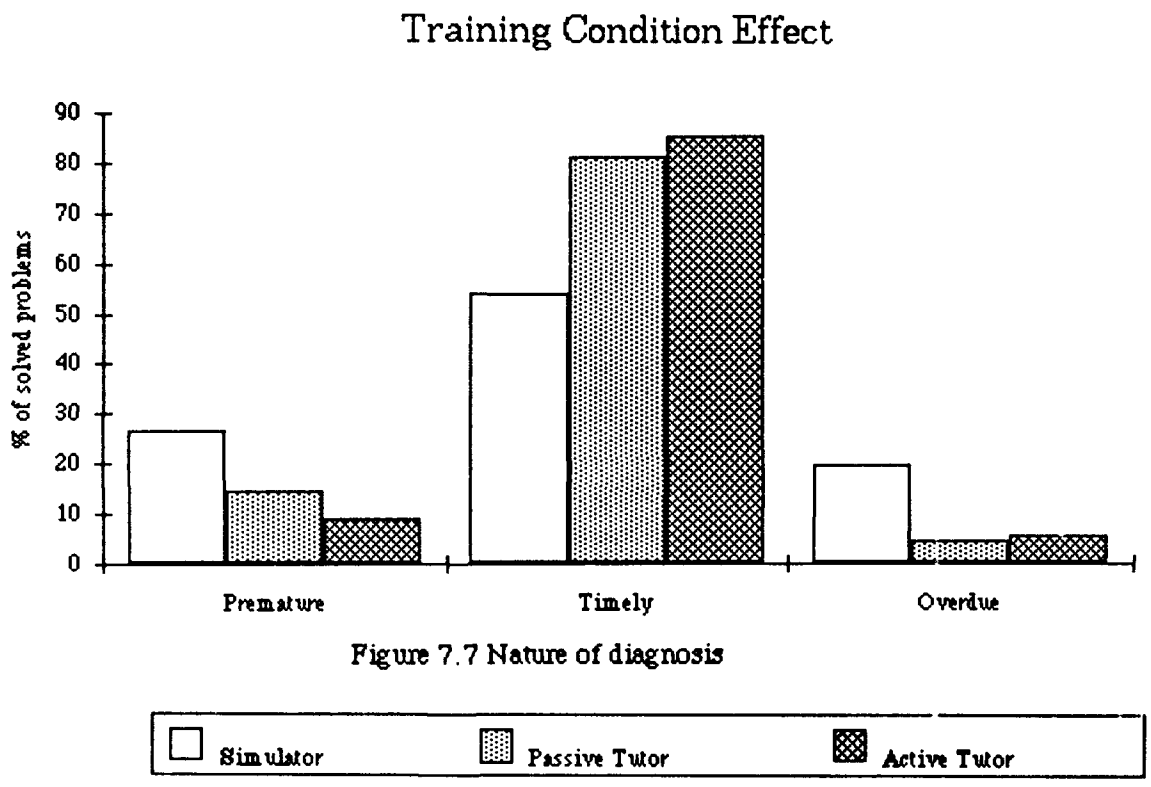


Figure 7.7 Nature of diagnosis

Effect of Seen Status

Of the ten problems presented to each subject in the data collection phase of the experiment, five had been seen earlier by the subjects during training. Two of these five problems were seen once and three were seen twice. Thus, seen status had three levels: unseen, seen once and seen twice. The effect of seen status on the performance is discussed in detail next. Summary of seen status effect is presented in Table 7.3.

Number of problems solved

Subjects were able to solve 95% of the seen problems and 88% of the unseen problems (Figure 7.8). A relatively poor performance of subjects with unseen problems was expected. However, the effect of seen status on the number of problems solved was statistically not significant (ANOVA Table 7.4a).

Troubleshooting time

Subjects were quickest to solve problems seen twice followed by problems seen once and unseen problems. On an average, each problem seen twice was solved in 2.64 minutes as compared to 2.89 minutes for a problem seen once and 3.78 minutes for an unseen problem (Figure 7.9). However, the effect of seen status on the troubleshooting time was also statistically not significant (ANOVA Table 7.4b).

Table 7.3 Summary of Seen Status Effect

	Seen Status				
Performance Measures	Seen Once (S1)	Seen Twice (S2)	Unseen (S0)	Units	Performance Comparison α=0.05
Problems solved	95.00	95.50	88.60	%	Not significant
Troubleshooting time	2.89	2.64	3.78	Minutes	Not significant
Number of informative actions	8.45	8.16	1[illegible]	Actions /problem	Not significant
Percentage of relevant informative actions	67.00	72.40	64.80	%	Not significant
Percentage of guesses	52.20	32.90	52.16	% of incorrect diagnoses	(S2) < (S1), (S0)
Instances of investigations in unaffected					
Schematics	0.30	0.08	0.38	Instances / problem	(S2) < (S1), (S0)
Subsystem	0.48	0.70	0.66		Not significant
Fluid-paths	0.93	1.12	1.48		Not significant
Nature of diagnosis					
Premature	3.50	2.30	32.30	% of solved problems	(S0) > (S1), (S2)
Timely	84.21	89.53	57.89		(S0) < (S1), (S2)
Overdue	12.28	8.13	9.77		Not significant

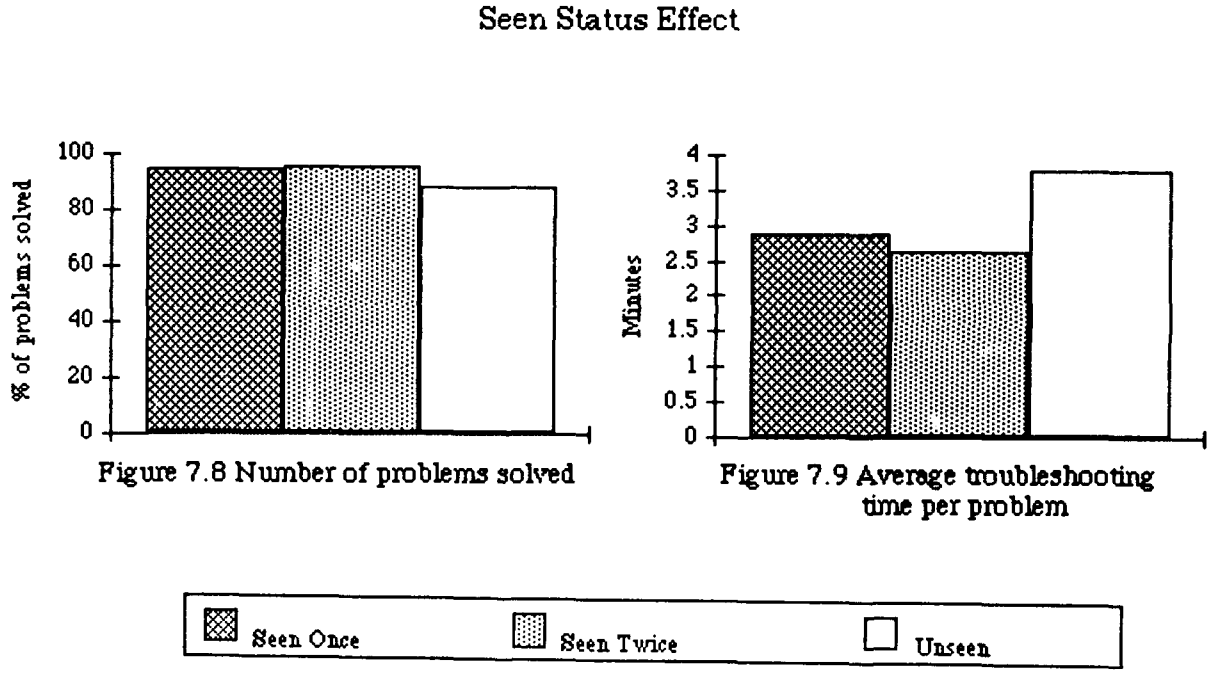


Figure 7.8 Number of problems solved

Figure 7.9 Average troubleshooting time per problem

Number of informative actions

On an average, subjects conducted 8.16 diagnostic tests while solving problems seen twice followed by 8.45 diagnostic tests while solving problems seen once and 10.20 diagnostic tests while solving unseen problems (Figure 7.10). The difference in the number of informative actions taken by the subjects for the three levels of seen status was, however, statistically not significant (ANOVA Table 7.4c).

Percentage of relevant informative actions

For problems seen twice, 72.4% of the diagnostic tests conducted by subjects were relevant. The percentage of relevant diagnostic tests decreased to 67% for problems seen once and dropped further to 64.8% for unseen problems (Figure 7.11). However, the difference in the percentage of informative actions that were relevant to the failure, for all levels of the seen status, was statistically not significant (ANOVA Table 7.4d).

Percentage of guesses

For problems seen twice, only 33% of the incorrect diagnoses were guesses as compared to 52% for each of the other two levels of seen status (Figure 7.12). Also, evidence of guessing strategy was noticed in 29% of the problems seen twice, 33% of the problems seen once, and 57% of unseen problems. Statistically, the percentage of incorrect diagnoses that were guesses was significantly influenced by the seen status of the problem with percentage of guesses being the lowest for problems seen twice (ANOVA Table 7.4e).

Seen Status Effect

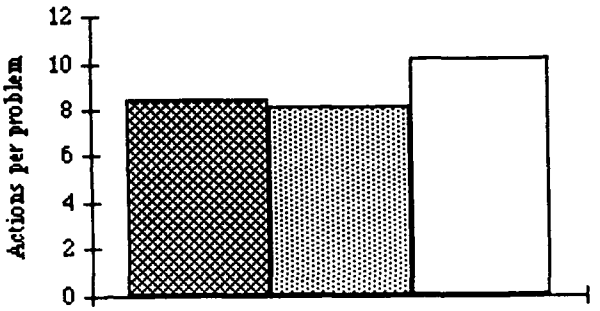


Figure 7.10 Number of informative actions

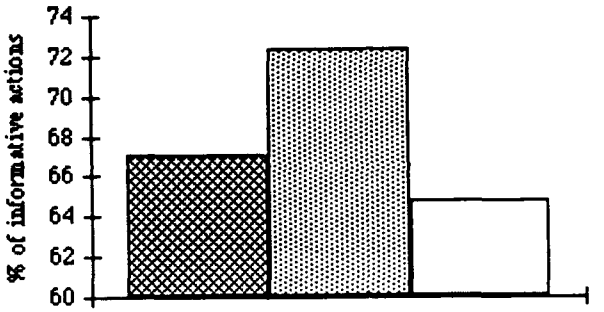


Figure 7.11 Percentage of relevant informative actions

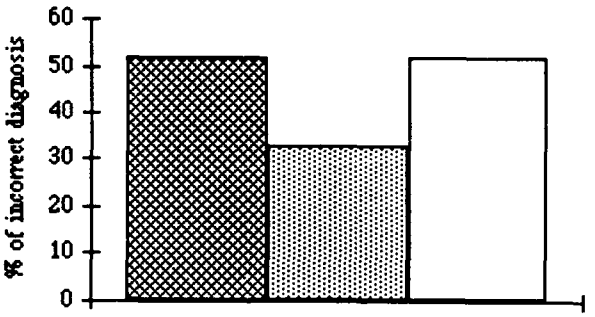
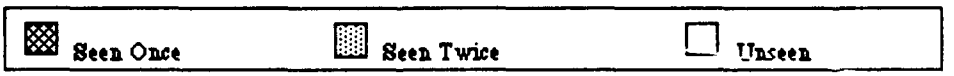


Figure 7.12 Percentage of guesses



Number of unaffected schematics/subsystems/fluid-paths investigated

The effect of seen status on the number of unaffected schematics investigated was significant. Instances of investigations in unaffected schematics were less for seen problems. However, the effect of seen status on the number of unaffected subsystems and fluid paths investigated was not significant (ANOVA Table 7.4f, 7.4g and 7.4h). A comparison of the number of investigations in unaffected schematics, subsystems and fluid paths made for seen and unseen problems is shown in Figure 7.13.

Nature of diagnosis

The analysis for detecting significant effect of seen status was done in a manner similar to the one explained for the training effect in Appendix F. The results indicate that the seen status of the problem had a significant effect on the nature of the diagnosis provided by the subjects. Subjects performed significantly better with problems they had seen during training. In other words, practice and familiarity with the problems influenced the nature of diagnosis provided by the subjects.

Of the problems solved, less than 3% were premature diagnoses with seen problems as compared to 32.3% premature diagnoses with unseen problems. Also, over 87% of the seen problems were solved in a timely manner whereas just about 58% of the diagnoses were timely for the unseen problems (Figure 7.14). However, the difference in the number of overdue diagnoses for seen and unseen problems was statistically not significant.

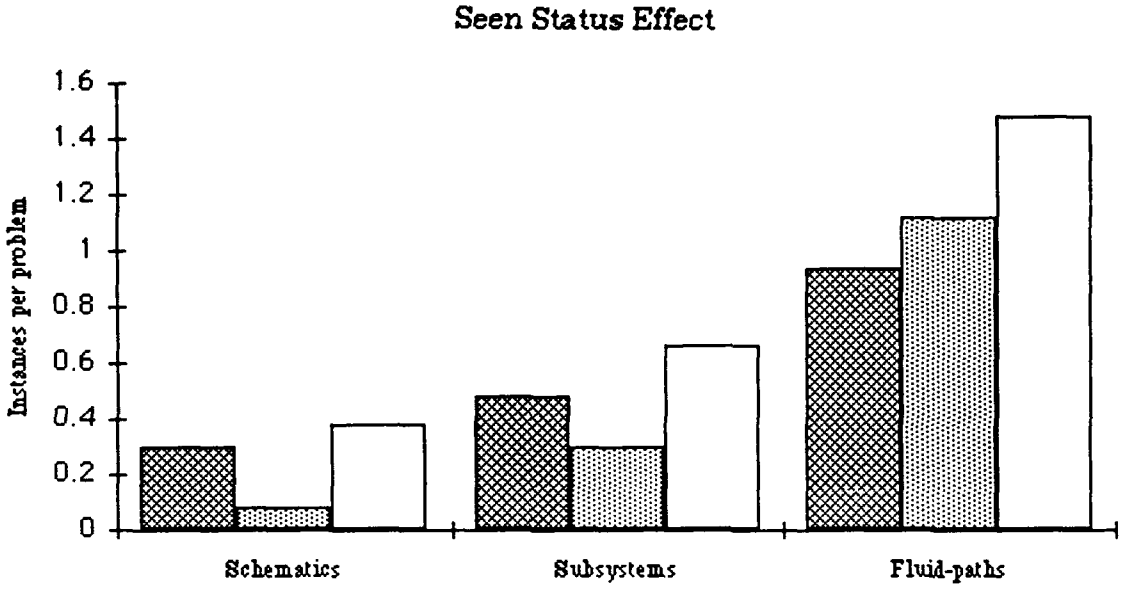


Figure 7.13 Number of investigations in unaffected *schematics/subsystems/fluid-paths*

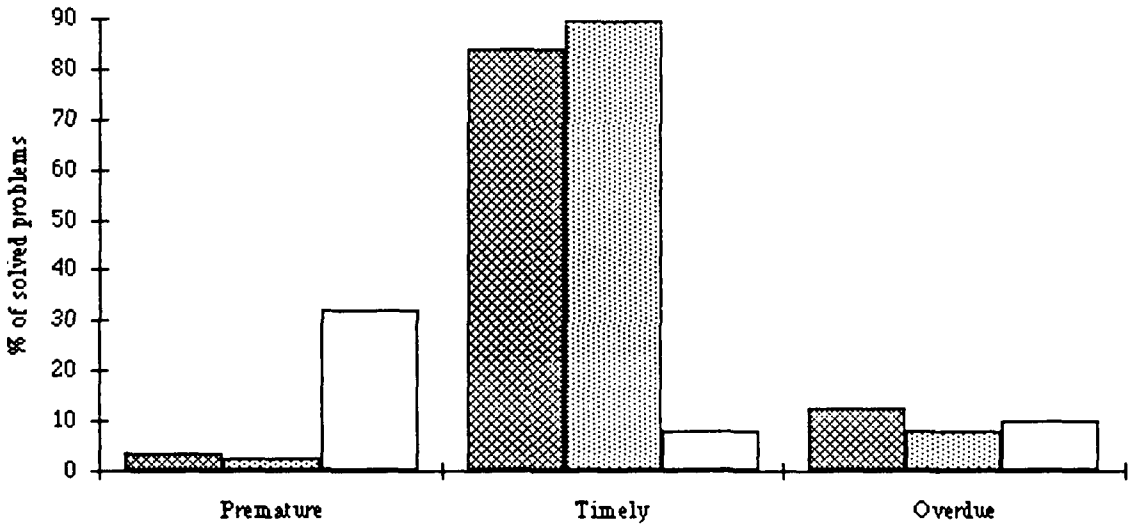
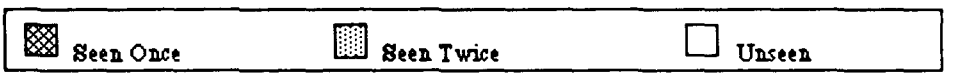


Figure 7.14 Nature of diagnosis



Interaction Effect: Training Condition by Seen Status

Training condition by seen status interaction effect can be seen for each performance measure in Figures 7.15 - 7.25. The data indicate that for most performance measures, the aided groups performed better than the unaided group. Also, there is evidence in the data that suggests that training with the tutor was more helpful in unfamiliar situations. However, the training condition by seen status interaction effect on the performance of subjects was statistically not significant. Some interesting trends observed from the data are discussed below.

First, it was observed that the subjects in Group III, unlike the subjects in the other two groups, were slightly slower at solving problems seen twice than they were at solving problems seen once. The inability of subjects aided by the active tutor to solve the more familiar problems faster can be attributed to lack of practice time. It appears that since these subjects had more activities to perform during the same period of time, they were unable to benefit from practice as much as those in the other two groups. On the other hand, Group I benefited the most from practice and hence solved the more familiar problems fastest. A similar interaction effect was also observed in the number of diagnostic tests conducted per problem by subjects in Group III. The number of diagnostic tests conducted for problems seen twice was slightly higher than the number for problems seen once.

Second, subjects in Group I used guessing strategy in 76% of the unseen problems and in 44% of the seen problems. In comparison, subjects in Group II used the guessing strategy in 54% of the unseen and 24% of the seen problems while the subjects in Group III used it in 42% of the unseen and 18% of the seen problems. Thus, the tendency to guess was sharply reduced by the tutor for both the seen and unseen problems even though it was more prominent with unseen problems in all three training groups. Moreover, the ability of the subjects in the aided groups to solve problems without guessing is evidence that these subjects were better prepared to cope with unfamiliar situations.

Third, subjects in Groups II and III performed fewer premature and overdue diagnosis and more timely diagnosis with unseen problems as compared to subjects in Group I. This is further evidence that training was transferred to unfamiliar situations better with the groups aided by the tutor.

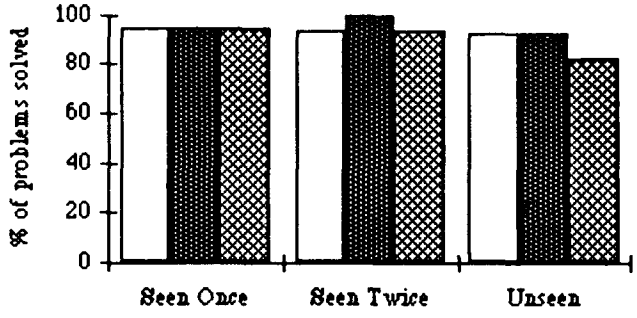


Figure 7.15 Number of problems solved

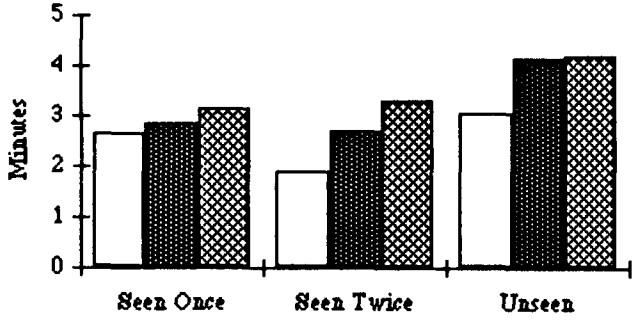


Figure 7.16 Average troubleshooting time per problem

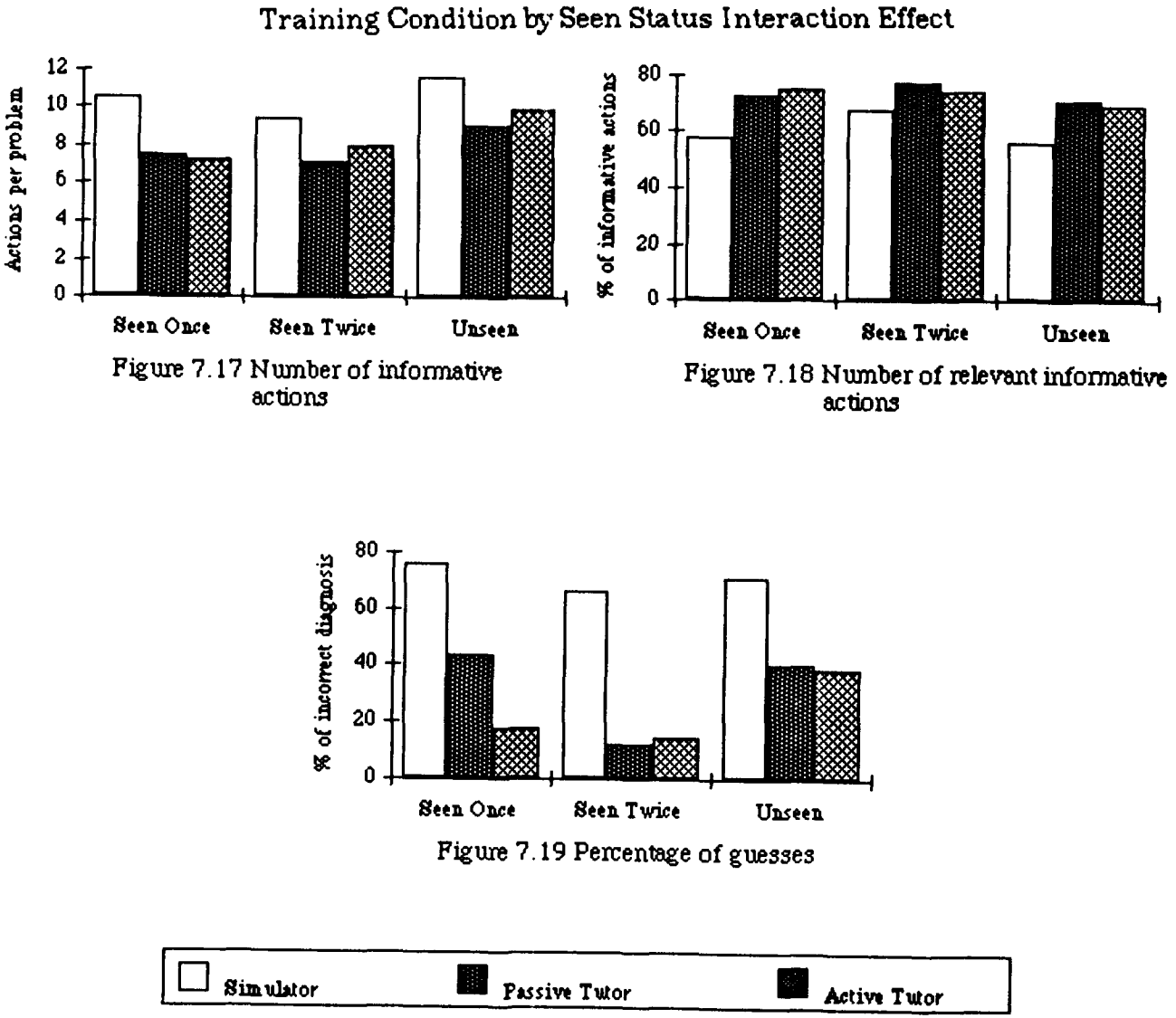


Figure 7.17 Number of informative actions

Figure 7.18 Number of relevant informative actions

Figure 7.19 Percentage of guesses

Training Condition by Seen Status Interaction Effect

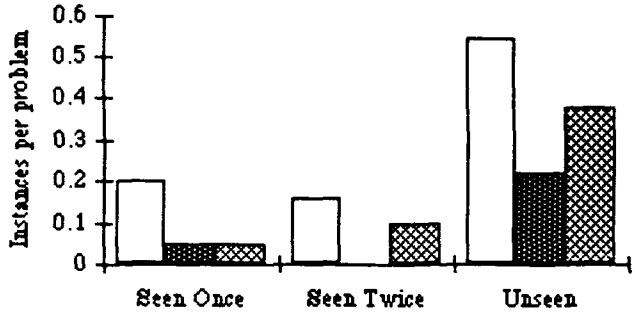


Figure 7.20 Number of instances of investigations in unaffected schematics

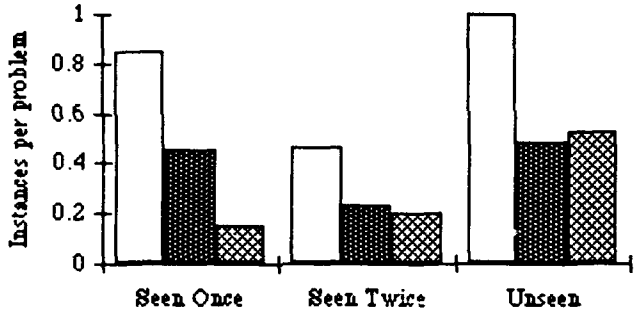
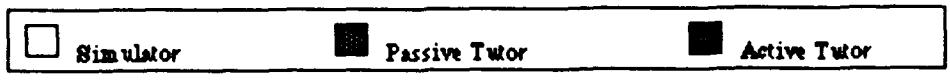


Figure 7.21 Number of instances of investigations in unaffected subsystems

Figure 7.22 Number of instances of investigations in unaffected fluid-paths



Training Condition by Seen Status Interaction Effect

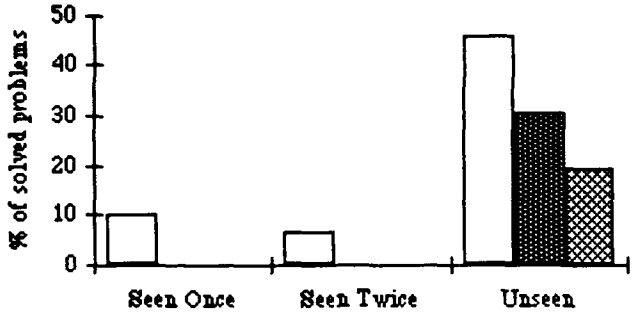


Figure 7.23 Number of premature diagnosis

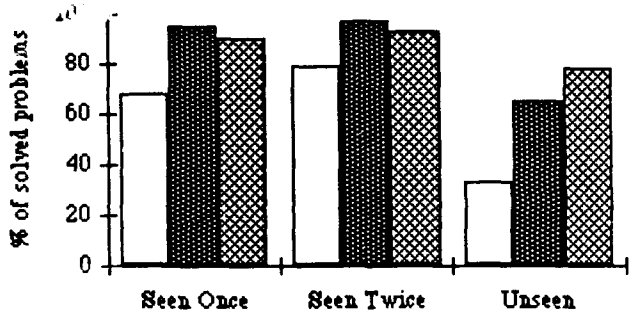
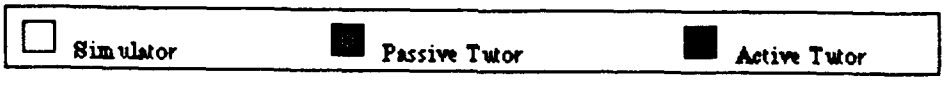


Figure 7.24 Number of timely diagnosis

Figure 7.25 Number of overdue diagnosis



Fourth, subjects in Groups II and III did not have even a single instance of premature diagnosis with seen problems. This suggested that the subjects aided by the tutor, at least for familiar problems, gathered all the relevant diagnostic information before identifying the failed component.

Random Effects

In the experiment, there were two main random effects: subject (nested within training condition) and problem (nested within seen status). Along with these two effects, the effects of following two factor interactions were also analyzed: (1) subject by seen status, and (2) problem by training condition. While variance in the data due to interaction effects was insignificant, variance in most of the performance measure data due to the two main random effects was significant.

Effect of Subject

Subjects that participated in the experiment came from different educational backgrounds with perhaps preferences for different styles of learning. Since their educational backgrounds and preferences for learning styles were not the factors of main interest in the experiment, the experiment was not designed to study the effect of individual differences in the subjects. However, since these subjects (nested within training conditions) were expected to contribute to the variation in data, they were included as sources of variation in the model used for the analysis of data.

It is clear from the analysis of the data that for most performance measures the subject effect was significant. Although the results were not surprising, characteristics of individual subjects responsible for causing significant effects could not be determined by this experiment. However, the data were closely examined to observe individual traits.

It was observed that certain subjects were more inclined to use a guessing strategy than others. While most waited to gather diagnostic information, there were a few that attempted to diagnose the problem based on initial symptom alone and without conducting a single diagnostic test. Even though the guessing strategy was significantly influenced by the training conditions, there were individual in each of the three training groups that indulged in guessing solutions.

There were some subjects in Groups II and III who became very conservative and spent time eliminating all probable causes that could be linked to an observed symptom before identifying the most likely failure. The conservative behavior was probably due to the explanations concerning affected gauges received by these subjects at the end of the session. Perhaps, these explanations made some of the subjects realize that the gauges could be affected in the same manner by multiple failures and hence the cautious behavior.

Also, there were some subjects in Group III who used the active tutor as an on-line associate to solve problems during training. These subjects were not able to solve some of the test problems when the tutor was withheld from them. These students confessed their dependence on the tutor in response to a query in the exit questionnaire.

Effect of Problem

There were ten problems used in the data collection phase of the experiment. Six problems involved a blocked shut mode of failure, two involved a stuck open mode of failure and the remaining two involved a leak out mode of failure. The three modes of failure were equally distributed in the seen and unseen problems.

Although the problems could be categorized depending upon the mode of failure, the extent of difficulty involved in solving these problems was not necessarily dependent of the mode of failure. Difficult as well easy problems could be found in each mode of failure. Since there was no formal method available to measure the degree of difficulty of these problems, categorization of problems depending upon the degree of difficulty was not possible. However, from marine engineers (Marine Safety International, 1983) we understood that not all problems were equally difficult. Therefore, problem (nested within seen status) was considered a possible source of variation in the data and included in the model to be used for the analysis of data.

The significant problem effect was primarily due to the diverse nature of the failures. Failures that affected subsystems with many interactions with other subsystems often had failure effects propagated to large portions of the system. For such failures, large number of abnormal gauge readings could be observed in a short period of time. If initial investigations were focused away from the location of the fault it often led to the formation of wrong hypothesis and many irrelevant diagnostic tests were conducted.

While analyzing the data for problem effect, one set of data was very conspicuous. It was observed that a particular unseen problem was not solved on 11 occasions; three times by subjects in Group I and four times by subjects in Groups II and III This problem involved ruptured boiler tubes. It is considered as one of the more difficult problems by experts. The reason for the difficulty is that the failure effects are rapidly transmitted across many interacting systems. Moreover, the subjects had not seen any failure of this type during the training period and as such were not prepared to deal with it. Therefore, their inability to solve this problem was not at all surprising.

Subjective Evaluation

Along with the analysis of the experimental data, a subjective evaluation of the instructional system was also done using comments made by the subjects and information solicited from them via the three questionnaires.

The comments made by subjects were very encouraging. During training, those trained on just the simulator repeatedly requested for an explanation of the solution and wished they had some means to distinguish the fluid paths in the schematic. They also thought that their ability to diagnose faults would have been strengthened had they been able to find out the function of some components. All these features that subjects in Group I wished for were provided to the subjects using the tutor and they appreciated the help it provided them. Some of the comments provided by the subjects from each of the three groups are reproduced in Appendix G.

It was also observed that subjects in the unaided group used the paper provided to them for comments less often than subjects in the two aided groups. Whereas subjects in the two aided groups used the paper to primarily record their incorrect diagnoses, subjects in the unaided group used it extensively to record the diagnostic tests conducted as well. This shows that the clipboard feature provided by the tutor was a useful aid.

The two identical questionnaires (Questionnaires 1 and 2) filled by subjects at the start and end of the training phase provided useful information about the knowledge acquired by the subjects. More subjects from the aided groups than the unaided group performed better while answering the same questions a second time at the end of training. Since the questions in these questionnaires were based on the material taught by the tutor, the results imply that the tutoring was effective.

The information gathered from the exit questionnaire (Questionnaire 3) was also useful. One of the questions in the questionnaire was related to the same unseen problem concerning ruptured boiler tubes that could not be solved by 11 subjects. The question required subjects to identify the affected gauges and their gauge readings when there was excessive leakage of feed water from boiler tubes. Since subjects, irrespective of their training group, were never given the solutions to the problems during the test sessions, there was no reason to expect different answers from subjects in aided and unaided groups. Moreover, these eleven subjects had no idea that the question they were required to answer was related to a problem they had left unsolved. Therefore, there was also no reason to believe that the subjects, in answering the question, will try to recall the affected gauges from memory.

The answers given by the 11 subjects to this question concerning ruptured boiler tubes provided some useful insight into the troubleshooting knowledge necessary to diagnose faults. While all the eight subjects from the aided groups correctly identified the affected level gauges along with their abnormal readings, only one out of the three subjects from the unaided group could answer the same question correctly. This implied two things. First, the tutor helped the students improve their causal reasoning. This enabled the eight subjects trained with the tutor to consider failure propagation effects and correctly identify the affected gauges and their readings. Second, knowledge of how components fail and how failure effects get propagated is not necessarily adequate to diagnose faults. This explains why, in spite of the causal knowledge these subjects acquired from the tutor, they were unable to apply it for solving the problem.

In addition, Questionnaire 3 helped corroborate the evidence available from the data that some students using the active tutor had become somewhat dependent on it to solve problems. Three students admitted that they had used the active tutor as an on-line associate and that perhaps was responsible for their poor performance when they had to work without it.

Finally, Questionnaire 3 also helped solicit constructive comments from the subjects about the tutor. A sample of some these comments is provided in Appendix H. These comments give us a good idea about the features of the tutor and its interface that were appreciated and those features that were used most often.

Discussion of Results

Analysis of the experimental data reveals that subjects in the unaided group developed a troubleshooting strategy distinctly different from the subjects that were trained using the computer-based tutor Vyasa.

The unaided group did not devise any good and consistent troubleshooting strategy and often relied on unguided search for the cause of the failure. As such, they conducted a large number of diagnostic tests with very few that were relevant to the failure being investigated. In the absence of a guided strategy and in their pursuit for a quick diagnosis, the unaided group performed a large number of incorrect diagnoses and investigated large numbers 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. As such, most of the informative actions they took were relevant to the failure being investigated. In spite of the same motivation as the unaided group to solve the problems fast, the aided groups did not indulge in guesswork. Most of their guesses were a result of panic that set in only 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, the unaided group did it in shorter time, on an average. This was not at all surprising considering that 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, the performance on most measures was better with seen than unseen problems. Among seen problems, those seen twice were often better recalled indicating that practice helped subjects develop symptom-cause associations. But, for Group III, it appears that the practice time was reduced due to other activities and this explained the relatively poor performance of subjects from this group when solving more familiar problems. Both the aided groups, however, 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 less guesses and hence fewer premature diagnoses but they were also able to solve less problems. Most of the variation in the performance between the two aided groups seems to have been caused by individual differences. There were some in Group III (the group with active tutor) that seem to have become somewhat dependent on the tutor and their performance suffered once the tutor was withheld in the test sessions. There were others in both the aided groups that became overly conservative and often investigated and eliminated less likely alternatives as well. While this may not necessarily be a bad troubleshooting approach considering that incorrect diagnoses are costly, any delay in diagnosing the fault can also be costly. Such accuracy-time tradeoffs are more a result of individual preferences rather than the instructional strategy used for training.

Conclusions

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. Those trained by the tutor formed plausible failure hypotheses based on observed symptoms and systematically eliminated them by conducting appropriate diagnostic tests. In comparison, those trained without the tutor did not develop good troubleshooting strategies. They relied rather heavily on guessing the solution. Furthermore, the tutor helped the students to recognize and integrate crucial diagnostic information in a timely manner that the students without the tutor were unable to do. Students trained by the tutor were better prepared for unfamiliar situations than those trained on the simulator.

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

some students as an on-line associate. These students often took the help of the active tutor to refine their failure hypotheses and thus became dependent on the tutor to solve problems. Performance of these students deteriorated when the active tutor was withdrawn.

In summary, results of the experiment show that a simulator alone is inadequate for training purposes. A simulator in conjunction with an effective computer-based tutor can, however, help develop efficient troubleshooting skills. Such a tutor must teach operators to identify useful diagnostic tests, use the results of these tests to formulate plausible hypotheses concerning failure, and systematically refine the hypotheses based on new diagnostic data until the cause of failure is identified. Operators trained by such a tutor are likely to rely less on guessing and more on abstract reasoning. Consequently, these operators are likely to provide incorrect diagnoses less often. In real-world, where there is a cost associated with each incorrect diagnosis, less incorrect diagnoses can save valuable time and reduce troubleshooting costs. However, since not all students are equally receptive to every tutoring strategy, provisions must be made in training programs for individual preferences and differences in abilities and styles. Otherwise, student's may become overly conservative or too dependent on the tutor for help. While conservative behavior may not necessarily be bad, too much dependence on the tutor is undesirable. Therefore, assistance provided by the tutor that directly helps students in solving the problems must be avoided to control the students' dependence on the tutor.

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

This concludes the discussion of the experimental results. The next chapter provides a summary of the results of this research and concludes with recommendations for future research.

ANOVA Tables

Table 7.4a Problems Solved

Source of Variation	df	MS	F	Test Statistic	Adj F	Significant $\alpha = 0.05$
Cond	2	0.0816	1.18		1.18	NO
Seen	2	0.1672	2.42	MSSEEN/ MSPROB	0.39	NO
Cond*Seen	4	0.0405	0.59		0.59	NO
Subj(Cond)	27	0.0768	1.11		1.11	NO
Prob(Seen)	7	0.4303	6.24		6.24	YES
Cond*Prob(Seen)	14	0.0365	0.53		0.53	NO
Subj*Seen(Cond)	54	0.0432	0.63		0.63	NO
Error	189	0.0690				

Table 7.4b Troubleshooting Time

Source of Variation	df	MS	F	Test Statistic	Adj F	Significant $\alpha = 0.05$
Cond	2	27.92	6.02	MSCOND/MSSUBJ	2.71	NO
Seen	2	83.28	17.94	MSSEEN/MSPROB	1.06	NO
Cond*Seen	4	3.30	0.72		0.72	NO
Subj(Cond)	27	10.29	2.21		2.21	YES
Prob(Seen)	7	78.39	16.87		16.87	YES
Cond*Prob(Seen)	14	1.90	0.41		0.41	NO
Subj*Seen(Cond)	54	2.93	0.63		0.63	NO
Error	165	4.64				

Table 7.4c Number of Informative Actions

Source of Variation	df	MS	F	Test Statistic	Adj F	Significant $\alpha = 0.05$
Cond	2	174.37	9.84	MSCOND/MSSUBJ	2.65	NO
Seen	2	113.88	6.43	MSSEEN/MSPROB	0.19	NO
Cond*Seen	4	7.071	0.40		0.40	NO
Subj(Cond)	27	65.76	3.71		3.71	YES
Prob(Seen)	7	579.14	32.68		32.68	YES
Cond*Prob(Seen)	14	7.77	0.44		0.44	NO
Subj*Seen(Cond)	54	18.54	1.05		1.05	NO
Error	189	17.72				

Table 7.4d Percentage of Relevant Informative Actions

Source of Variation	df	MS	F	Test Statistic	Adj F	Significant $\alpha = 0.05$
Cond	2	0.470	9.87	MSCOND/MSSUBJ	3.64	YES
Seen	2	0.257	5.39	MSSEEN/MSPROB	0.927	NO
Cond*Seen	4	0.063	1.31		1.31	NO
Subj(Cond)	27	0.129	2.71		2.71	YES
Prob(Seen)	7	0.277	5.81		5.81	YES
Cond*Prob(Seen)	14	0.036	0.75		0.75	NO
Subj*Seen(Cond)	54	0.032	0.67		0.67	NO
Error	189	0.048				

Table 7.4e Percentage of Guesses

Source of Variation	df	MS	F	Test Statistic	Adj F	Significant $\alpha = 0.05$
Cond	2	1.672	18.09	MSCOND/MSSUBJ	8.4	YES
Seen	2	2.326	25.17	MSSEEN/MSPROB	8.1	YES
Cond*Seen	4	0.070	0.76		0.76	NO
Subj(Cond)	27	0.199	2.16		2.16	YES
Prob(Seen)	7	0.287	3.11		3.11	YES
Cond*Prob(Seen)	14	0.040	0.43		0.43	NO
Subj*Seen(Cond)	54	0.093	1.01		1.01	NO
Error	189	0.092				

Table 7.4f Instances of Investigations in Unaffected Schematics

Source of Variation	df	MS	F	Test Statistic	Adj F	Significant $\alpha = 0.05$
Cond	2	1.10	4.75	MSCOND / (MSSUBJ + MSCOND*PROB - MSERROR)	3.13	NO
Seen	2	3.08	13.30	MSSEEN / MSCOND*PROB	6.90	YES
Cond*Seen	4	0.10	0.42	MSCOND*SEEN / MSCOND*PROB	0.22	NO
Subj(Cond)	27	0.14	0.59		0.59	NO
Prob(Seen)	7	0.91	3.94	MSPROB / MSCOND*PROB	2.04	NO
Cond*Prob(Seen)	14	0.45	1.93		1.93	YES
Subj*Seen(Cond)	54	0.13	0.55		0.55	NO
Error	189	0.23				

Table 7.4g Instances of Investigations in Unaffected Subsystems

Source of Variation	df	MS	F	Test Statistic	Adj F	Significant $\alpha = 0.05$
Cond	2	5.8787	12.88	MSCOND/MSSUBJ	6.74	YES
Seen	2	3.8316	8.38	MSSEEN/MSPROB	0.86	NO
Cond*Seen	4	0.4583	1.00		1.00	NO
Subj(Cond)	27	0.8706	1.91		1.91	YES
Prob(Seen)	7	4.4595	9.77		9.77	YES
Cond*Prob(Seen)	14	0.7590	1.66		1.66	NO
Subj*Seen(Cond)	54	0.3800	0.84		0.84	NO
Error	189	0.4563				

Table 7.4h Instances of Investigations in Unaffected Fluid-paths

Source of Variation	df	MS	F	Test Statistic	Adj F	Significant $\alpha = 0.05$
Cond	2	22.0162	18.88	MSCOND/MSSUBJ	6.61	YES
Seen	2	7.6838	6.59	MSSEEN/MSPROB	0.57	NO
Cond*Seen	4	0.5488	0.47		0.47	NO
Subj(Cond)	27	3.3396	2.86		2.86	YES
Prob(Seen)	7	13.3200	11.43		11.43	YES
Cond*Prob(Seen)	14	0.7160	0.61		0.61	NO
Subj*Seen(Cond)	54	1.2950	1.11		1.11	NO
Error	189	1.1656				

CHAPTER VIII

CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

The research reported in this dissertation is summarized in this chapter. In addition, some implications of this research in the field of science and engineering are discussed. A discussion of future research issues concludes this chapter.

In Chapter I, inadequacies of current training programs for diagnostic problem solving in real-world engineering systems were identified. An intelligent computer-based tutoring system was considered a good alternative to current training programs. Since this alternative was seen to hold promise, issues related to its development became the focus of this research.

In Chapter II, it was observed that in spite of the progress made in the field of intelligent tutoring, not many ideas of existing ITSs have been successfully extended to real-world applications. One of the reasons for the limited success was attributed to the simplicity of the domains and tasks considered by most existing ITSs. Real-world engineering systems, in contrast, are more complex due to their size, interaction between subsystems, and dynamics. In addition, problem solving by human operators in many of the engineering systems is often not well-understood. Therefore, development of computer-based training programs for real-world applications still poses many practical problems.

Furthermore, the few applications that do exist for training operators of engineering systems are not supported by an explicit structured methodology for organizing the large volume of interrelated knowledge associated with these systems. Without such a framework for organizing knowledge, rapid construction of tutoring systems for new domains continues to be a complicated task. Therefore, the current state of the technology needs to be supplemented by a methodology for organizing system and task knowledge in a coherent architecture to facilitate rapid construction of ITSs for complex dynamic systems.

In Chapter III, a framework for organizing the various components of knowledge in an instructional system was proposed to consolidate the current ITS technology. The purpose of the knowledge organization framework was to facilitate quick construction of intelligent tutors for diagnostic problem solving in complex dynamic domains. In order to demonstrate the feasibility of the proposed knowledge organization framework, the framework was first applied to decompose system, task, and pedagogical knowledge for teaching diagnostic problem solving task in a marine power plant domain. Next, the components of knowledge were organized in a coherent ITS architecture to develop a prototype instructional system. Implementation details of this prototype, **Turbinia-Vyasa**, were described in Chapter IV. Details of interaction with this instructional system were described in Chapter V. Finally, an experimental study was conducted to evaluate the architecture of the instructional system. Details of the experiment were described in Chapter VII and its results discussed in Chapter VIII.

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

Implications of the Research

The research reported is a contribution to the field of training for diagnostic problem solving in realistic, complex dynamic domains. It has contributed a methodology to build computer-based training programs that can provide trainees practice and exposure to real-world problems. The research has also developed a vocabulary to comprehend and organize knowledge that is used not only to identify failures but also to categorize modes of failure, recognize typical abnormal system behaviors associated with different modes of failure, and explain cause-effect associations. In addition to these contributions, the research has several implications for science and engineering. Some of these implications are discussed below.

Researchers interested in postulating theories of cognition can use **Turbinia-Vyasa** to study the process of diagnostic problem solving in a realistic environment. For example, differences in the novice and expert problem solving behaviors can be determined by

examining the differences in the troubleshooting strategies used by novices and experts to solve the same problem. Also, the process of transformation from a novice to an expert in terms of changes or modifications in the troubleshooting strategies can be determined over time using **Turbinia-Vyasa**. In addition, differences in troubleshooting strategies for familiar and unfamiliar situations can be examined to find what type of knowledge and experience prepare operators to diagnose unseen faults.

For the AI community, this research demonstrates the applicability of AI techniques beyond the "micro-world" of simple domains. If knowledge is properly organized and its granularity and detail appropriately controlled by the context of the problem solving activity, the volume of knowledge that must be represented to make the ITS effective is manageable. Therefore, there is no reason why further progress in the field of intelligent tutoring should suffer despite the complexities of "real-world" domains.

For those involved case-based reasoning research, this research provides a structured indexing mechanism. The structure-function-behavior model provides a structured taxonomy for effective memory organization which can be a good alternative to some ad hoc methods of indexing often used in case-based knowledge organization. Even for those in cognitive science who study organization of knowledge in long term memory, the vocabulary provided by the structure-function-behavior model can help express and explain memory organization in humans.

From an engineering perspective, this research is expected to motivate the development of tutoring and training systems and implementation not only within the navy but also in other organizations where training is of prime concern and traditional methods are not as effective. The results of this research should encourage the use of computer-based instructional systems to reinforce the current simulator-based training for operators of complex dynamic systems. This should eventually help us develop more effective training programs. Furthermore, the tools provided by this research can help build cost effective prototypes that may be needed to convince people of the benefits of using computer-based training.

Also, **Turbinia-Vyasa** provides engineers with a cost effective way to try new ideas and get a good estimate of the success of a proposed innovation without having to cope with the risk of adversely affecting performance in the real environment. Through trials on **Turbinia-Vyasa**, engineers can design new control room displays for power plants to improve the

performance of supervisory controllers. Furthermore, since most operators are prone to resist changes in their work environment, **Turbinia-Vyasa** can also be used to determine apriori the willingness of operators to accept changes.

In conclusion it must, however, be emphasized that while this research should reduce the complexity of designing and developing intelligent tutoring systems, the task of knowledge acquisition for a new domain will remain a difficult task. The developer must acquire the domain-specific knowledge about the system, the operator's task, and instructional strategies before the design tools proposed here can be used to effectively organize the knowledge for an intelligent tutoring system.

Future Research

In addition to the discussion in the previous section, this research opens up several avenues for future research. Recommendations for future research include both improvements and extensions of current work. Some of the issues that can be explored in future are briefly discussed in this section.

First, the teaching capability of **Turbinia-Vyasa** can be enhanced by providing students the ability to manipulate control devices and observe the effects of their actions. Since the simulated system behavior in **Turbinia-Vyasa** is computed from individual behavior of modeled components, providing the instructional system with this additional capability will require modeling the control devices for more than a single control setting.

Second, the scope of the diagnostic problem solving task taught by **Turbinia-Vyasa** can be increased by refining the simulation methodology to include the ability to simulate multiple failures and situations arising from cascading of failures. Also, since gauges (sensors) and failures in them have not been modeled in **Turbinia-Vyasa**, it excludes a large number of possible real-world scenarios that cannot be considered by the instructional system. Addition of these features in **Turbinia-Vyasa** can make the training environment more realistic.

Third, the scope of training imparted through **Turbinia-Vyasa** can also be extended to include more details of the failure. In engineering systems like power plants, fault diagnosis is seldom confined to detecting the failed component. Instead, it also involves

repair and/or replacement of the component or a part of it. **Turbinia-Vyasa** at present has no knowledge of the parts that make up the components of a power plant. By remodeling components as collection of subcomponents (parts), the tutor can be given the capability to train the mechanics that are actually responsible for repairing the failed unit. In addition, if the tutor provides students the capability to replace suspected components and observe the effects of the change on system behavior, it may add to the student's understanding of failures and help rectify misconceptions.

Fourth, the schematic interface of **Turbinia-Vyasa** can also be improved by using better graphics to represent meaningful objects on the schematic. For instance, engineers have a standard method of graphically representing pumps and turbines. In engineering drawings, both these components are commonly represented by a trapezoid. For pumps that compress gas, the cross section of the trapezoid decreases in the direction of flow. For turbines where gas is expanded, the cross section of the trapezoid increases in the direction of flow. Thus, the trapezoidal shape of pumps and turbines indicates whether the process involved is compression or expansion in the direction of flow. Therefore, changing the shapes of pumps and turbines in accordance with standard engineering practices in **Turbinia-Vyasa**'s schematic interface can help students develop a better mental model of the functions performed by these components. Furthermore, since the technology now exists to incorporate photographic images, students can be shown the structural changes in the components that are responsible for the failure. Use of high resolution graphics and photographs can improve a student's understanding of the failures and their effects on the system behavior.

Fifth, an authoring tool that can elicit knowledge about new failures from even those who are not familiar with the implementation details of **Turbinia-Vyasa** can add to the versatility of the instructional system. In the current version of **Turbinia-Vyasa**, there is provision for expanding the knowledge-base of specific cases of failure but the task of adding knowledge requires understanding the details of implementation.

Finally, the current research provides a strong basis for a research program to develop intelligent operator associates. An intelligent tutor- plus-operator-associate that integrates the functional characteristics of both an aid and an on-line associate can be used with both the training simulators and actual systems. Such a tutor-associate can help aid the human operator during diagnostic problem solving by providing help at appropriate

levels, compensate for routine system failures and help the human develop the expertise necessary for problem solving during system operation.

APPENDIX A

INSTRUCTION MANUALS

OPERATOR INSTRUCTIONS FOR TURBINIA

TURBINIA is a simulated marine power plant. As a naval trainee, you will learn to troubleshoot common failures in a marine power plant using TURBINIA. This instructional manual will describe your interaction with TURBINIA after a brief description of a typical marine power plant and its control.

Introduction

A marine power plant is a collection of components configured to produce mechanical work from thermal energy. This energy transformation takes place in components called the *turbines*. A ship that uses steam as a medium to carry the thermal energy to the turbines is said to be steam-driven. In a steam-driven ship the source of thermal energy is usually fossil or nuclear fuel. This section describes the functioning of a fossil fuel-oil fired, steam-driven marine power plant.

The process of producing mechanical work in a steam-driven marine power plant can be decomposed into several stages. Each stage is associated with one of the four phases in the steam cycle: generation, expansion, condensation and feed.

Steam Generation

Figure 1 shows the configuration of components in the generation phase of the steam cycle. This phase of the steam cycle takes place in the boiler. The boiler is comprised of *tubes* and a *steam drum*. The boiler tubes contain water that is heated by flue-gases resulting from the fuel burned in the *furnace*. This heat transfer is by conduction through the tube walls. Heating of water in the tubes produces steam. This steam accumulates over the water surface in the steam drum and is called *saturated steam*. Saturated steam is sometimes also referred to as wet steam because of its moisture content.

Continuous steam generation in the boiler increases the steam pressure in the drum. Boilers are rated by the steam pressure they can handle in the drum. In a 1200-psig boiler, for example, the maximum steam pressure permitted is 1200-psig. A safety valve is activated to release pressure whenever it exceeds the maximum value.

The steam pressure in the drum controls the temperature at which the water boils in the drum. Since the temperature of saturated steam accumulating above the water surface is the same as the temperature of the water, the saturated steam temperature depends upon the steam pressure. This temperature at which the water and saturated steam coexist in the drum is called the saturation temperature. The highest possible saturation temperature is attained in a boiler when the boiler operates at its maximum rated pressure. Since the thermal energy of steam in the drum is proportional to its saturation temperature, the heat content of the saturated steam is maximum at the highest boiler operating pressure.

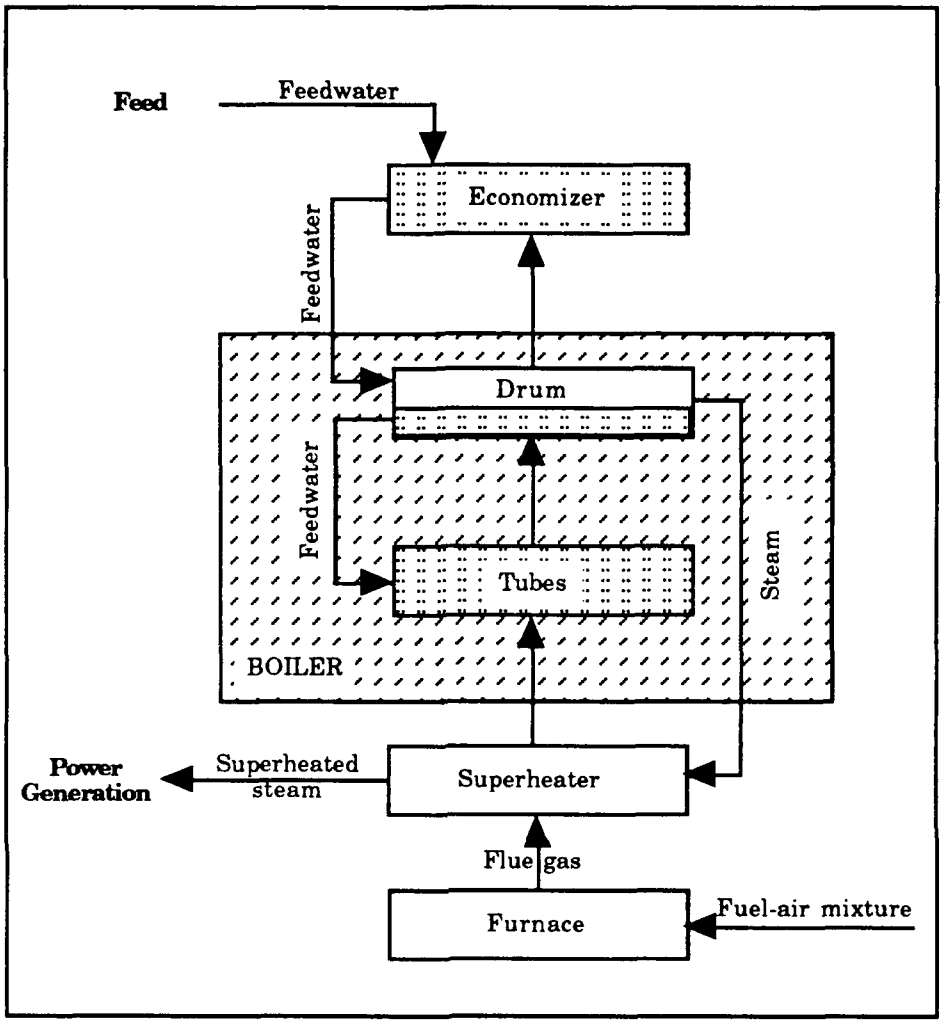


Figure 1. Steam Generation

Even though the boilers are designed for high operating pressures, the saturated steam does not contain enough thermal energy to operate the turbines at their best efficiency. Thermal energy of steam is increased by passing it through tubes in the section of the boiler closest to the furnace. This section of the boiler is commonly known as the *superheater*. The superheater is responsible for adding heat to saturated steam at constant pressure. The heat added to saturated steam in the superheater is called sensible heat. Sensible heat increases the temperature of the steam beyond the saturation temperature and makes it drier. The steam from the superheater is called superheated steam and the increase in steam temperature in the superheater measures the degree of superheat.

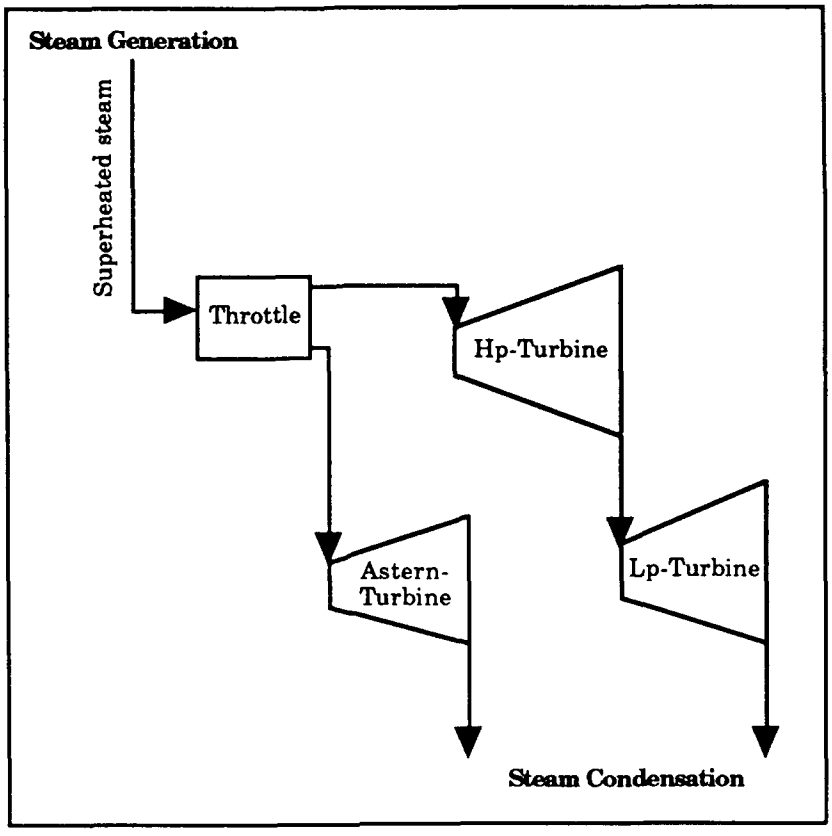


Figure 2. Steam Expansion

Steam Expansion

The second phase of the steam cycle takes place in two steps. First, the superheated steam from the boiler expands in a *high pressure turbine* to convert thermal energy to mechanical work. Then, since the steam still contains a considerable amount of thermal energy, it is expanded further in a *low pressure turbine* connected to the exhaust of the high pressure turbine. Figure 2 shows the arrangement of low and high pressure turbines in a power plant.

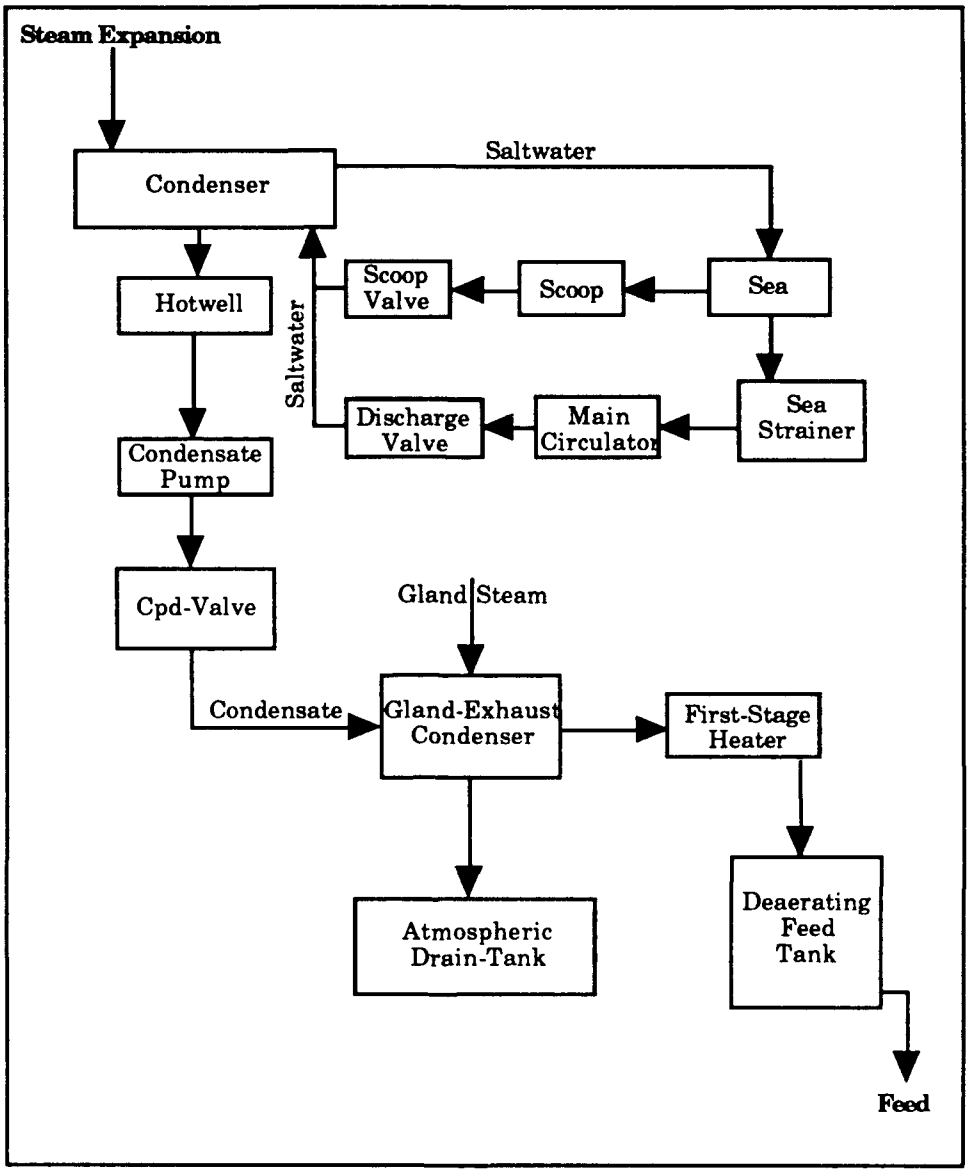


Figure 3. Steam Condensation

Steam Condensation

After expansion, the third phase of the steam cycle is steam condensation which takes place in the main *condenser* (Figure 3). The condenser is a sealed container with tubes that carry cold sea water. When the steam passes over these tubes it loses latent heat to the cold water. When sufficient latent heat is withdrawn from the steam, it changes phase and turns back into water, called *condensate*.

Steam pressure at the turbine exit is low and steam can flow into the main condenser only if the pressure in the condenser is lower. Since the condensate occupies less volume than the same amount of steam and because the condenser is a sealed container, condensation

creates a vacuum in the condenser shell. This vacuum in the condenser shell helps maintain a continuous flow of steam from the turbines to the condenser.

As the steam from the turbines turns into condensate, it flows into a collecting tank called the *hotwell*. The *condensate-pump* then pumps the condensate to the *deaerating-feed-tank* via the *gland-exhaust-condenser*. In the gland-exhaust-condenser, the condensate from hotwell serves as the cooling medium to condense steam from the turbine glands. While the condensate from the gland-exhaust-condenser flows to the deaerating-feed-tank, the condensed gland steam is returned to the condensate system by way of *atmospheric-drain-tank*.

Feed

Feed, the last phase of the steam cycle, begins at the deaerating-feed-tank. The deaerating-feed-tank is a storage tank for feedwater. It also contains apparatus to remove dissolved oxygen entrained in the condensate. The other major components in the feed phase are the main *feed-pump*, the *feed-water-regulator* and the *economizer*. These components are shown in Figure 4. The main feed pump is responsible for pumping water to the boiler. The feed-water-regulator regulates feedwater into the economizer enroute to the boiler. The economizer is a heat exchanger that preheats the feedwater.

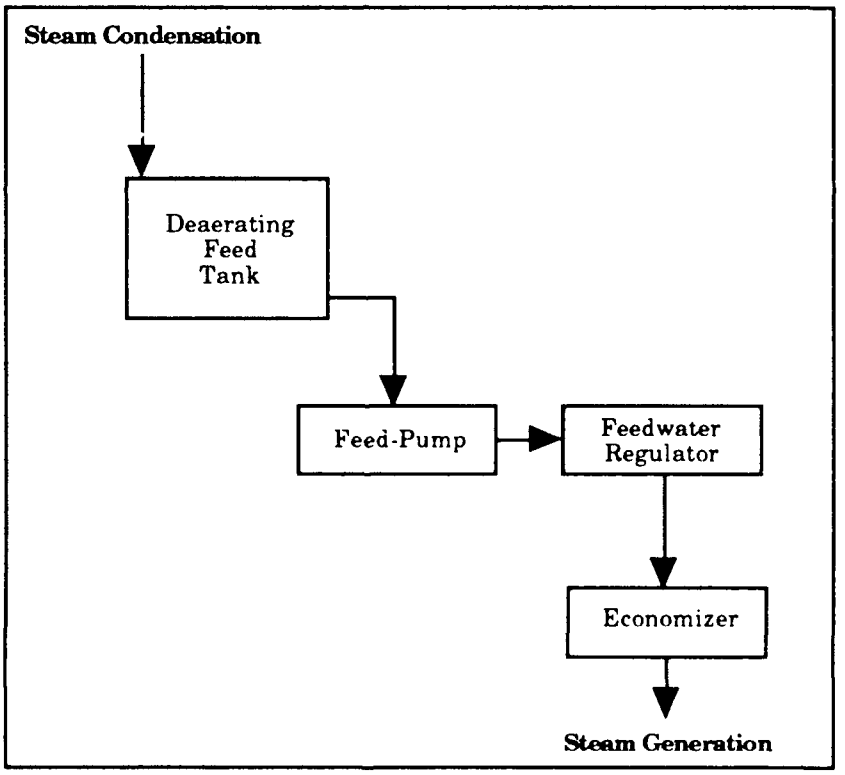


Figure 4. Feed

Each of the four phases of the steam cycle perform an important system function. The collection of components responsible for the function constitute a functional subsystem. Thus, **steam generation**, **steam expansion** (or **power generation**), **steam condensation**, and **feedwater preheating** are also essential *subsystems* of a marine power plant. In addition, a

power plant typically has subsystems that perform other functions necessary for its operation. **Combustion, auxiliary steam use, control air, lubrication,** and **saltwater service** are examples of such subsystems.

Combustion involves burning the fuel-air mixture prepared in the *burner*. The thermal energy released during combustion is used to heat water in the boiler. The components that make up the combustion subsystem are shown in Figure 5. These components lie along two fluid paths: *combustion air* and *fuel-oil*.

Combustion air is supplied to the burner by a forced draft fan operated by either a steam turbine or an electric motor. Fuel-oil is supplied to the burner by pumping fuel from a *settling-tank*. For proper combustion, both the combustion air and the fuel-oil need to be at the proper pressure and temperature. Furthermore, for complete combustion the mass of air required is fourteen times the mass of fuel-oil.

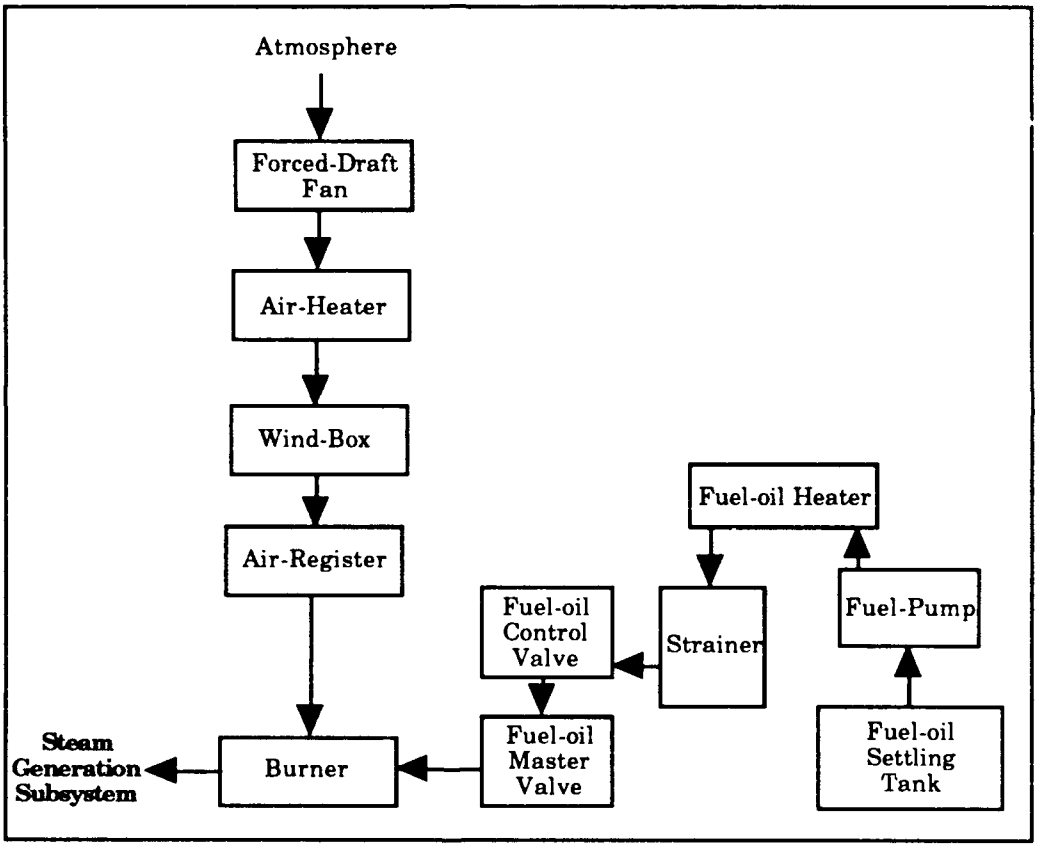


Figure 5. Combustion Subsystem

Incorrect quantity or improper heating of either the combustion air or fuel-oil causes combustion problems. Inadequate quantity or insufficient heating of combustion air and excessive flow of fuel-oil or insufficient heating of it causes incomplete combustion. Incomplete combustion causes dark smoke in the boilers. On the other hand, excess quantity of combustion air in the fuel-air mixture either due to increased flow rate of air or reduced flow rate of fuel-oil extinguishes the flame in the furnace. Excessive preheating of either the combustion air or the fuel-oil causes yet another combustion problem called preignition. In preignition, the fuel starts to burn before it reaches the burner.

The auxiliary steam use subsystem shown in Figure 6 uses *desuperheated steam* for various purposes. Desuperheated steam, unlike the superheated steam, is low pressure steam. Desuperheated steam is obtained by passing superheated steam through the *desuperheater*. Low pressure desuperheated steam is used (1) by the auxiliary power units to run equipment such as the *feedwater-pump*, *fuel-pump*, *saltwater-service-pump* and the *forced-draft -fan*; (2) to preheat the fuel-oil and the feedwater in the deaerating-feed-tank; and (3) by the the gland seals to prevent leakage of air into the turbine casings and steam leakage out of the casings.

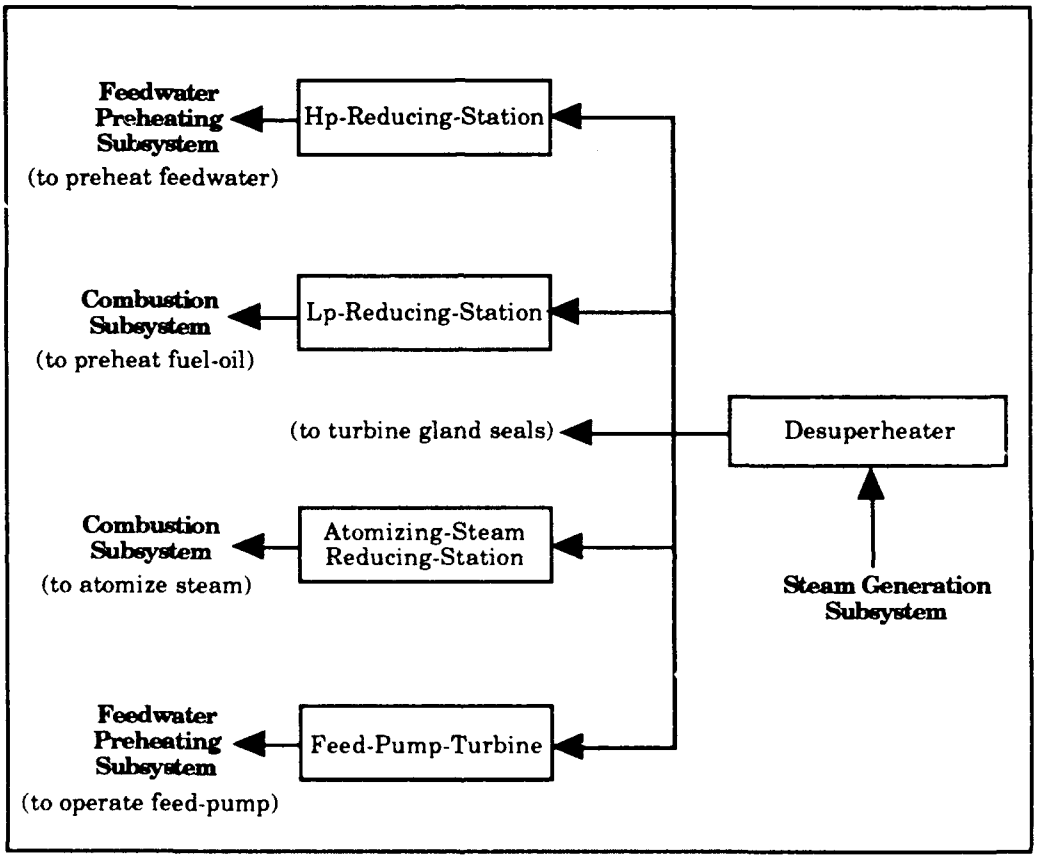


Figure 6. Auxiliary Steam Use Subsystem

The control air subsystem is responsible for distributing control air to many valves and regulators. These valves and regulators are operated by the control air. Figure 7 shows the components that constitute the control air subsystem of a marine power plant.

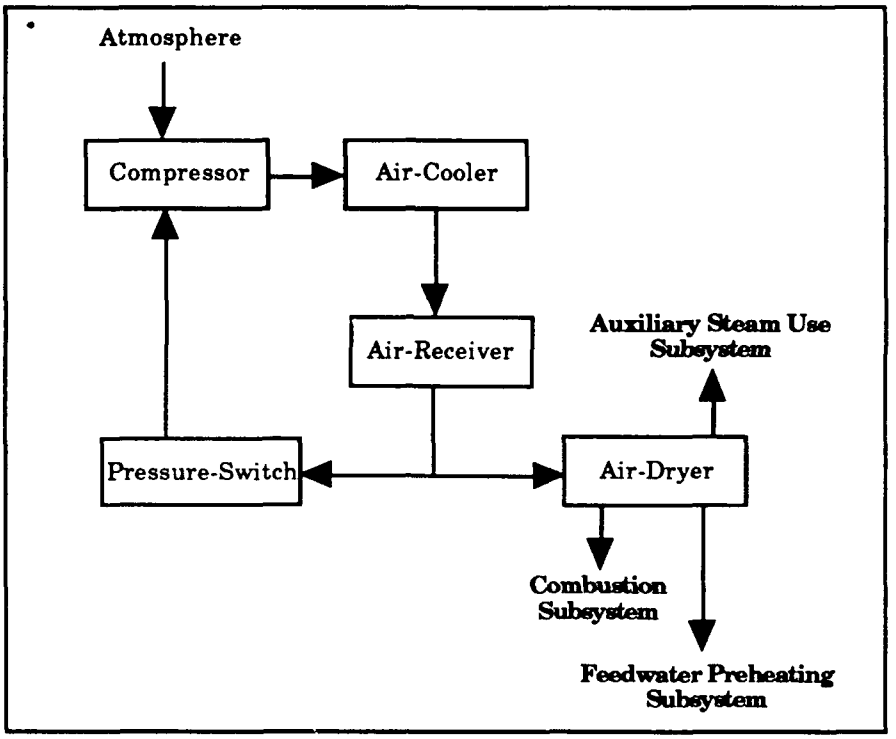


Figure 7. Control Air Subsystem

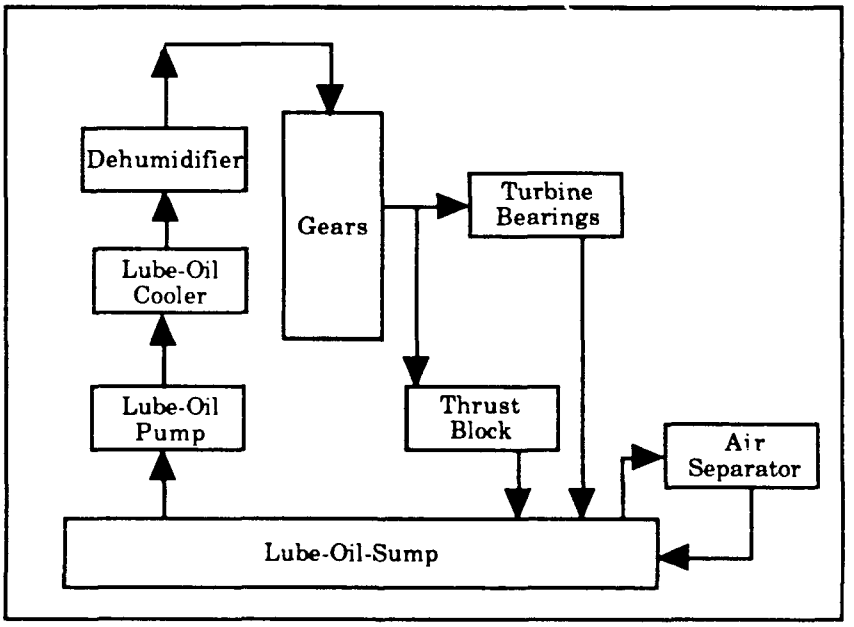


Figure 8. Lubrication Subsystem

The lubrication subsystem has the primary purpose of lubricating moving parts and removing the heat produced by friction. The subsystem consists of a pump that draws lube-oil from the *oil-sump* and distributes it to those components that need lubrication. Figure 8 shows the components in the lubrication subsystem.

The saltwater service subsystem distributes cold sea water to remove heat from units dissipating heat. Figure 9 shows sections of the power plant cooled by saltwater.

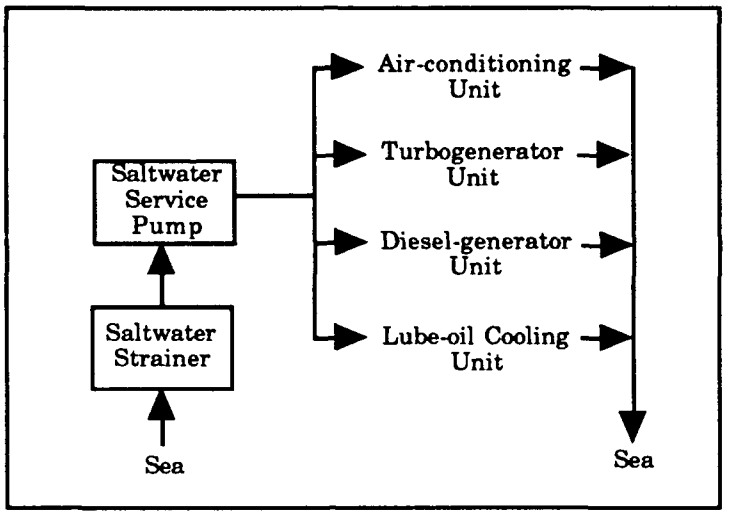


Figure 9. Saltwater Service Subsystem

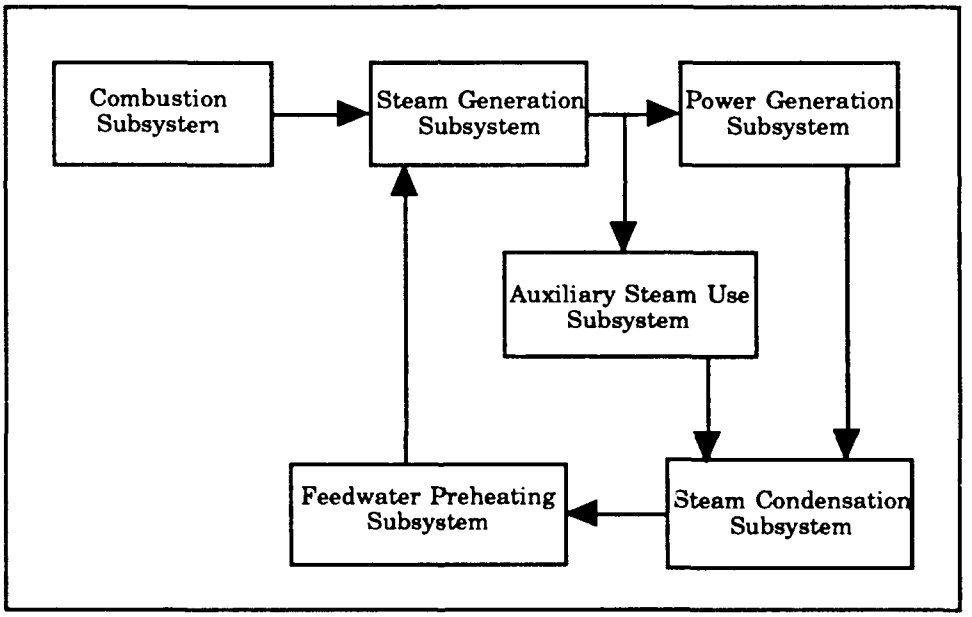


Figure 10. Interacting Subsystems of Marine Power Plant

This section described the decomposition of a marine power plant into nine functional subsystems. It also described the role each subsystem plays in achieving the overall goal of producing power. The interaction between the subsystems to produce power is summarized in Figure 10. For constant power supply, the operating conditions for these subsystems can be set to safely meet the demand. However, the demand for power in a ship is never constant but varies with load. Control systems manage the power plant so that it can satisfy the changes in the demand for power due to fluctuating load. The boiler control system is the most important among all control systems in a marine power plant. The boiler control system of most modern Navy vessels is sophisticated and needs minimum human

intervention. Some components of the automatic boiler control system (ABC) are described next.

Automatic Boiler Control System

Navy vessels typically have the following three ABC systems: automatic combustion control (ACC), feedwater control (FWC), and makeup and excess feed control systems. These control systems perform the functions of measuring, comparing, computing and correcting. In each control system, a state value of interest is measured; compared to a desired value; a new operating condition, if necessary, is computed; and finally a correction made in the operating conditions to reduce the deviation between the measured and the desired value of the state.

Automatic Combustion Control System. The function of the automatic combustion control system is to maintain the boiler drum pressure at a constant value during steady and changing load conditions. The ACC system accomplishes this task by

(1) constantly measuring the steam drum pressure and combustion air flow;
(2) comparing the steam drum pressure to the specified designed value;
(3) computing the amount of change, if any, in the furnace combustion; and
(4) correcting furnace combustion as needed.

When the steam demand on the boiler is increased, the steam drum pressure decreases because the rate of steam withdrawal from the drum becomes greater than the rate of steam production in the boiler. This pressure drop is sensed by the ACC system and an increase in furnace combustion is computed to meet the increase in the demand for steam. Computing the increase in furnace combustion involves computing the increase in the supply of combustion air and a proportionate increase in the supply of fuel-oil to assure complete combustion. The ACC system controls the combustion air flow by regulating the supply of steam to the forced draft fan turbine and controls the fuel-oil flow by positioning the main-fuel-oil-control-valve. The measurement of air flow provides the ACC system with the feedback necessary to perform this function.

Feedwater Control System. The function of the feedwater control system is to maintain a constant water level in the steam drum. The FWC system automatically does this by

(1) measuring the steam drum water level and the feedwater flow rate to the boiler;
(2) comparing the measured water level in the drum to a designed value;
(3) computing the required change, if any, to the rate of feedwater flow; and
(4) correcting the feedwater flow rate as needed.

When the load is steady, the feedwater flow rate into the boiler equals the rate of steam consumption and the water level in the steam drum is normal. But, when the load changes, so does the demand for steam. Any change in this demand is detected and the feedwater flow rate is increased or decreased to equal the steam flow rate out of the boiler. The actual control of feedwater flow is accomplished by adjusting the air-operated diaphragm of the feedwater regulator between the feed pump and the boiler.

Makeup and Excess Feed Control System. Operation of a steam-driven power plant often requires the addition or removal of water from the steam cycle. The makeup and excess feed control system is responsible for doing this and for maintaining a specified level of feedwater in the deaerating-feed-tank.

Whenever the level in the deaerating-feed-tank deviates from the specified value, water is either withdrawn from or added to the deaerating-feed-tank. In both cases the process is facilitated by two standby tanks. The two standby tanks are the atmospheric-drain-tank

and the distillate-tank. When the feedwater level in the deaerating-feed-tank falls below normal, the *makeup-feed-regulator* is adjusted by the control system to increase flow from the standby tanks. Increased flow into the deaerating-feed-tank compensates for the loss in the feedwater level. Similarly, a *deaerating-dump-regulator* is activated by the control system to withdraw excess feedwater from the deaerating-feed-tank when the level in the tank rises above the normal value.

In addition to the automatic boiler control system, a power plant has several other controls which are not discussed here because they are not relevant to your task. However, knowledge concerning some common modes of failure in components of a power plant is useful for diagnosing faults and is described next.

Common Modes of Failure

A mechanical component in a physical system like the marine power plant can fail in more than one way. The four most common modes of failure for components of TURBINIA are: (a) blocked-shut, (b) stuck-open, (c) leak-in, and (d) leak-out. Faults in components fit one or more of these four mode types.

A blocked-shut component offers greater than normal resistance to the flow of fluid for the desired operating condition. A valve that cannot position its vane to a larger opening demanded by the new operating condition or a strainer that is clogged with dirt are examples of the blocked-shut mode of failure.

A stuck-open component offers less than normal resistance to the flow of fluid for the desired operating condition. A valve that refuses to position its vane to a smaller opening on command is an example of the stuck-open mode of failure.

A component failed in leak-in mode allows undesirable or excess flow of fluid *into* it, while a leak-out mode of failure causes undesirable passage of fluid *out* of the component. A vacuum tank that allows air to leak in from outside and a ruptured piping that allows the fluid it carries to leak out from it are examples of leak-in and leak-out modes of failure respectively.

Each failure mode is responsible for a system behavior that manifests in the form of a typical pattern of abnormal state values. During diagnostic problem solving, it is often helpful to identify the failure mode from system behavior and confine the search to components that fail in the identified mode. The typical system behavior associated with a fault also depends upon the phase of the fluid in the affected path. The following set of examples explains the abnormal system behavior for each of the four modes of failure in liquid and gas paths.

A blocked-shut mode of failure in a liquid path causes the liquid *level* downstream to be lower than normal and the level upstream higher than normal. A similar blocked-shut mode of failure in a gas path, on the other hand, decreases the downstream gas *pressure* and increases the upstream pressure.

A stuck-open mode of failure in a liquid path causes the liquid *level* downstream to be higher than normal and the level upstream lower than normal. A similar stuck-open mode of failure in the gas path increases the downstream gas *pressure* and decreases the upstream pressure.

When a container that stores liquid allows more of the same liquid to leak in, the *level* of the liquid in the container increases. When the same container stores gas and allows more of it to leak in from the high pressure surroundings, the *pressure* in the container becomes abnormally higher.

A ruptured component that allows liquid to leak out causes a drop in the liquid *level* upstream as well as downstream from the place of leakage. A similarly ruptured component carrying gas causes a drop in *pressure* upstream and downstream from the place of leakage.

Although there is a typical system behavior associated with each mode of failure, it is not always easy to observe the abnormal behavior in a real system. This is due to the limited number of available gauges. Therefore, pressures, temperatures, and flows cannot be measured across every component. Furthermore, certain components can prevent propagation of expected abnormal behavior past them. For instance, a source-sink such as a deaerating-feed-tank located downstream in the blocked-shut condensate path prevents further propagation of low level downstream from the tank. The deaerating-feed-tank imposes such a behavior on the system because it is an infinite source of feedwater which temporarily compensates for any loss of water level. A summary of typical system behavior associated with the four failure modes is shown in Table 1.

Failure Mode	Fluid	State	Abnormal Behavior Upstream	Abnormal Behavior Downstream	Propagation Limited By Upstream	Propagation Limited By Downstream
Blocked-Shut	Liquid	Level	High	Low	Infinite Sink	Infinite Source
	Gas	Pressure	High	Low	Safety Valve	Infinite Source
Stuck-Open	Liquid	Level	Low	High	Infinite Source	Infinite Sink
	Gas	Pressure	Low	High	Infinite Source	Safety Valve
Leak-In	Liquid	Level	High	High	Infinite Sink	Infinite Sink
	Gas	Pressure	High	High	Safety Valve	Safety Valve
Leak-Out	Liquid	Level	Low	Low	Infinite Source	
	Gas	Pressure	Low	Low		

Table 1. Typical Abnormal System Behavior

This completes a description of a typical marine power plant, its control systems, the four common modes of failure in components of the power plant, and the typical abnormal system behavior associated with each of the four failure modes. The next section describes TURBINIA's interface.

The Interface

TURBINIA, the marine power plant simulator has been developed on a dual screen Apple Macintosh II workstation. The dual screen configuration consists of one 19" color monitor and a 13" color monitor. In this set up, the larger monitor is the left screen and the smaller monitor is the right screen. A single button computer mouse that can point to all locations on both screens is the only input device. You will use this mouse to interact with the direct manipulation interface of TURBINIA. Almost all your actions involve moving the mouse cursor to a desired location and clicking on the mouse button. All valid user actions have appropriate response while invalid actions are ignored by the system. Valid actions at TURBINIA's interface are described in detail later.

TURBINIA's interface consists of seven schematic windows, a schematic menu, a requests menu, a symptom display dialog, a communication dialog, and several error dialogs.

The seven *schematics* display the physical connections between the components of the power plant. You will use these schematics to investigate components and probe gauges attached to these components.

The *schematic menu* displays seven icons each representing one of the seven schematics. You will use these icons to access the schematics.

The *requests menu* has three icons. You will use the first icon to request for an opportunity to diagnose the fault, the second to temporarily halt the simulation and the third to resume the simulation.

The *symptom display dialog* shows the initial symptoms observed at the time you begin your troubleshooting task.

The *communication dialog* is used as a medium to present textual messages.

The *error dialogs* convey appropriate messages when you make a mistake.

The display of *error dialogs*, *symptom display dialog*, and the text on the *communication dialog* is accompanied by a beep to draw your attention to these events.

A more detailed description of the interface and valid forms of your interaction with it follows. This section will also provide you with a guided tour to your first session with TURBINIA.

A session with TURBINIA

Welcome to your first session with TURBINIA. You will soon be troubleshooting a simulated failure in a marine power plant. Your first session will be short containing a single problem but subsequent sessions will be of 45 minutes each and will require you to solve three problems. Use the instructions in this section to guide yourself through the first session. The instructions should help you become familiar with the direct manipulation interface of TURBINIA.

At the beginning of every session, the dual screen Apple Macintosh II workstation displays two menus on the large screen and two dialog boxes on the small screen. The two menus on the large screen are the *schematic menu* and the *requests menu*. A *communications dialog* is displayed on the bottom edge of the small screen and an *output file path dialog* is displayed above the communications dialog. This display of the two screens at the start of every session is also shown in Figure 11. If you are starting your first session now, make sure that the screens in front of you look like Figure 11.

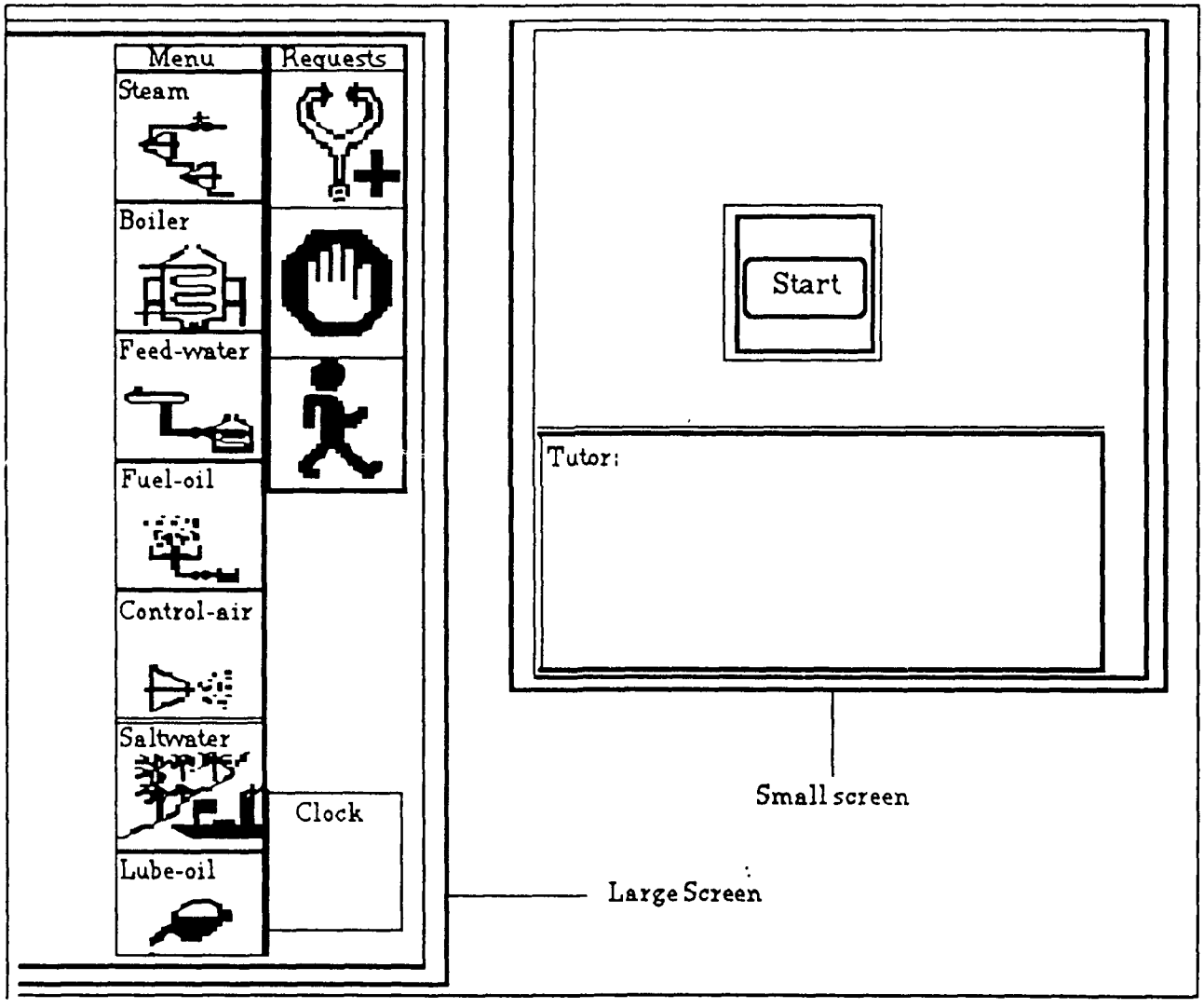


Figure 11. Configuration of Screens

Schematic menu:

The schematic menu on the large screen and also shown in Figure 12, displays seven icons. Each icon represents one of the seven schematics of the simulated power plant. The names of the seven schematics are also provided in the textual form above each icon. The seven schematics are the steam, boiler, feed-water, fuel-oil, control-air, saltwater and lube-oil. You can access any of the seven schematics by clicking on the icon representing the schematic.

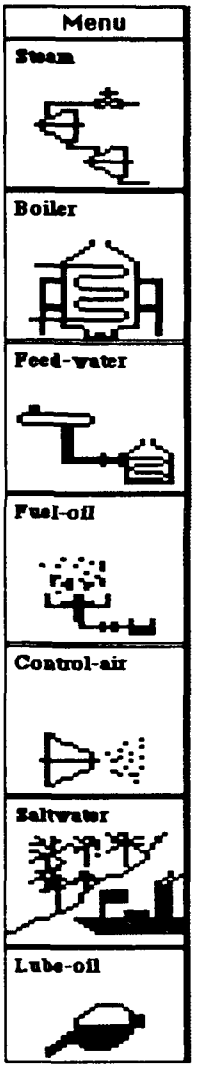


Figure 12. Schematic Menu

Requests menu:

The requests menu adjacent to the schematic menu displays three icons (Figure 13). The first is the diagnose icon. You click on diagnose icon when you have sufficient evidence to confirm your failure hypothesis. By clicking on the diagnose icon, you indicate to

TURBINIA your intention to identify the component responsible for the observed abnormal behavior. You are later provided with an example of how to use the diagnose icon.

The second icon on the requests menu is the stop icon. A click on the stop icon can halt the simulation. However, not all subjects in this experiment need to stop the simulation. In fact you are one of those that need not. Therefore, if you click on the stop icon, an error dialog conveys this message to you.

The third icon on the requests menu is the resume icon. The resume icon is used to restart simulation after it has been halted. Since you will not be stopping the simulation in your sessions, the resume icon has been disabled. All disabled icons in this application have an inverted-gray or tan colored background as compared to enabled icons that have gray for their background color.

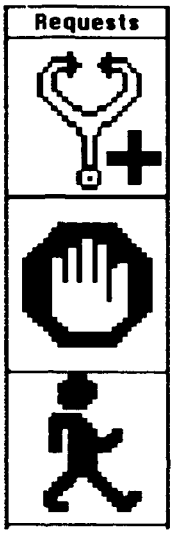


Figure 13. Requests Menu

Communication dialog:

There is a computer-based tutor built on top of TURBINIA to assist users to troubleshoot the power plant. However, not all subjects in this experiment are aided by the tutor. You belong to the group that is unaided. Thus, the tutor, in your case, will only process your request for diagnosis and inform you if your selection of failed component is correct. This communication will take place in the communication dialog on the bottom edge of the small monitor (Figure 14). All communications through this dialog box will be accompanied by a beep.

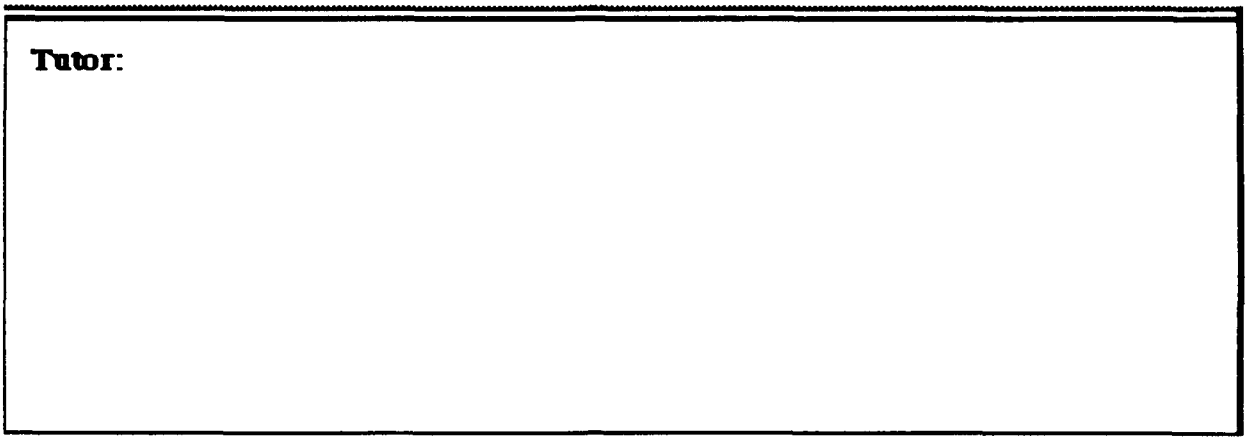


Figure 14. Communication Dialog

Output file path dialog:

Output file path dialog is displayed above the communication dialog on the smaller screen at the beginning of each new session. This dialog is also shown in Figure 15. Output file path dialog expects you type in a name of the file to store your performance data. You begin every session by typing in your last name to create this file. As you type in, you should see the characters appear in the editable text region bounded by a rectangle in the output file path dialog. When you are done typing in your name, check the spelling. If you have made a spelling error, use the delete key on your keyboard to erase characters and make the corrections. When you have your last name spelt correctly, hit the return key.

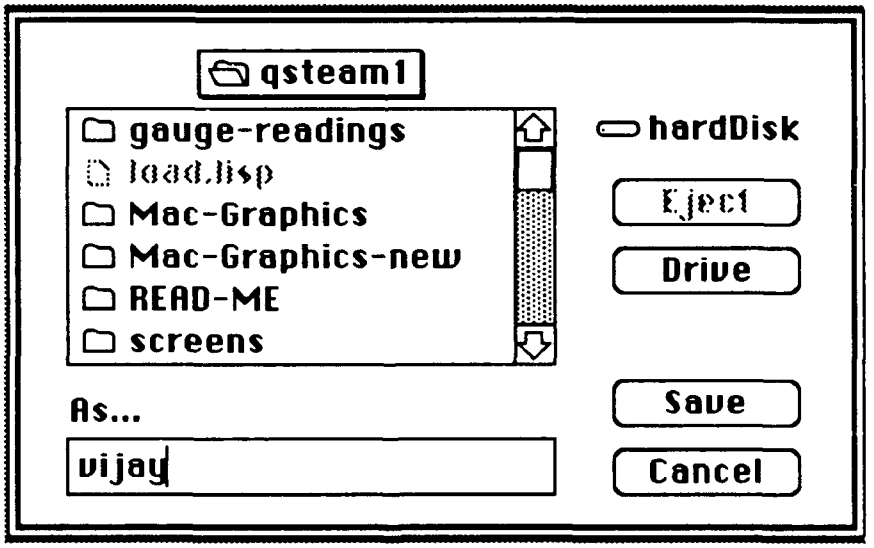


Figure 15. Output File Path Dialog

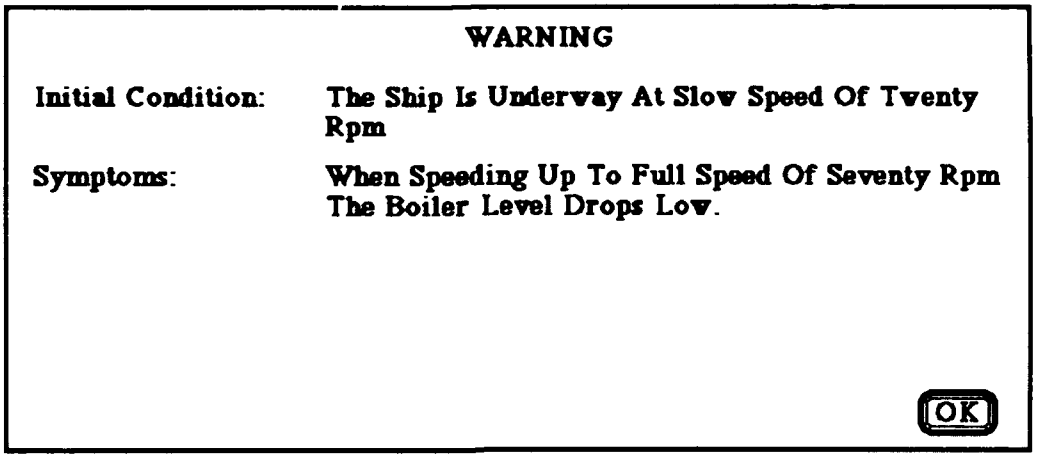


Figure 16. Symptom Display Dialog

Symptom display dialog:

After you have hit the return key, you should see the symptom display dialog appear with a beep in the center of the large screen. The symptom display dialog should look like the one shown in Figure 16. The symptom display dialog shows you the simulated ship's initial operating condition and the first symptoms that indicate the existence of a problem. For your first session, TURBINIA has picked a failure that has caused the feedwater level in the boiler to fall. This information is conveyed to you through the symptom display dialog currently displayed in front of you on the left screen. Your task is to identify the failed component responsible for this abnormal system behavior.

Since the time taken to solve problems is also important, each problem in TURBINIA is simulated for 15 minutes. There is, however, no cascading of failure in this 15 minute period. Therefore, your task is confined to identifying a single component responsible for abnormal system behavior. During the next 15 minutes, you may have to make several investigations before you can accomplish your task. This guided tour of your first session will familiarize you with the actions that are necessary to achieve your goal.

You do not have to memorize the ship's initial operating conditions or the symptoms displayed in the symptom display dialog because you are provided with the facility to recall this information. However, remember that the symptom display dialog is a special dialog used by the application which deactivates your mouse for regions outside the dialog box. Only when you complete interaction with such a dialog, does the mouse get activated again for regions outside the dialog box. TURBINIA has several of these special dialogs called *modal dialogs*. These dialogs respond with a beep if you click the mouse elsewhere without first completing the interaction with them. As an example, try clicking the mouse with the cursor on one of the icons in the schematic menu while the modal symptom display dialog is still visible on the screen. The normal system response to clicking on a schematic menu icon is to display the schematic associated with the icon selected. But this response is currently suppressed by the open modal symptom display dialog. Instead, the system sounds a "beep" to remind you to first finish interacting with the open modal dialog.

You should click on "OK" button in the symptom display dialog to proceed further. Clicking on "OK" completes your interaction with the symptom display dialog and the modal dialog is closed.

Schematics:

Schematics are pictorial representations of the simulated marine power plant. Each schematic presents a view into the structure of the system. A schematic shows the sequence in which components and the gauges appear in the system. If you now click on the boiler icon in the schematic menu, the boiler schematic will be displayed. The boiler schematic is shown in Figure 17. Although the boiler schematic has been chosen as an example to explain the various features of the schematic interface, we could as well have selected any other schematic for this purpose.

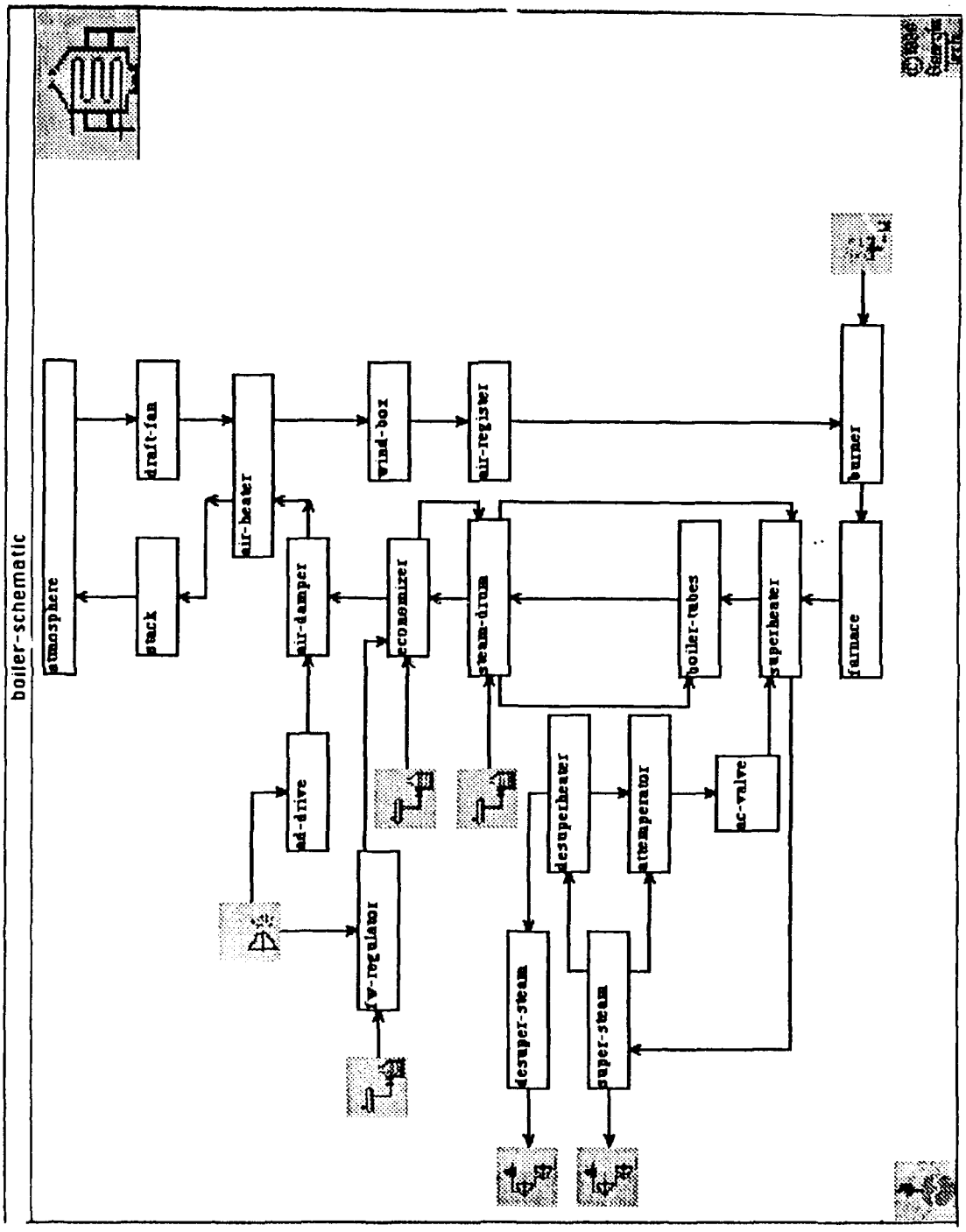


Figure 17. Boiler Schematic

All the components in the boiler schematic have been represented by rectangles. The connections between components are shown by firm lines connecting the components. These firm lines are known as connectors. The direction of flow of fluid from one component to another is shown by the arrow head on these connectors. For example, the economizer and the drum have a two way connection. The connection from the economizer to the drum represents the flow of feedwater while the connection in the reverse direction represents the flow of flue-gases.

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

Now click on the feedwater icon at the end of the input connector of feedwater-regulator and see what happens. You should notice two things. First, the display switches to feedwater schematic. Second, the boiler icon on output connector from the hp-heater is highlighted with a red band around it. The highlighted boiler icon helps you establish the physical connection between the hp-heater and the feedwater-regulator. Click on the highlighted boiler icon to get back to the boiler schematic. Notice that the boiler schematic now has two feedwater icons highlighted, one connected to the feedwater-regulator you clicked on earlier, and the other to the economizer. This simply means that the hp-heater is connected to both the feedwater-regulator and the economizer in the boiler schematic.

Most components of TURBINIA are uniquely represented in one of the seven schematics. However, there are a few that have multiple representations. For example, the condenser and the hp-heater appear in both the steam and the feedwater schematics. Switch to the steam schematic by selecting the steam icon in the schematic menu and locate the condenser and the hp-heater. Multiple representation of these components in schematics is indicated by feedwater icons adjacent to these components. Notice that these icons do not have a connector attached to them. Click on any one of these two icons and your display will switch to the feedwater schematic. The rectangular boxes marked condenser and hp-heater, in this feedwater schematic, are another representation of the same condenser and hp-heater you saw in the steam schematic.

Troubleshooting for failure indicated by the symptoms at the beginning of the session involves gathering information about system states. You collect information concerning system states using a two-action sequence. The first action of the sequence is called the investigative action. Investigative action enables you to display gauges attached to a component, if any. The second action is the informative action that allows you to access the actual gauge reading. The investigative and the informative actions are now explained with an example.

In the current session you have been asked to detect the failure responsible for decreasing water level in the boiler. It is therefore reasonable to investigate components near the boiler. Click on the boiler icon in the schematic menu to view the portion of the power plant with abnormal behavior. As a part of the process to confirm the symptom indicated, move the cursor over the drum and click on it. You have now taken an investigative action and all gauges attached or relevant to the drum are displayed as a result of this action. This action also highlights the last investigated component, the drum, in blue.

There are three types of gauges in TURBINIA: pressure, temperature, and flow-or-level gauges. These three gauges are represented by icons with letters P, T and L inscribed in them to indicate pressure, temperature, and level respectively. Although there is no visible distinction between the flow and the level gauges, it may be helpful to remember that level gauges are attached to tanks such as the deaerating-feed-tank, fuel-oil-settling-tank, atmospheric-drain-tank, distillate-tank, hotwell, and drum. In fact, in TURBINIA, there is just one gauge that measures flow and is located in the fuel-oil path across the strainer.

The drum you are investigating has all three types of gauges attached to it. There are two pressure gauges, one on the flue-gas connector to the economizer and the other on the steam drum. There is a temperature gauge on the feedwater connector from the economizer and a level gauge on the steam drum. The pressure gauge on the steam drum measures the saturated steam pressure in the drum itself and as such is not located over a connector.

To view the reading of a displayed gauge, you have to click on it. This action of probing a gauge is called an informative action. Select the displayed level gauge on the drum and click on it. First, the drum is lowlighted in yellow-brown and then an icon appears near the gauge. This icon is a qualitative representation of the current level. TURBINIA uses five different qualitative representations of state values. These five are normal, low, slightly-low, slightly-high and high, each represented by an icon as shown in Figure 18.

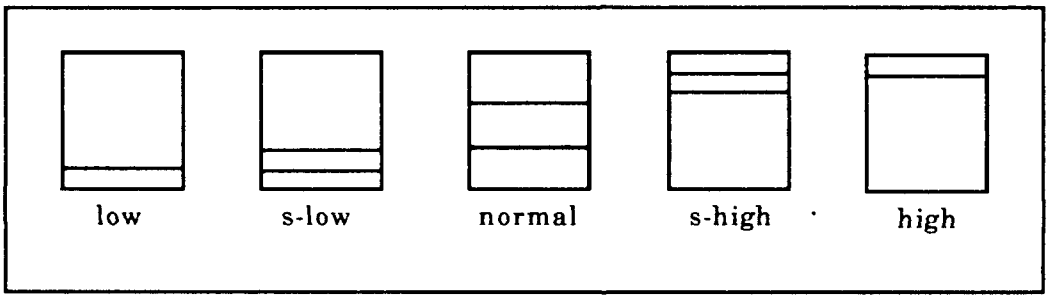


Figure 18. Qualitative State Representation

When you click on the level gauge attached to the drum you should see an icon indicating low level appear at the bottom of the gauge. However, if you see an icon that indicates a slightly-low or normal reading, do not be alarmed. Remember that TURBINIA starts simulating the failure condition at the beginning of the session and if the failure is located far away from the drum, the failure effects will take time to propagate to the drum. Therefore, the low level symptom at the drum, indicated at the beginning of the session, may not yet be visible. You will, however, be able to observe a low level reading during the course of the session.

You should never assume that a gauge reading will be the same at all times after you have observed it. While the gauge readings may change with time, the displayed gauge readings are not dynamically updated. Therefore, you must take an informative action when you need to see the current gauge reading. Thus, if you did not find the drum level low earlier, keep repeating the informative action of selecting the drum's level gauge and you will eventually find it to be low.

You can access any displayed gauge or a gauge reading only until the time you take a new investigative action. Click on the superheater to see what this means. You will discover that all gauges attached to the drum and the probed gauge readings that were visible disappear. Instead, two new gauges, one pressure and the other temperature attached to the

superheater are now displayed in the superheated steam path. Along with the appearance of the gauges, the newly investigated superheater is highlighted in blue.

There are certain components that do not have gauges attached to them. The air-damper is a good example of such a component in the boiler schematic. If you click on the air-damper, all visible gauges and gauge readings on the schematic disappear. Also, the last investigated component is lowlighted and the air-damper is highlighted. However, no new gauges are displayed because there are none attached to the air-damper.

The troubleshooting task typically involves several investigative and informative actions in one or more schematics. Assume that you have conducted several tests and now you have enough evidence to support hypothesis about the failure. Your next valid action, under these circumstances, is to submit a request for conveying the diagnosis. To make this request you click on the diagnose icon in the requests menu. Go ahead and click on the diagnose icon to see how TURBINIA prepares itself to accept your diagnosis.

When you click on the diagnose icon, TURBINIA asks you to select the failed component. This message is conveyed to you through a text appearing in the communications dialog at the bottom of the small screen. You then select the component that, in your opinion, is responsible for the abnormal system behavior. Selecting a failed component is an action identical to investigative action. However, this time, when you click on a component, no gauges are displayed. Instead, if your diagnosis is correct, a message congratulating you appears on the communication dialog. Otherwise, an error dialog accompanied by a beep is displayed over the schematic. This error dialog is shown in Figure 19.

As an example of erroneous diagnose, click on the economizer. The economizer is not the failed component responsible for the current abnormal system behavior. When you complete the click, an error dialog appears over the boiler schematic. This error dialog is a modal dialog. You can close this error dialog by selecting one of the two options available. If you choose "try again" you remain in the diagnose mode and can revise your diagnosis. On the other hand, if you choose "investigate", you are back in the troubleshooting mode where selecting a component displays the gauges attached to it. Click first on "investigate" in the error dialog and then on the drum and you will notice that you are out of the diagnose mode and the four gauges attached to the drum reappear on the screen.

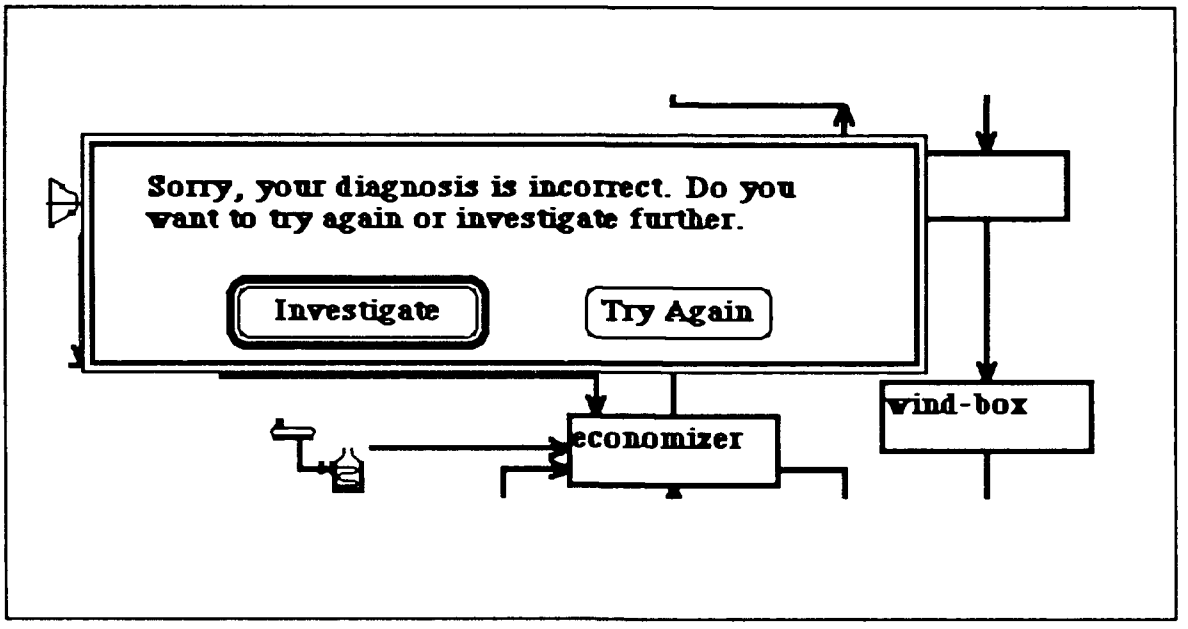


Figure 19. Incorrect Diagnosis

On all schematics you will observe the presence of two other icons that do not represent a schematic and have not also been discussed as yet. One icon appears on the right top corner and the other on the right bottom corner of all schematics. The icon on the right bottom corner is a Georgia Tech. copyright icon. This icon is disabled and has no response. The icon at right top corner is a symptom icon and is used to recall the initial symptoms. Click on this icon and you will see the symptom display dialog reappear. Thus, you can access the ship's initial operating conditions and the initial symptoms at any time. You can once again close the modal symptom display dialog by clicking on the "OK" button.

This section has introduced you to all your valid interactions at TURBINIA's interface. Your valid interactions will be briefly summarized in the next section. Following the summary is a description of how your performance will be measured.

Summary of Valid Actions

Provided below is a list of actions that you will perform while interacting with TURBINIA.

Call-for-schematic-action: This is an action you perform to call a new schematic or switch between schematics. There are two ways this may be done. You can either click on an icon in the schematic menu that represents the schematic you want to view, or, if a schematic is currently displayed, click on a similar icon in the schematic itself. If you are investigating components along a suspected path in a schematic that ends up in an icon, you may prefer to use the icon in the schematic itself to switch to a new schematic. By using the icon in the schematic, a highlighted icon in the new schematic properly orients you to continue investigations along the suspected path.

Investigative-action: During your entire period of interaction with TURBINIA, you are either in troubleshooting or in diagnose mode. When in troubleshooting mode, your action of clicking on the mouse button with cursor on a selected component constitutes an investigative-action. You perform investigative-action to view all gauges attached to the input and output sides of the component being investigated. A new investigative-action always makes the displayed gauges and the gauge readings of the last investigated component disappear from the screen. When no gauges are displayed in response to an investigative-action, it implies that the component investigated has no gauges attached to it.

Informative-action: The gauges displayed following an investigative-action, when probed, display the gauge reading. The action of probing displayed gauges by clicking the mouse on the gauge is called an informative-action.

Diagnose-request-action: This action is performed to switch from the troubleshooting mode to diagnose mode. You perform this action when you are prepared to indicate your diagnosis. Clicking on the mouse button after selecting the diagnose icon in the requests menu constitutes a diagnose-request-action.

Diagnostic-action: Diagnostic-action is performed following the diagnose-request-action. In diagnostic action you select the component that you suspect is responsible for the observed abnormal system behavior. In indicating your diagnosis, you select the component in the same manner as you do when investigating the component. Thus, diagnostic-action is an investigative-action in diagnose mode.

Modal-dialog-action: Modal dialogs deactivate the mouse in regions outside the dialog box. Before the mouse button can be reactivated for regions outside the modal dialog box, you are required to terminate interaction with the modal dialog. Terminating interaction with a modal dialog requires selecting a button dialog item. The action of selecting the button dialog item in the displayed modal dialog is called modal-dialog-action. Symptom display dialog and error dialogs are examples of modal dialog that require modal-dialog-actions.

Measuring Troubleshooting Performance on TURBINIA

Although your ultimate goal is to identify the failed component responsible for abnormal system behavior, your performance is affected by other factors. This section will discuss these factors so that you have a better feel for what is expected of you.

Correct diagnosis: Successful fault diagnosis is the most important measure of your troubleshooting ability. However, since the problems are tough, your inability to solve problems has to be evaluated in conjunction with other factors.

Troubleshooting time: The total amount of time taken for troubleshooting is an important performance measure for those who successfully solve the problem. Those who solve the problems in less time have a better performance rating.

Number of relevant actions: Even though every informative action has some informational content, some have more relevance than others for the failure being investigated. Also, there is a minimum number of relevant informative actions necessary to diagnose each failure. The number of relevant informative actions past this minimum number taken to solve a problem is a measure of diagnostic performance. Smaller number of relevant informative actions required to correctly diagnose the fault implies better performance.

Number of irrelevant actions: The informative actions that have no relevance to the current problem are said to be irrelevant. The larger the number of such irrelevant informative actions during your troubleshooting exercise, the worse is the diagnostic performance.

Number of incorrect diagnosis: You are penalized any time you make an incorrect diagnosis. However, the penalty depends upon the component incorrectly identified as failed. At any stage during the troubleshooting process there are likely candidates for failed component based on the observed abnormal system states. The likelihood that a component may have failed increases or decreases as you conduct more diagnostic tests. Selecting a likely component as the cause of abnormal system behavior does not penalize you as much as picking a component that cannot have failed. Therefore, even though your performance is adversely affected by an incorrect diagnosis, it is considered worse if the suspected component cannot have failed based on the symptoms at the time you express the diagnosis.

Investigation of unaffected schematics: For each failure, there are only few schematics, subsystems and fluid paths that are affected. Affected schematics are those schematics that have gauges with abnormal readings. Investigating components in schematics that are unaffected by the failure reflects the inability on your part to relate the symptoms to the correct structural location of the power plant. Thus, investigating components in unaffected schematics reduces your performance rating.

Investigation of unaffected subsystems: Like the schematics, investigating components in subsystems unaffected by failure reduces your performance rating.

Investigation of unaffected fluid paths: Once again, like the previous two factors, investigating components in unaffected fluid paths harm your performance.

This manual has guided you through your first session, made you familiar with TURBINIA's interface, and has described how your performance will be measured during the experiment. From the next session, you will begin your formal training in troubleshooting marine power plants.

Since it is vital for my experimental results, you are requested not to discuss any aspect of this experiment with other subjects.

GOOD LUCK!

OPERATOR INSTRUCTIONS FOR TURBINIA-VYASA (PASSIVE MODE)

TURBINIA-VYASA is an instructional system that trains operators to troubleshoot marine power plants. TURBINIA is the name of the simulated marine power plant used in the instructional system and VYASA is the computer-based tutor that teaches the troubleshooting task using TURBINIA. As a naval trainee, you will be trained to diagnose some common failures in a marine power plant using this instructional system. This instructional manual will describe your interaction with both TURBINIA and VYASA following a brief description of a typical marine power plant and its control system.

Introduction

A marine power plant is a collection of components configured to produce mechanical work from thermal energy. This energy transformation takes place in components called the *turbines*. A ship that uses steam as a medium to carry the thermal energy to the turbines is said to be steam-driven. In a steam-driven ship the source of thermal energy is usually fossil or nuclear fuel. This section describes the functioning of a fossil fuel-oil fired, steam-driven marine power plant.

The process of producing mechanical work in a steam-driven marine power plant can be decomposed into several stages. Each stage is associated with one of the four phases in the steam cycle: generation, expansion, condensation and feed.

Steam Generation

Figure 1 shows the configuration of components in the generation phase of the steam cycle. This phase of the steam cycle takes place in the boiler. The boiler is comprised of *tubes* and a *steam drum*. The boiler tubes contain water that is heated by flue-gases resulting from the fuel burned in the *furnace*. This heat transfer is by conduction through the tube walls. Heating of water in the tubes produces steam. This steam accumulates over the water surface in the steam drum and is called *saturated steam*. Saturated steam is sometimes also referred to as wet steam because of its moisture content.

Continuous steam generation in the boiler increases the steam pressure in the drum. Boilers are rated by the steam pressure they can handle in the drum. In a 1200-psig boiler, for example, the maximum steam pressure permitted is 1200-psig. A safety valve is activated to release pressure whenever it exceeds the maximum value.

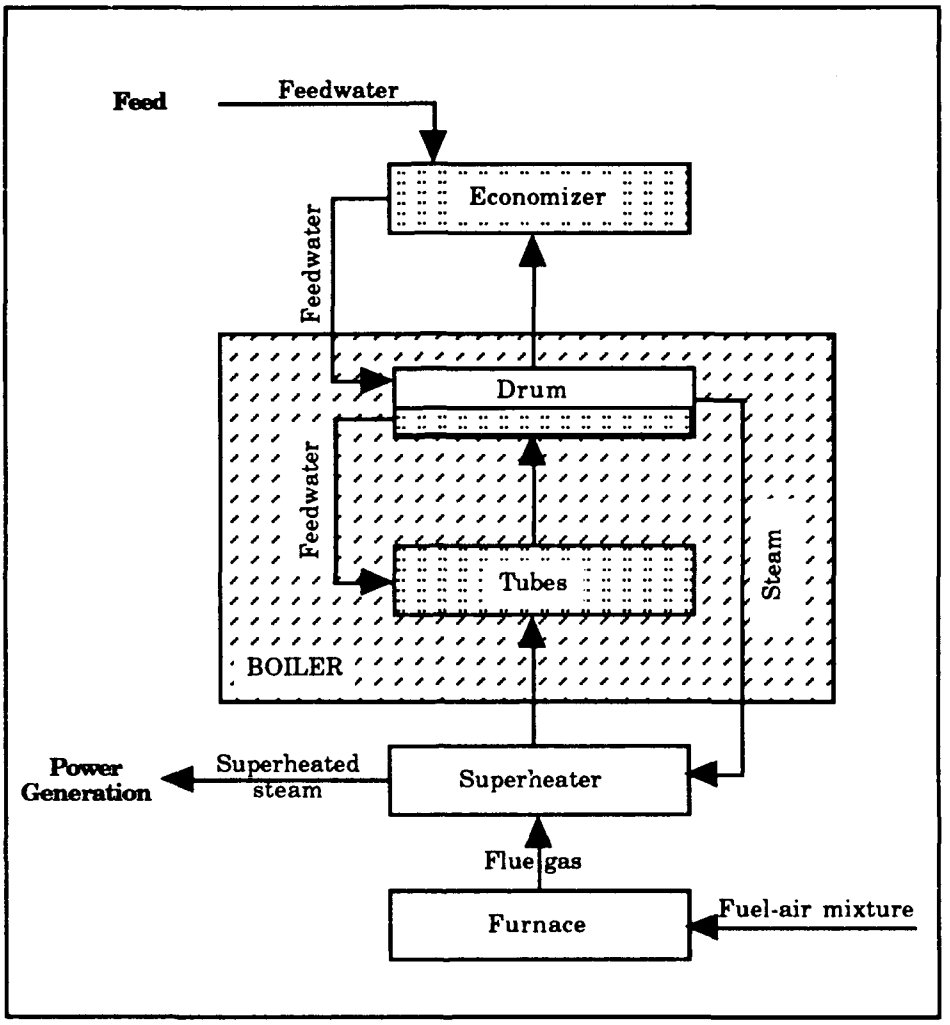


Figure 1. Steam Generation

The steam pressure in the drum controls the temperature at which the water boils in the drum. Since the temperature of saturated steam accumulating above the water surface is the same as the temperature of the water, the saturated steam temperature depends upon the steam pressure. This temperature at which the water and saturated steam coexist in the drum is called the saturation temperature. The highest possible saturation temperature is attained in a boiler when the boiler operates at its maximum rated pressure. Since the thermal energy of steam in the drum is proportional to its saturation temperature, the heat content of the saturated steam is maximum at the highest boiler operating pressure.

Even though the boilers are designed for high operating pressures, the saturated steam does not contain enough thermal energy to operate the turbines at their best efficiency. Thermal energy of steam is increased by passing it through tubes in the section of the boiler closest to the furnace. This section of the boiler is commonly known as the *superheater*. The superheater is responsible for adding heat to saturated steam at constant pressure. The heat added to saturated steam in the superheater is called sensible heat. Sensible heat increases

the temperature of the steam beyond the saturation temperature and makes it drier. The steam from the superheater is called superheated steam and the increase in steam temperature in the superheater measures the degree of superheat.

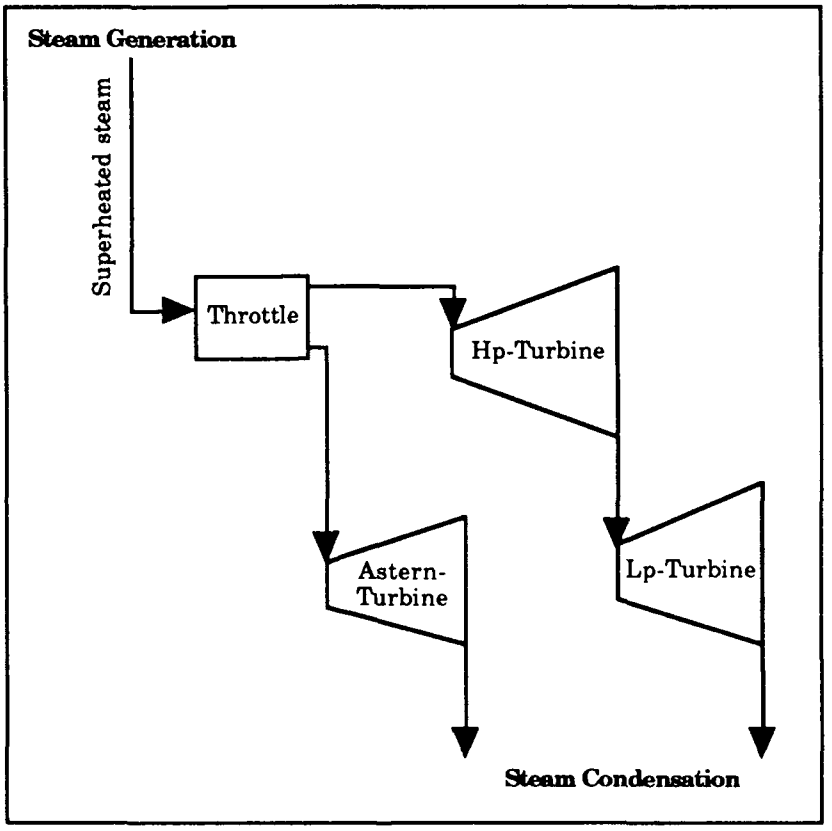


Figure 2. Steam Expansion

Steam Expansion

The second phase of the steam cycle takes place in two steps. First, the superheated steam from the boiler expands in a *high pressure turbine* to convert thermal energy to mechanical work. Then, since the steam still contains a considerable amount of thermal energy, it is expanded further in a *low pressure turbine* connected to the exhaust of the high pressure turbine. Figure 2 shows the arrangement of low and high pressure turbines in a power plant.

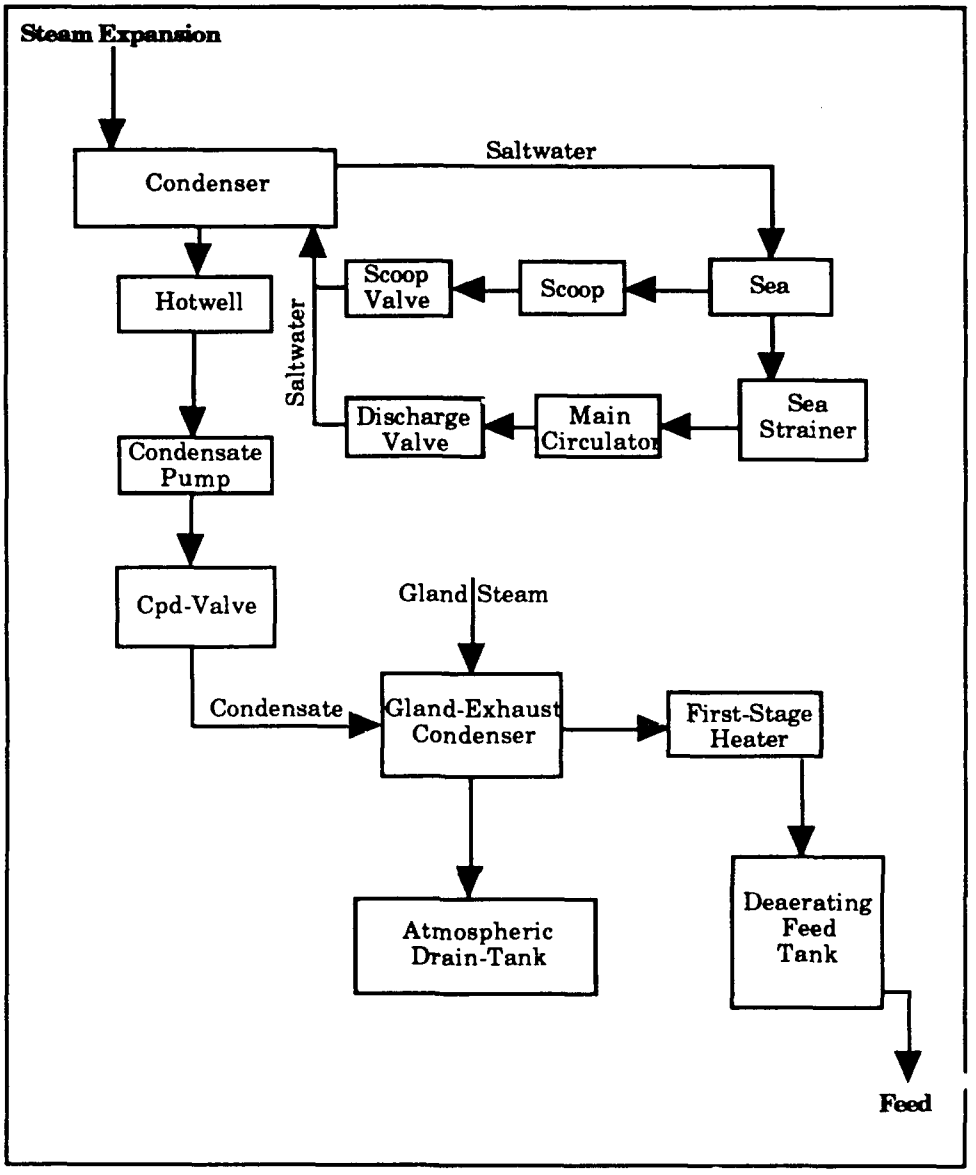


Figure 3. Steam Condensation

Steam Condensation

After expansion, the third phase of the steam cycle is steam condensation which takes place in the main *condenser* (Figure 3). The condenser is a sealed container with tubes that carry cold sea water. When the steam passes over these tubes it loses latent heat to the cold water. When sufficient latent heat is withdrawn from the steam, it changes phase and turns back into water, called *condensate*.

Steam pressure at the turbine exit is low and steam can flow into the main condenser only if the pressure in the condenser is lower. Since the condensate occupies less volume than the same amount of steam and because the condenser is a sealed container, condensation creates a vacuum in the condenser shell. This vacuum in the condenser shell helps maintain a continuous flow of steam from the turbines to the condenser.

As the steam from the turbines turns into condensate, it flows into a collecting tank called the *hotwell*. The *condensate-pump* then pumps the condensate to the *deaerating-feed-tank* via the *gland-exhaust-condenser*. In the gland-exhaust-condenser, the condensate from hotwell serves as the cooling medium to condense steam from the turbine glands. While the condensate from the gland-exhaust-condenser flows to the deaerating-feed-tank, the condensed gland steam is returned to the condensate system by way of *atmospheric-drain-tank*.

Feed

Feed, the last phase of the steam cycle, begins at the deaerating-feed-tank. The deaerating-feed-tank is a storage tank for feedwater. It also contains apparatus to remove dissolved oxygen entrained in the condensate. The other major components in the feed phase are the main feed pump, the feed water regulator and the economizer. These components are shown in Figure 4. The main feed pump is responsible for pumping water to the boiler. The feed water regulator regulates feedwater into the economizer enroute to the boiler. The economizer is a heat exchanger that preheats the feedwater.

Each of the four phases of the steam cycle perform an important system function. The collection of components responsible for the function constitute a functional subsystem. Thus, **steam generation**, **steam expansion** or **power generation**, **steam condensation**, and **feedwater preheating** are also essential *subsystems* of a marine power plant. In addition, a power plant typically has subsystems that perform other functions necessary for its operation. **Combustion**, **auxiliary steam use**, **control air**, **lube oil**, and **saltwater service** are examples of such subsystems.

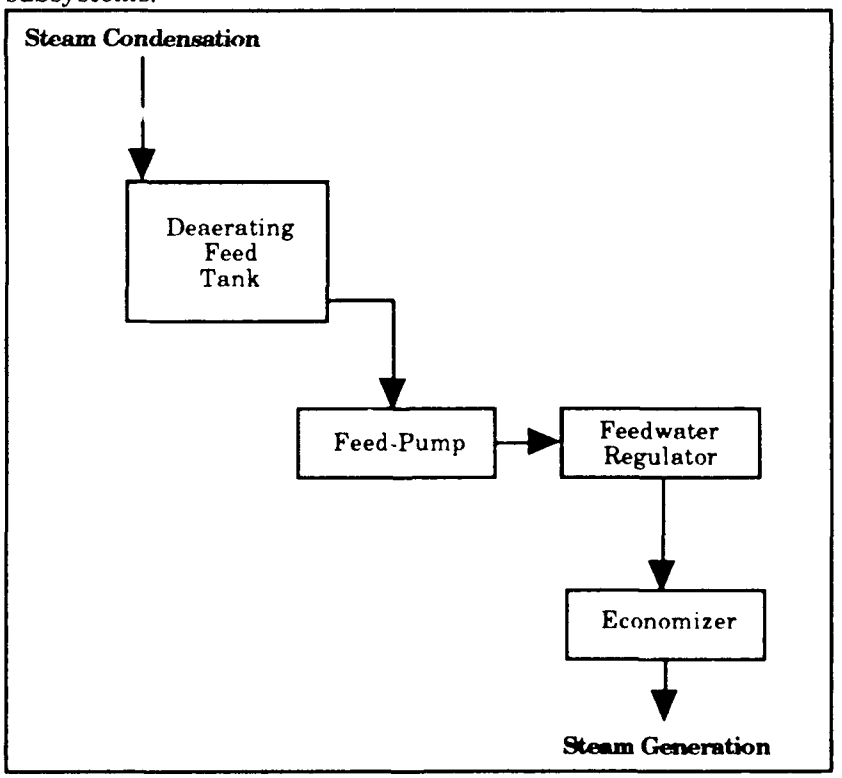


Figure 4. Feed

Combustion involves burning the fuel-air mixture prepared in the *burner*. The thermal energy released during combustion is used to heat water in the boiler. The components that make up the combustion subsystem are shown in Figure 5. These components lie along two fluid paths: *combustion air* and *fuel-oil*.

Combustion air is supplied to the burner by a forced draft fan operated by either a steam turbine or an electric motor. Fuel-oil is supplied to the burner by pumping fuel from a *settling-tank*. For proper combustion, both the combustion air and the fuel-oil need to be at the proper pressure and temperature. Furthermore, for complete combustion the mass of air required is fourteen times the mass of fuel-oil.

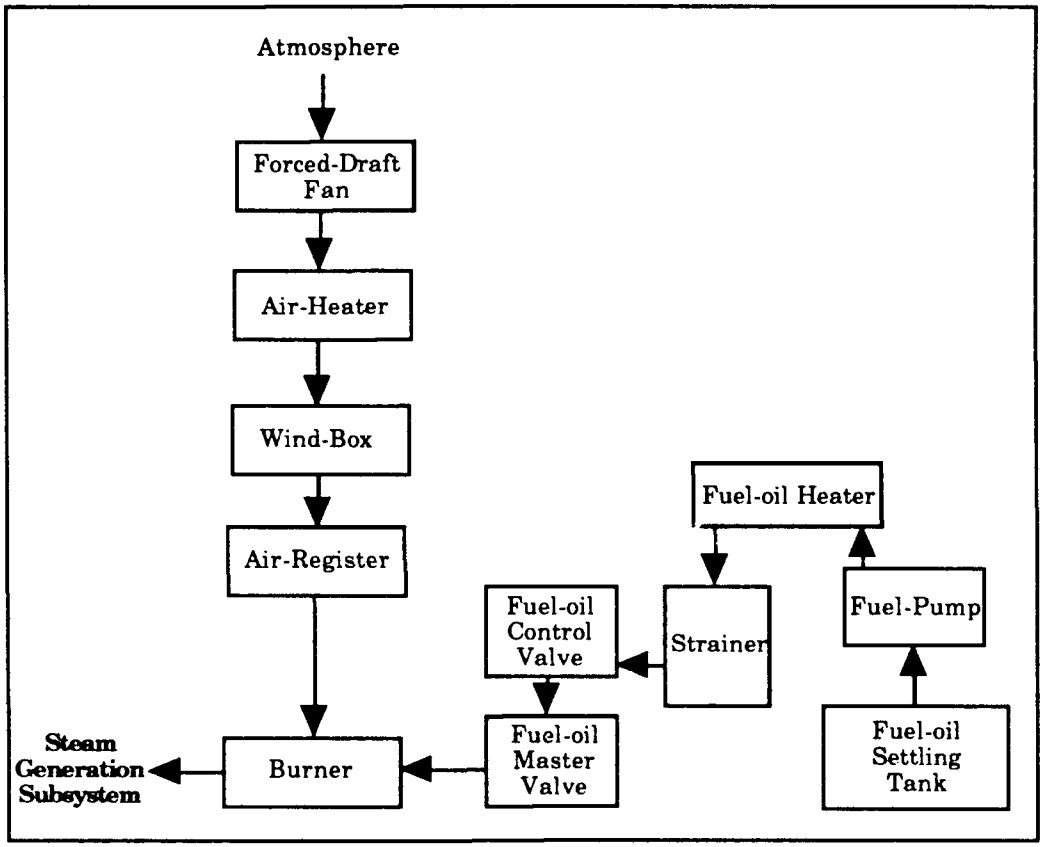


Figure 5. Combustion Subsystem

Incorrect quantity or improper heating of either the combustion air or fuel-oil causes combustion problems. Inadequate quantity or insufficient heating of combustion air and excessive flow of fuel-oil or insufficient heating of it causes incomplete combustion. Incomplete combustion causes dark smoke in the boilers. On the other hand, excess quantity of combustion air in the fuel-air mixture either due to increased flow rate of air or reduced flow rate of fuel-oil extinguishes the flame in the furnace. Excessive preheating of either the combustion air or the fuel-oil causes yet another combustion problem called preignition. In preignition, the fuel starts to burn before it reaches the burner.

The auxiliary steam use subsystem shown in Figure 6 uses *desuperheated steam* for various purposes. Desuperheated steam, unlike the superheated steam, is low pressure steam. Desuperheated steam is obtained by passing superheated steam through the *desuperheater*. Low pressure desuperheated steam is used (1) by the auxiliary power units

to run equipment such as the *feedwater-pump*, *fuel-pump*, *saltwater-service-pump* and the *forced-draft -fan*; (2) to preheat the fuel-oil and the feedwater in the deaerating-feed-tank; and (3) by the the gland seals to prevent leakage of air into the turbine casings and steam leakage out of the casings.

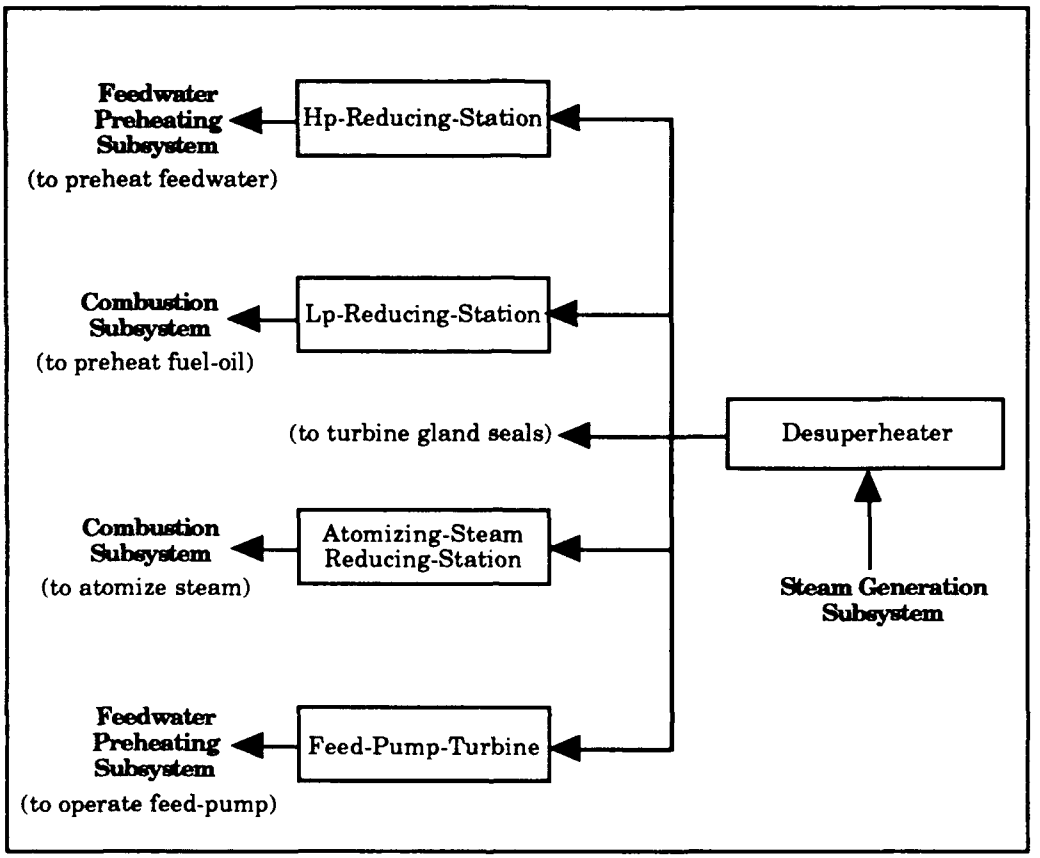


Figure 6. Auxiliary Steam Use Subsystem

The control air subsystem is responsible for distributing control air to many valves and regulators. These valves and regulators are operated by the control air. Figure 7 shows the components that constitute the control air subsystem of a marine power plant.

The lubrication subsystem has the primary purpose of lubricating moving parts and removing the heat produced by friction. The subsystem consists of a pump that draws lube-oil from the *oil-sump* and distributes it to those components that need lubrication. Figure 8 shows the components in the lubrication subsystem.

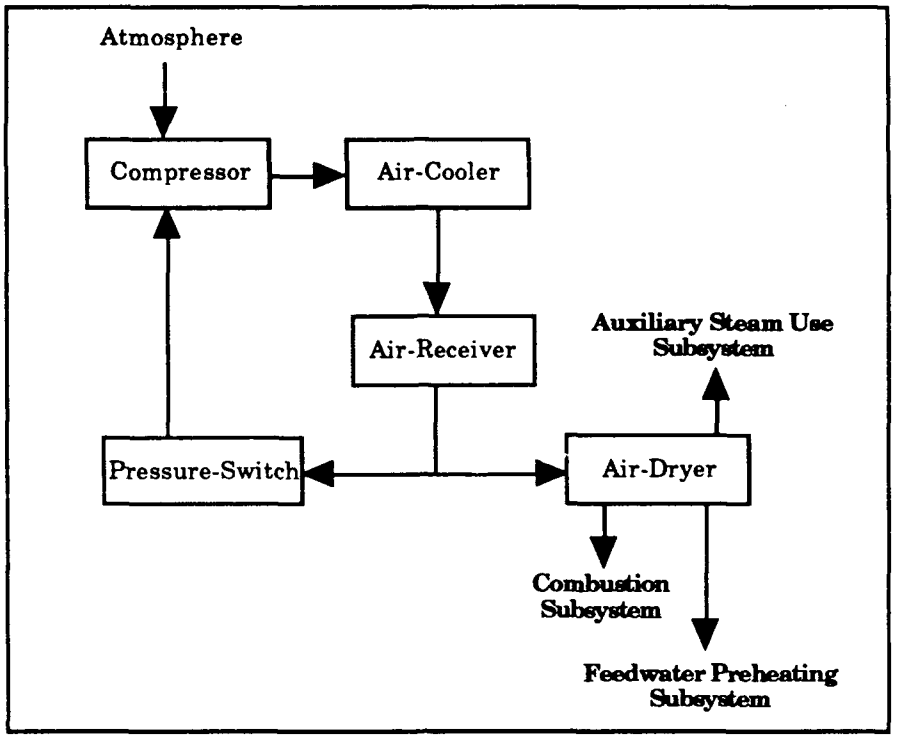


Figure 7. Control Air Subsystem

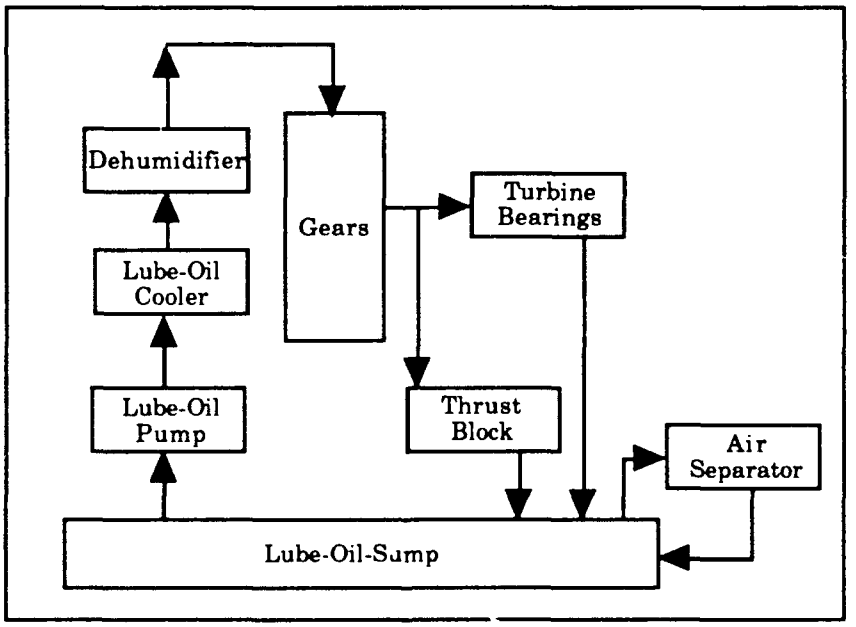


Figure 8. Lubrication Subsystem

The saltwater service subsystem distributes the cold sea water to remove heat from units dissipating heat. Figure 9 shows sections of the power plant cooled by the saltwater.

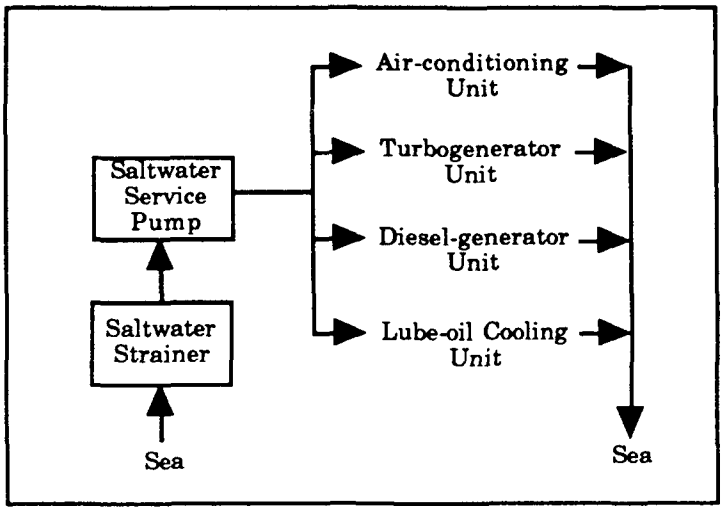


Figure 9. Saltwater Service Subsystem

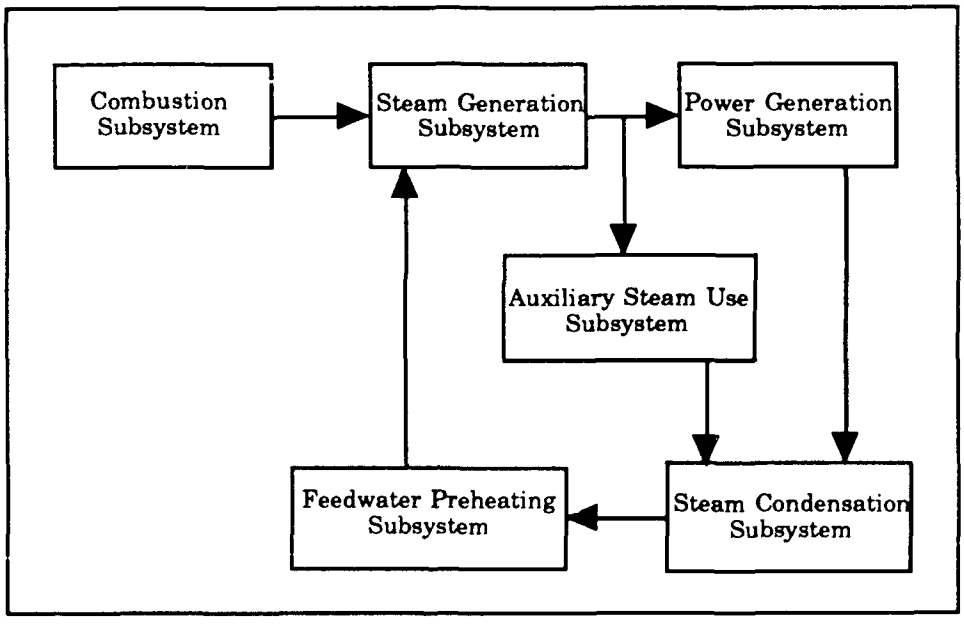


Figure 10. Interacting Subsystems of Marine Power Plant

This section described the decomposition of a marine power plant into nine functional subsystems. It also described the role each subsystem plays in achieving the overall goal of producing power. The interaction between the subsystems to produce power is summarized in Figure 10. For constant power supply, the operating conditions for these subsystems can be set to safely meet the demand. However, the demand for power in a ship is never constant but varies with load. Control systems manage the power plant so that it can satisfy the changes in the demand for power due to fluctuating load. The boiler control system is the most important among all control systems in a marine power plant. The boiler control system of most modern Navy vessels is sophisticated and needs minimum human intervention. Some components of the automatic boiler control system (ABC) are described next.

Automatic Boiler Control System

Navy vessels typically have the following three ABC systems: automatic combustion control (ACC), feedwater control (FWC), and makeup and excess feed control systems. These control systems perform the functions of measuring, comparing, computing and correcting. In each control system, a state value of interest is measured; compared to a desired value; a new operating condition, if necessary, is computed; and finally a correction made in the operating conditions to reduce the deviation between the measured and the desired value of the state.

Automatic Combustion Control System. The function of the automatic combustion control system is to maintain the boiler drum pressure at a constant value during steady and changing load conditions. The ACC system accomplishes this task by

(1) constantly measuring the steam drum pressure and combustion air flow;
(2) comparing the steam drum pressure to the specified designed value;
(3) computing the amount of change, if any, in the furnace combustion; and
(4) correcting furnace combustion as needed.

When the steam demand on the boiler is increased, the steam drum pressure decreases because the rate of steam withdrawal from the drum becomes greater than the rate of steam production in the boiler. This pressure drop is sensed by the ACC system and an increase in furnace combustion is computed to meet the increase in the demand for steam. Computing the increase in furnace combustion involves computing the increase in the supply of combustion air and a proportionate increase in the supply of fuel-oil to assure complete combustion. The ACC system controls the combustion air flow by regulating the supply of steam to the forced draft fan turbine and controls the fuel-oil flow by positioning the main-fuel-oil-control-valve. The measurement of air flow provides the ACC system with the feedback necessary to perform this function.

Feedwater Control System. The function of the feedwater control system is to maintain a constant water level in the steam drum. The FWC system automatically does this by

(1) measuring the steam drum water level and the feedwater flow rate to the boiler;
(2) comparing the measured water level in the drum to a designed value;
(3) computing the required change, if any, to the rate of feedwater flow; and
(4) correcting the feedwater flow rate as needed.

When the load is steady, the feedwater flow rate into the boiler equals the rate of steam consumption and the water level in the steam drum is normal. But, when the load changes, so does the demand for steam. Any change in this demand is detected and the feedwater flow rate is increased or decreased to equal the steam flow rate out of the boiler. The actual control of feedwater flow is accomplished by adjusting the air-operated diaphragm of the feedwater regulator between the feed pump and the boiler.

Makeup and Excess Feed Control System. Operation of a steam-driven power plant often requires the addition or removal of water from the steam cycle. The makeup and excess feed control system is responsible for doing this and for maintaining a specified level of feedwater in the deaerating-feed-tank.

Whenever the level in the deaerating-feed-tank deviates from the specified value, water is either withdrawn from or added to the deaerating-feed-tank. In both cases the process is facilitated by two standby tanks. The two standby tanks are the atmospheric-drain-tank

and the distillate-tank. When the feedwater level in the deaerating-feed-tank falls below normal, the *makeup-feed-regulator* is adjusted by the control system to increase flow from the standby tanks. Increased flow into the deaerating-feed-tank compensates for the loss in the feedwater level. Similarly, a *deaerating-dump-regulator* is activated by the control system to withdraw excess feedwater from the deaerating-feed-tank when the level in the tank rises above the normal value.

In addition to the automatic boiler control system, a power plant has several other controls which are not discussed here because they are not relevant to your task. However, knowledge concerning some common modes of failure in components of a power plant is useful for diagnosing faults and is described next.

Common Modes of Failure

A mechanical component in a physical system like the marine power plant can fail in more than one way. The four most common modes of failure for components of TURBINIA are: (a) blocked-shut, (b) stuck-open, (c) leak-in, and (d) leak-out. Faults in components fit one or more of these four mode types.

A blocked-shut component offers greater than normal resistance to the flow of fluid for the desired operating condition. A valve that cannot position its vane to a larger opening demanded by the new operating condition or a strainer that is clogged with dirt are examples of the blocked-shut mode of failure.

A stuck-open component offers less than normal resistance to the flow of fluid for the desired operating condition. A valve that refuses to position its vane to a smaller opening on command is an example of the stuck-open mode of failure.

A component failed in leak-in mode allows undesirable or excess flow of fluid *into* it, while a leak-out mode of failure causes undesirable passage of fluid *out* of the component. A vacuum tank that allows air to leak in from outside and a ruptured piping that allows the fluid it carries to leak out from it are examples of leak-in and leak-out modes of failure respectively.

Each failure mode is responsible for a system behavior that manifests in the form of a typical pattern of abnormal state values. During diagnostic problem solving, it is often helpful to identify the failure mode from system behavior and confine the search to components that fail in the identified mode. The typical system behavior associated with a fault also depends upon the phase of the fluid in the affected path. The following set of examples explains the abnormal system behavior for each of the four modes of failure in liquid and gas paths.

A blocked-shut mode of failure in a liquid path causes the liquid *level* downstream to be lower than normal and the level upstream higher than normal. A similar blocked-shut mode of failure in a gas path, on the other hand, decreases the downstream gas *pressure* and increases the upstream pressure.

A stuck-open mode of failure in a liquid path causes the liquid *level* downstream to be higher than normal and the level upstream lower than normal. A similar stuck-open mode of failure in the gas path increases the downstream gas *pressure* and decreases the upstream pressure.

When a container that stores liquid allows more of the same liquid to leak in, the *level* of the liquid in the container increases. When the same container stores gas and allows more of it to leak in from the high pressure surroundings, the *pressure* in the container becomes abnormally higher.

A ruptured component that allows liquid to leak out causes a drop in the liquid *level* upstream as well as downstream from the place of leakage. A similarly ruptured component carrying gas causes a drop in *pressure* upstream and downstream from the place of leakage.

Although there is a typical system behavior associated with each mode of failure, it is not always easy to observe the abnormal behavior in a real system. This is due to the limited number of available gauges. Therefore, pressures, temperatures, and flows cannot be measured across every component. Furthermore, certain components can prevent propagation of expected abnormal behavior past them. For instance, a source-sink such as a deaerating-feed-tank located downstream in the blocked-shut condensate path prevents further propagation of low level downstream from the tank. The deaerating-feed-tank imposes such a behavior on the system because it is an infinite source of feedwater which temporarily compensates for any loss of water level. A summary of typical system behavior associated with the four failure modes is shown in Table 1.

Failure Mode	Fluid	State	Abnormal Behavior Upstream	Abnormal Behavior Downstream	Propagation Limited By Upstream	Propagation Limited By Downstream
Blocked-Shut	Liquid	Level	High	Low	Infinite Sink	Infinite Source
	Gas	Pressure	High	Low	Safety Valve	Infinite Source
Stuck-Open	Liquid	Level	Low	High	Infinite Source	Infinite Sink
	Gas	Pressure	Low	High	Infinite Source	Safety Valve
Leak-In	Liquid	Level	High	High	Infinite Sink	Infinite Sink
	Gas	Pressure	High	High	Safety Valve	Safety Valve
Leak-Out	Liquid	Level	Low	Low	Infinite Source	
	Gas	Pressure	Low	Low		

Table 1. Typical Abnormal System Behavior

This completes a description of a typical marine power plant, its control systems, the four common modes of failure in components of the power plant, and the typical abnormal system behavior associated with each of the four failure modes. The next section describes TURBINIA's interface.

The Interface

TURBINIA- the marine power plant simulator, and VYASA- the computer-based tutor have been developed on a dual screen Apple Macintosh II workstation. The dual screen configuration consists of one 19" color monitor and a 13" color monitor. In this set up, the larger monitor is the left screen and the smaller monitor is the right screen. A single button computer mouse that can point to all locations on both screens is the only input device. You will use this mouse to interact with the direct manipulation interface of both TURBINIA and VYASA. Almost all your actions involve moving the mouse cursor to a desired location and clicking on the mouse button. All valid user actions have appropriate response while invalid actions are ignored by the system. Valid actions at the joint TURBINIA-VYASA interface are described in detail later.

The interface to TURBINIA-VYASA consists of seven schematic windows, a schematic menu, a requests menu, a communication dialog, multiple levels of hierarchically organized passive tutor dialogs, a symptom display dialog and several error dialogs.

The seven *schematics* display the physical connections between the components of the power plant. You will use these schematics to investigate components and probe gauges attached to these components.

The *schematic menu* displays seven icons each representing one of the seven schematics. You will use these icons to access the schematics.

The *requests menu* has three icons. You will use the first icon to request for an opportunity to diagnose the fault, the second to temporarily halt the simulation and the third to resume the simulation.

The *communication dialog* is used by VYASA to provide instructions.

The *passive tutor dialogs* establish your communications with VYASA when you seek knowledge concerning the structure, function and behavior of the subsystems and the components. You will use the passive tutor dialogs to explore the tutor's knowledge-base.

The *symptom display dialog* shows the initial symptoms observed at the time you begin your troubleshooting task.

The *error dialogs* convey appropriate messages when you make a mistake.

The display of *error dialogs*, *symptom display dialog*, or new text on the *communication dialog* is accompanied by a beep.

A more detailed description of the interface and valid forms of your interaction with the system follows. You will now be given a guided tour of your first session with the instructional system.

A session with TURBINIA-VYASA

Welcome to your first session with TURBINIA-VYASA. You will soon be troubleshooting a simulated failure in a marine power plant. You will be aided in your task by the computer-

based tutor VYASA. This computer-based tutor functions in two modes: passive and active. For your training sessions, the tutor will function in only the passive mode. In the passive mode, the tutor will respond to your queries but will not intervene on its own to provide you with instructions. Your first session has been designed to make you familiar with the joint interface of TURBINIA and VYASA. This first session will be short containing a single problem. Subsequent sessions will be of 45 minutes each and will require you to solve three problems. Use the instructions in this section to guide yourself through the first session.

At the beginning of every session, the dual screen Apple Macintosh II workstation displays two menus on the large screen and two dialog boxes on the small screen. The two menus on the large screen are the *schematic menu* and the *requests menu*. A *communications dialog* is displayed on the bottom edge of the small screen and an *output file path dialog* is displayed above the communications dialog. This display of the two screens at the start of every session is also shown in Figure 11. If you are starting your first session now, make sure that the screens in front of you look like Figure 11.

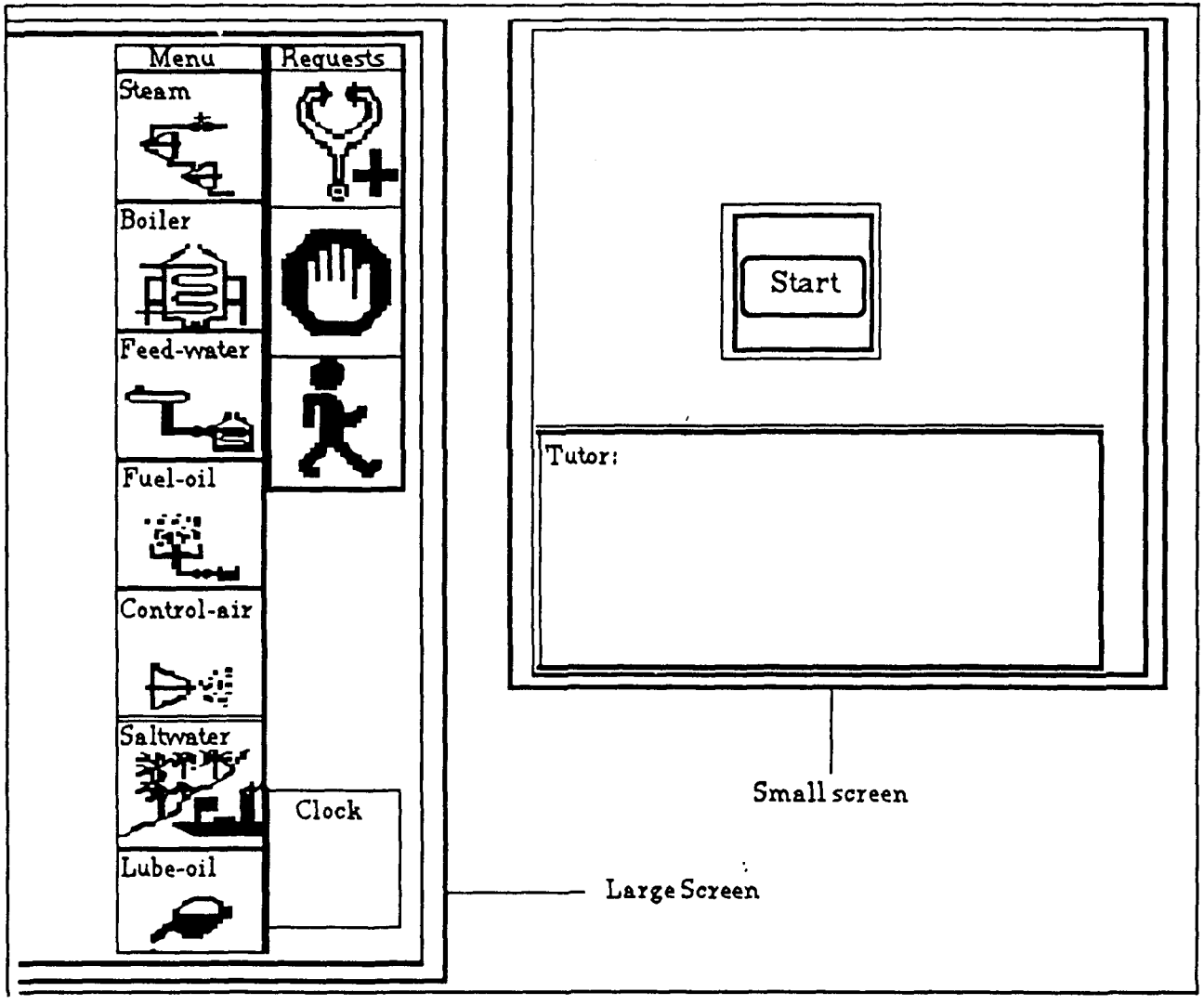


Figure 11. Configuration of Screens

Schematic menu:

The schematic menu on the large screen and also shown in Figure 12, displays seven icons. Each icon represents one of the seven schematics of the simulated power plant. The names of the seven schematics are also provided in the textual form above each icon. The seven schematics are the steam, boiler, feed-water, fuel-oil, control-air, saltwater and lube-oil. You can access any of the seven schematics by clicking on the icon representing the schematic.

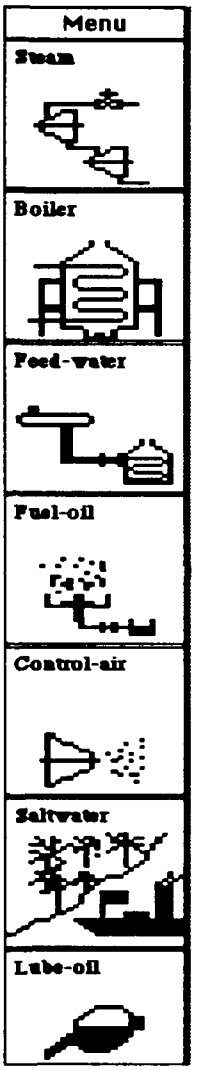


Figure 12. Schematic Menu

Requests menu:

The requests menu adjacent to the schematic menu displays three icons (Figure 13). The first is the diagnose icon. You click on diagnose icon when you have sufficient evidence to confirm your failure hypothesis. By clicking on the diagnose icon, you indicate to the instructional system your intention to identify the component responsible for the observed

abnormal behavior. You are later provided with an example of how to use the diagnose icon.

The second icon on the requests menu is the stop icon. A click on the stop icon can halt the simulation putting you in a mode to interact with the passive tutor. Your interaction with the passive tutor is later described in detail.

The third icon on the requests menu is the resume icon. The resume icon is used to restart simulation after it has been halted to communicate with the passive tutor. Since you are currently not interacting with the passive tutor, the resume icon is shown disabled. All disabled icons in this application have an inverted-gray or tan colored background as compared to enabled icons that are shown in gray.

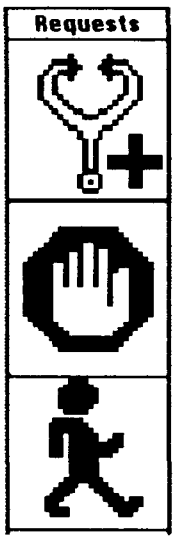


Figure 13. Requests Menu

Communication dialog:

The computer-based tutor VYASA communicates with you through textual messages and instructions presented on the communication dialog. This communication dialog is displayed on the bottom edge of the small monitor (Figure 14). All communications through this dialog box are accompanied by a beep.

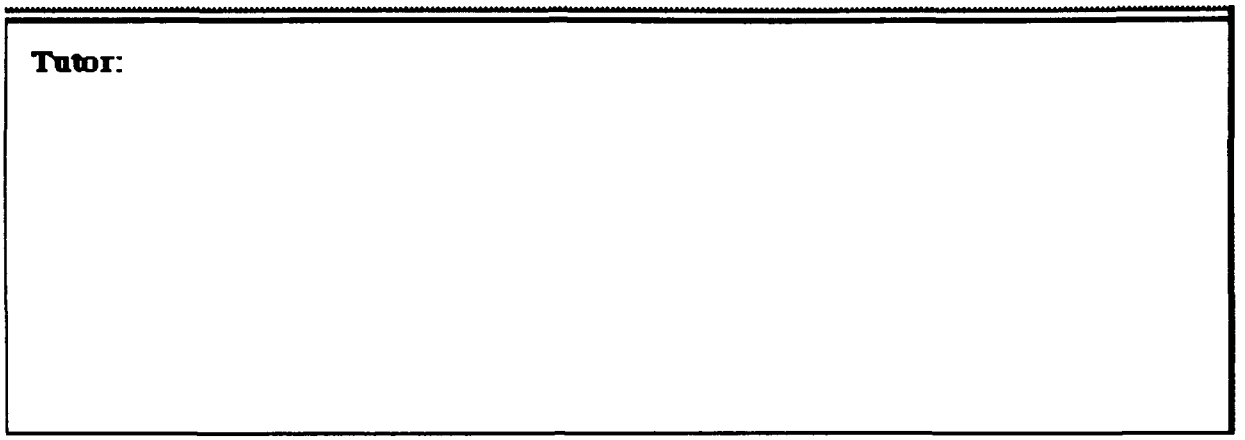


Figure 14. Communication Dialog

Output file path dialog:

Output file path dialog is displayed above the communication dialog on the small screen at the beginning of each new session. This dialog is also shown in Figure 15. Output file path dialog expects you type in a name of the file to store your performance data. You begin every session by typing in your last name to create this file. As you type in, you should see the characters appear in the editable text region bounded by a rectangle in the output file path dialog. When you are done typing in your name, check the spelling. If you have made a spelling error, use the delete key on your keyboard to erase characters and make the corrections. When you have your last name spelt correctly, hit the return key.

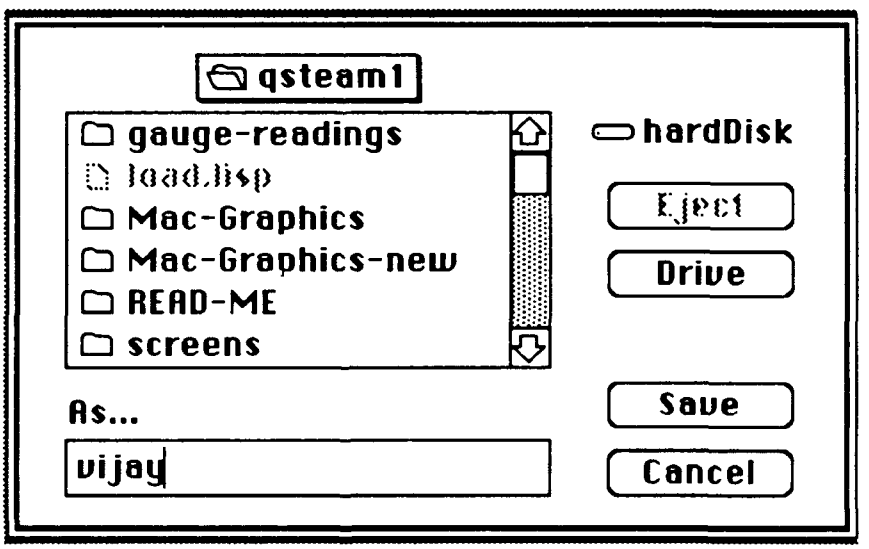


Figure 15. Output File Path Dialog

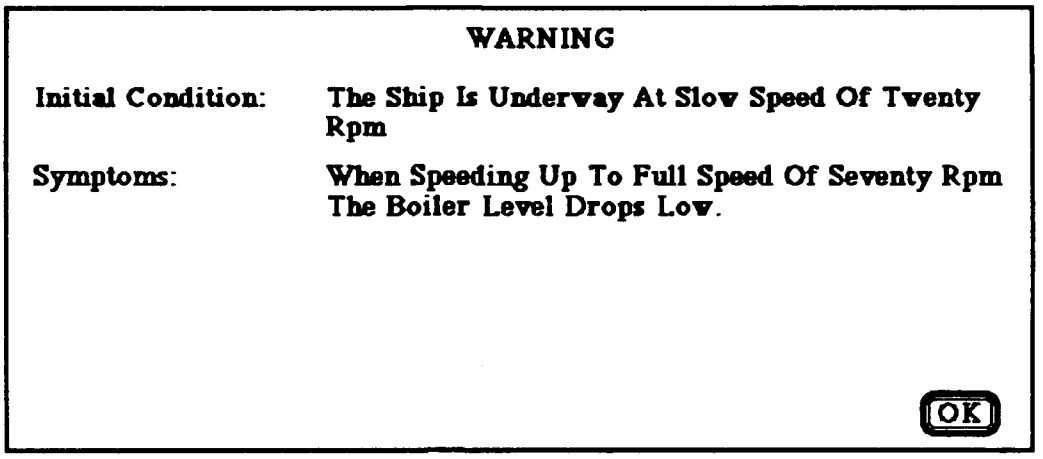


Figure 16. Symptom Display Dialog

Symptom display dialog:

After you have hit the return key, you should see the symptom display dialog appear with a beep in the center of the large screen. The symptom display dialog should look like the one shown in Figure 16. The symptom display dialog shows you the simulated ship's initial operating condition and the first symptoms that indicate the existence of a problem. For this first session, VYASA has picked a failure that has caused the feedwater level in the boiler to fall. This information is conveyed to you through the symptom display dialog currently displayed in front of you on the left screen. Your task is to identify the failed component responsible for this abnormal system behavior.

Since the time taken to solve problems is also important, each problem is simulated for 15 minutes. There is, however, no cascading of failure in this 15 minute period. Therefore, your task is confined to identifying a single component responsible for abnormal system behavior. During the next 15 minutes, you may have to make several investigations before you can accomplish your task. This guided tour of your first session will familiarize you with the actions that are necessary to achieve your goal.

You do not have to memorize the ship's initial operating conditions or the symptoms displayed in the symptom display dialog because you are provided with the facility to recall this information. However, remember that the symptom display dialog is a special dialog used by the application which deactivates your mouse for regions outside the dialog box. Only when you complete interaction with such a dialog, does the mouse get activated again for regions outside the dialog box. This instructional system has several of these special dialogs called the *modal dialogs*. These dialogs respond with a beep if you click the mouse elsewhere without first completing the interaction with them. As an example, try clicking the mouse with the cursor on one of the icons in the schematic menu while the modal symptom display dialog is still visible on the screen. The normal system response to clicking on a schematic menu icon is to display the schematic associated with the icon selected. But this response is currently suppressed by the open modal symptom display dialog. Instead, the system sounds a "beep" to remind you to first finish interacting with the open modal dialog.

You should click on "OK" button in the symptom display dialog to proceed further. Clicking on "OK" completes your interaction with the symptom display dialog and the modal dialog is closed.

Schematics:

Schematics are pictorial representations of the simulated marine power plant. Each schematic presents a view into the structure of the system. A schematic shows the sequence in which components and the gauges appear in the system. If you now click on the boiler icon in the schematic menu, the boiler schematic will be displayed. The boiler schematic is shown in Figure 17. Although the boiler schematic has been chosen as an example to

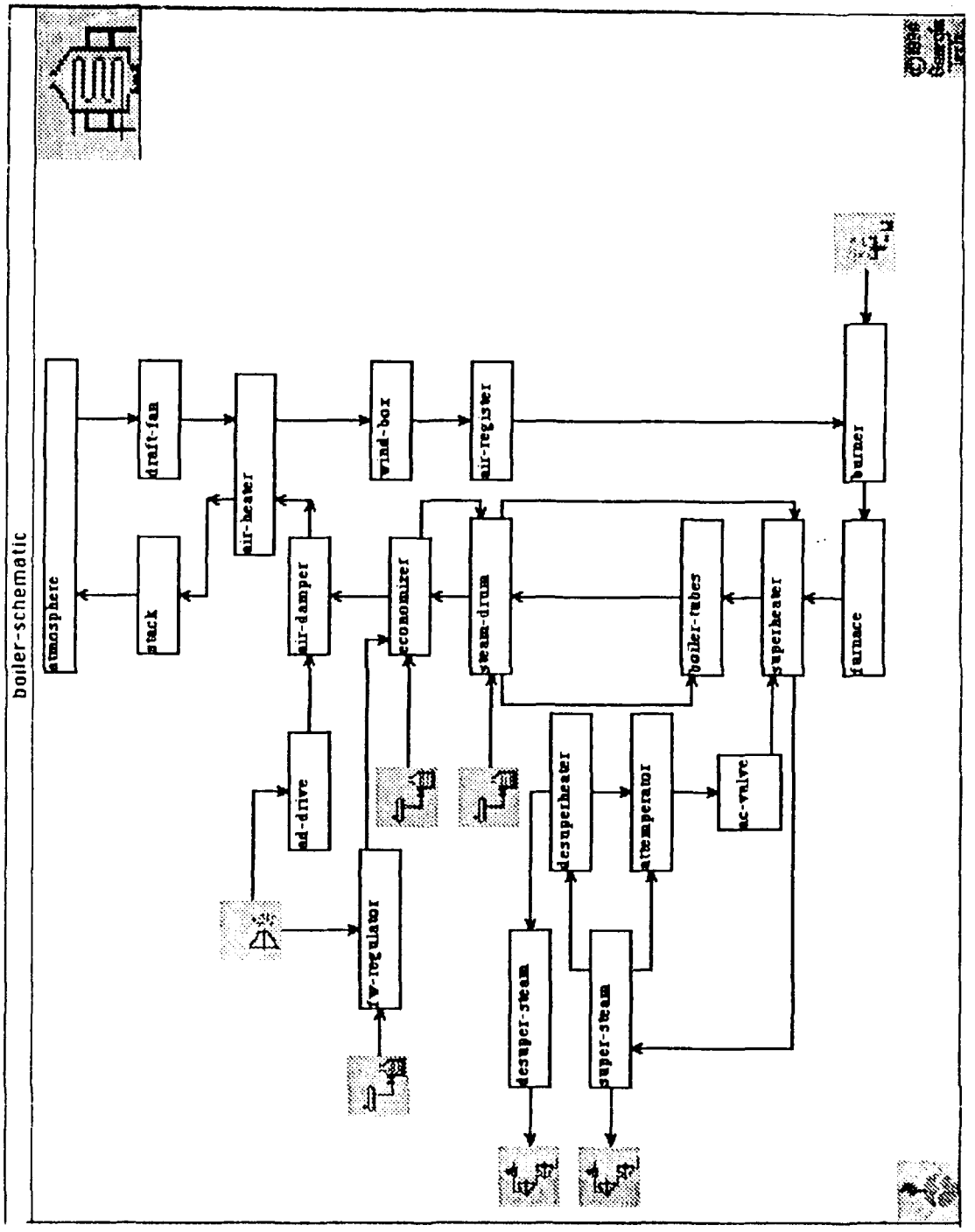


Figure 17. Boiler Schematic

explain the various features of the schematic interface, we could as well have selected any other schematic for this purpose.

All the components in the boiler schematic have been represented by rectangles. The connections between components are shown by firm lines connecting the components. These firm lines are known as connectors. The direction of flow of fluid from one component to another is shown by the arrow head on these connectors. For example, the economizer and the drum have a two way connection. The connection from the economizer to the drum represents the flow of feedwater while the connection in the reverse direction represents the flow of flue-gases.

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

Now click on the feedwater icon at the end of the input connector of feedwater-regulator and see what happens. You should notice two things. First, the display switches to feedwater schematic. Second, the boiler icon on output connector from the hp-heater is highlighted with a red band around it. The highlighted boiler icon helps you establish the physical connection between the hp-heater and the feedwater-regulator. Click on the highlighted boiler icon to get back to the boiler schematic. Notice that the boiler schematic now has two feedwater icons highlighted, one connected to the feedwater-regulator you clicked on earlier, and the other to the economizer. This simply means that the hp-heater is connected to both the feedwater-regulator and the economizer in the boiler schematic.

Most components of TURBINIA are uniquely represented in one of the seven schematics. However, there are a few that have multiple representations. For example, the condenser and the hp-heater appear in both the steam and the feedwater schematics. Switch to the steam schematic by selecting the steam icon in the schematic menu and locate the condenser and the hp-heater. Multiple representation of these components in schematics is indicated by feedwater icons adjacent to these components. Notice that these icons do not have a connector attached to them. Click on any one of these two icons and your display will switch to the feedwater schematic. The rectangular boxes marked condenser and hp-heater, in this feedwater schematic, are another representation of the same condenser and hp-heater you saw in the steam schematic.

Troubleshooting for failure indicated by the symptoms at the beginning of the session involves gathering information about system states. You collect information concerning system states using a two-action sequence. The first action of the sequence is called the investigative action. Investigative action enables you to display gauges attached to a component, if any. The second action is the informative action that allows you to access the actual gauge reading. The investigative and the informative actions are now explained with an example.

In the current session you have been asked to detect the failure responsible for decreasing water level in the boiler. It is therefore reasonable to investigate components near the boiler. Click on the boiler icon in the schematic menu to view the portion of the power plant with abnormal behavior. As a part of the process to confirm the symptom indicated, move the cursor over the drum and click on it. You have now taken an investigative action and all gauges attached or relevant to the drum are displayed as a result of this action. This action also highlights the last investigated component, the drum, in blue.

There are three types of gauges in TURBINIA: pressure, temperature, and flow-or-level gauges. These three gauges are represented by icons with letters P, T and L inscribed in them to indicate pressure, temperature, and level respectively. Although there is no visible distinction between the flow and the level gauges, it may be helpful to remember that level gauges are attached to tanks such as the deaerating-feed-tank, fuel-oil-settling-tank, atmospheric-drain-tank, distillate-tank, hotwell, and drum. Furthermore, there is just one gauge that measures flow and is located in the fuel-oil path across the strainer.

The drum you are investigating has all three types of gauges attached to it. There are two pressure gauges, one on the flue-gas connector to the economizer and the other on the steam drum. There is a temperature gauge on the feedwater connector from the economizer and a level gauge on the steam drum. The pressure gauge on the steam drum measures the saturated steam pressure in the drum itself and as such is not located over a connector.

To view the reading of a displayed gauge, you have to click on it. This action of probing a gauge is called an informative action. Select the displayed level gauge on the drum and click on it. First, the drum is lowlighted in yellow-brown and then an icon appears near the gauge. This icon is a qualitative representation of the current level. TURBINIA uses five different qualitative representations of state values. These five are normal, low, slightly-low, slightly-high and high, each represented by an icon as shown in Figure 18.

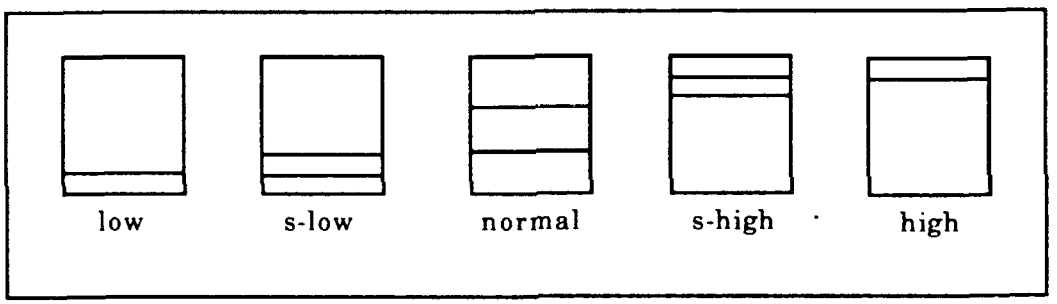


Figure 18. Qualitative State Representation

When you click on the level gauge attached to the drum you should see an icon indicating low level appear at the bottom of the gauge. However, if you see an icon that indicates a slightly-low or normal reading, do not be alarmed. Remember that TURBINIA starts simulating the failure condition at the beginning of the session and if the failure is located far away from the drum, the failure effects will take time to propagate to the drum. Therefore, the low level symptom at the drum, indicated at the beginning of the session, may not yet be visible. You will, however, be able to observe a low level reading during the course of the session.

You should never assume that a gauge reading will be the same at all times after you have observed it. While the gauge readings may change with time, the displayed gauge readings are not dynamically updated. Therefore, you must take an informative action when you need to see the current gauge reading. Thus, if you did not find the drum level low earlier, keep repeating the informative action of selecting the drum's level gauge and you will eventually find it to be low.

You can access any displayed gauge or a gauge reading only until the time you take a new investigative action. Click on the superheater to see what this means. You will discover that all gauges attached to the drum and the probed gauge readings that were visible

disappear. Instead, two new gauges, one pressure and the other temperature attached to the superheater are now displayed in the superheated steam path. Along with the appearance of the gauges, the newly investigated superheater is highlighted in blue.

There are certain components that do not have gauges attached to them. The air-damper is a good example of such a component in the boiler schematic. If you click on the air-damper, all visible gauges and gauge readings on the schematic disappear. Also, the last investigated component is lowlighted and the air-damper is highlighted. However, no new gauges are displayed because there are none attached to the air-damper.

The troubleshooting task typically involves several investigative and informative actions in one or more schematics. Assume that you have conducted several tests and now you have enough evidence to support hypothesis about the failure. Your next valid action, under these circumstances, is to submit a request for conveying the diagnosis. To make this request you click on the diagnose icon in the requests menu. Go ahead and click on the diagnose icon to see how VYASA prepares to accept your diagnosis.

When you click on the diagnose icon, VYASA asks you to select the failed component. This message is conveyed to you through a text appearing in the communications dialog at the bottom of the small screen. You then select the component that, in your opinion, is responsible for the abnormal system behavior. Selecting a failed component is an action identical to investigative action. However, this time, when you click on a component, no gauges are displayed. Instead, if your diagnosis is correct, a message congratulating you appears on the communication dialog. Otherwise, an error dialog accompanied by a beep is displayed over the schematic. This error dialog is shown in Figure 19.

As an example of erroneous diagnose, click on the economizer. The economizer is not the failed component responsible for the current abnormal system behavior. When you complete the click, an error dialog appears over the boiler schematic. This error dialog is a modal dialog. You can close this error dialog by selecting one of the two options available. If you choose "try again" you remain in the diagnose mode and can revise your diagnosis. On the other hand, if you choose "investigate", you are back in the troubleshooting mode where selecting a component displays the gauges attached to it. Click first on "investigate" in the error dialog and then on the drum and you will notice that you are out of the diagnose mode and the four gauges attached to the drum reappear on the screen.

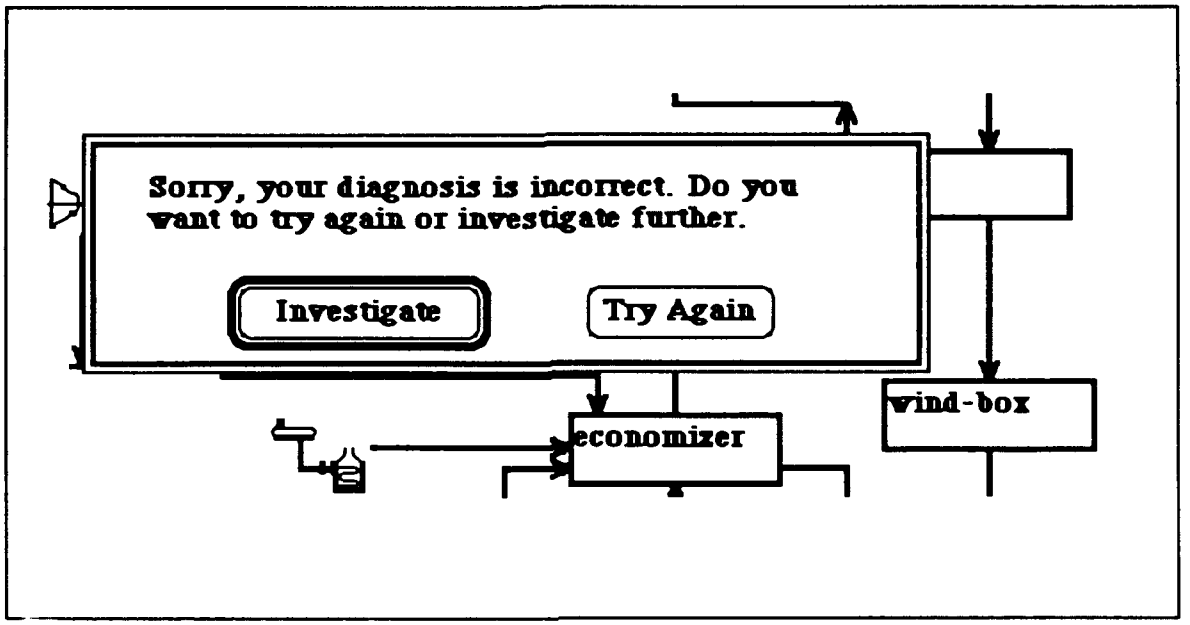


Figure 19. Incorrect Diagnosis

On all schematics you will observe the presence of two other icons that do not represent a schematic and have not also been discussed as yet. One icon appears on the right top corner and the other on the right bottom corner of all schematics. The icon on the right bottom corner is a Georgia Tech. copyright icon. This icon is disabled and has no response. The icon at right top corner is a symptom icon and is used to recall the initial symptoms. Click on this icon and you will see the symptom display dialog reappear. Thus, you can access the ship's initial operating conditions and the initial symptoms at any time. You can once again close the modal symptom display dialog by clicking on the "OK" button.

In this session you have not yet had the opportunity to interact with VYASA- your computer-based tutor. VYASA is capable of functioning in a passive and an active mode, but for your sessions it will only function in the passive mode. You will now explore and experience the capabilities of VYASA in the passive mode.

VYASA in Passive Mode:

Click on the stop icon in the requests menu to halt the simulation and invoke VYASA in passive mode. Notice how the stop and the resume icons change their background colors. The stop icon background changes to tan indicating that it has been disabled. At the same time, the background of resume icon turns gray indicating that it has been enabled. Also notice that the cursor changes shape and turns into a "?". All these changes indicate that you have now invoked the passive tutor and temporarily halted the simulation.

After you have clicked on the stop icon you will also see a *help-levels passive tutor dialog* appear in the top left corner of the large monitor. This dialog box is also shown in Figure 20. This dialog has seven buttons of which two are enabled. The two highlighted buttons indicate the levels of help that the tutor can provide in the passive mode. Starting from the "failure" and "system" buttons, you can explore the entire knowledge-base of the tutor.

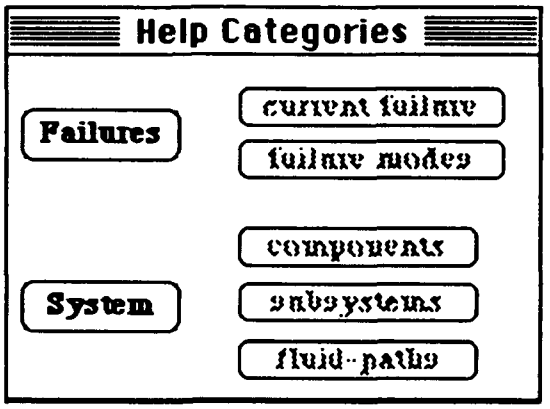


Figure 20. Help-Levels Passive Tutor Dialog

You choose the system button in the help-levels dialog to access knowledge about the system. When you click on the system button, the "components", "subsystems", and "fluid-path" buttons are enabled. These three buttons provide you with further options to select the type of system knowledge description you want to access. When you click on any one of these three buttons, a new passive tutor dialog associated with the selected button appears next to the help-levels dialog. This new dialog also contains several selectable items. You can, by selecting items in the passive tutor dialogs, explore the entire knowledge-base of the tutor at the component, subsystem and fluid-path levels.

Although interacting with passive tutor dialogs is intuitive, it is helpful to remember the following five aspects of interaction:

(1) Clicking on highlighted items (buttons, icons, and text) are valid actions and each action has an associated response.

(2) The response from the tutor usually involves one of the following:
   (a) the appearance of a new dialog box with certain items highlighted;
   (b) the highlighting of lowlighted buttons in the same dialog box to present further options;
   (c) an answer to your query as text in the dialog box or as graphics in the schematics.

   Both (a) and (b) enable you to make your query more specific.

(3) You can click on any highlighted item in any of the displayed dialog boxes to initiate communication with the tutor. Only those boxes that are relevant to the current query are kept open by the tutor.

(4) The context of the information contained in any passive tutor dialog, if unclear, can be gathered from its parent dialog appearing to its left.

(5) Finally, as long as you interact with the passive tutor your cursor will continue to be in the shape of a "?" and investigative actions in the schematics will not be possible. Although an investigative action in this mode will highlight the component, no gauges will be displayed. This response to an investigative action does not necessarily imply that there are no gauges attached to this component.

After you have explored system knowledge to your satisfaction, come back to the help-levels dialog and click on the failures button. You will notice that the tutor closes all the dialogs to the right of help-levels dialog since they are relevant to the system knowledge and not the failures. Furthermore, the two buttons in the help-levels dialog associated with failures are

highlighted and the buttons associated with the system button are lowlighted. The two highlighted buttons are the "current-failure" and "failure-modes" buttons.

Using the failure-modes button you can access information concerning typical system behavior associated with each mode of failure in the liquid and gas paths. Along with the abnormal behavior, the circumstances under which the propagation of abnormal behavior may be curtailed is also provided. Go ahead and click on the failure-modes button and access the tutor's knowledge concerning the four failure modes.

After exploring the tutor's knowledge of the failure modes, click on the current-failure button. The current-failure button brings up a clipboard that extends to the smaller screen on the right. This clipboard presents a summary of observed results from your diagnostic actions. For example, based on the observed gauge readings, the clipboard displays the schematics, subsystems and fluid-paths that contain the affected gauges. The clipboard also displays the most likely mode of current failure if it can be inferred from the tests conducted. The extended portion of the clipboard on the smaller screen displays some of the gauges probed along with their gauge readings. These are mostly those gauges that the tutor considers critical for your diagnostic task.

In the current session you have investigated the steam drum in the boiler-schematic and have found the feedwater level to be low. Therefore, your current clipboard shows the boiler-schematic, the steam-generation subsystem, and the steam path as the affected schematic, subsystem, and fluid-path respectively. The location of the drum's level gauge along with the gauge reading observed are also displayed on the extended clipboard.

Sometimes you may notice that the extended clipboard has a blue colored marker next to the gauge reading. This marker indicates that the reading of that particular gauge has changed since it was last viewed. Once you re-investigate the gauge, the marker disappears and the clipboard is updated to contain the latest information. If you notice a marker appear and then disappear on its own, it simply means that the gauge reading has not stabilized or is perhaps oscillating. You will get a chance to see an example of this a little later.

By now you should be familiar with the capabilities of your computer-based tutor in the passive mode. But since your goal of identifying the failed component has not yet been accomplished, you need to continue investigating. Furthermore, since the time available to solve the problem in a session is limited, you do not want to spend all of it interacting with the tutor. After all, it is your ability to solve problems that will earn you credit and not your ability to query the tutor. Therefore, do not waste any more time and click on the resume icon to get back to the troubleshooting mode. Notice that the background colors of resume and stop icons have reverted to their original colors and the cursor is back in the shape of an arrow. However, all the passive tutor dialogs last displayed are still visible on the screens. Since the clipboard was the last passive tutor dialog displayed in your case, it is still visible even though you are back in the troubleshooting mode.

After returning to the troubleshooting mode, call up the feedwater-schematic and investigate the distillate-tank. You will once again be able to view gauges attached to the investigated component which was not possible when you were interacting with the tutor. Now if you probe the level gauge on the distillate-tank and find it to be abnormal you will notice that this information is posted on the still visible extended clipboard. Even if you find that the reading is normal, you will notice that your diagnostic findings get recorded on the extended portion of the clipboard. This happens because VYASA considers this test as a significant troubleshooting clue. With the passage of time, you will notice that a red marker appears next to the distillate-tank's level gauge reading on the extended clipboard.

This marker will disappear after some time only to reappear a little later. Such a behavior is an indication that the level gauge reading on the distillate-tank is fluctuating.

You can also obtain useful information concerning the current mode of failure from the clipboard. When you have conducted enough diagnostic tests that match the typical abnormal behavior associated with a particular mode of failure, the tutor posts a message on the clipboard. You are informed of the mode of failure that you should suspect based on the test results obtained by you thus far. For example, if for the current failure you investigate the deaerating-feed-tank and find its level to be high, you will notice that "blocked-shut" appears as the most likely mode of failure on the clipboard. This mode of failure is inferred from the two gauge readings you have observed thus far: a high feedwater level in the deaerating-feed-tank and a low feedwater level in the steam drum. Your knowledge of blocked-shut mode of failure should help you infer that the failed component lies between the deaerating-feed-tank and the steam-drum in the feedwater path.

If you are not sure of the components that lie between the tank and the drum in the feedwater path you may want to inquire about it from the tutor. Fluid path information is part of system knowledge. You can seek this information from the tutor by directly selecting the system button in the displayed help-levels dialog instead of invoking the tutor via the stop icon. However, notice that your action has the same effect on the cursor shape and the background colors of stop and resume icons as you would expect when you invoke the tutor via the stop icon. After you have found the fluid path information you were seeking you can get back into the troubleshooting mode by clicking on the resume icon.

By now you have been introduced to all the features of the instructional system. You may freely interact with it for a while till the session ends. If you have any questions, feel free to ask.

In this session you are not provided with the solution to the problem unless you happen to diagnose it on your own. However, in the rest of the sessions the solution will be provided to you at the end of the session. The solution will be supplemented with explanations for abnormal behavior of individual gauges. The explanation should help you form a causal model of fault propagation. Since these explanations can help improve your performance in subsequent sessions, you are recommended to pay attention to them.

This section has introduced you to all your valid interactions at TURBINIA-VYASA interface. You are now familiar with all the features of the computer-based tutor. You will be using this tutor in subsequent sessions to learn the task of troubleshooting a simulated marine power plant. Your valid interactions with the instructional system described thus far are briefly summarized in the next section. Following the summary is a description of how your performance will be measured.

Summary of Valid Actions

Provided below is a list of actions that you will perform while interacting with TURBINIA-VYASA.

Call-for-schematic-action: This is an action you perform to call a new schematic or switch between schematics. There are two ways this may be done. You can either click on an icon in the schematic menu that represents the schematic you want to view, or, if a schematic is currently displayed, click on a similar icon in the schematic itself. If you are investigating components along a suspected path in a schematic that ends up in an icon, you may prefer to use the icon in the schematic itself to switch to a new schematic. By using the icon in the schematic, a highlighted icon in the new schematic properly orients you to continue investigations along the suspected path.

Investigative-action: During your entire period of interaction with the marine power plant simulator, you are either in troubleshooting or in diagnose mode. When in troubleshooting mode, your action of clicking on the mouse button with cursor on a selected component constitutes an investigative-action. You perform investigative-action to view all gauges attached to the input and output sides of the component being investigated. A new investigative-action always makes the displayed gauges and the gauge readings of the last investigated component disappear from the screen. When no gauges are displayed in response to an investigative-action, it implies that the component investigated has no gauges attached to it.

Informative-action: The gauges displayed following an investigative-action, when probed, display the gauge reading. The action of probing displayed gauges by clicking the mouse on the gauge is called an informative-action.

Diagnose-request-action: This action is performed to switch from the troubleshooting mode to diagnose mode. You perform this action when you are prepared to indicate your diagnosis. Clicking on the mouse button after selecting the diagnose icon in the requests menu constitutes a diagnose-request-action.

Diagnostic-action: Diagnostic-action is performed following the diagnose request action. In diagnostic-action you select the component that you suspect is responsible for the observed abnormal system behavior. In indicating your diagnosis, you select the component in the same manner as you do when investigating the component. Thus, diagnostic-action is an investigative-action in diagnose mode.

Modal-dialog-action: Modal dialogs deactivate the mouse in regions outside the dialog box. Before the mouse button can be reactivated for regions outside the modal dialog box, you are required to terminate interaction with the modal dialog. Terminating interaction with a modal dialog requires selecting a button dialog item. The action of selecting the button dialog item in the displayed modal dialog is called modal-dialog-action. Symptom display dialog and error dialogs are examples of modal dialog that require modal-dialog-actions.

Help-request-action: The action of clicking on the stop icon to halt the simulation and invoke the passive tutor is called the help-request-action. Once the help-levels dialog is displayed on the screen you will probably never need to access the passive tutor through the stop icon. After seeking help from the passive tutor the first time you will be able to

communicate with it again by selecting any of the highlighted items in any of the displayed passive tutor dialogs.

Resume-request-action: Every time you initiate interaction with the tutor it is your responsibility to bring the system back to the troubleshooting mode before you can continue with diagnosis. The action of clicking on resume icon to get back to the troubleshooting mode, after you have completed interaction with the tutor, is called resume-request-action.

Tutor-dialog-action: All actions that involve clicking on highlighted items in passive and active tutor dialogs are called tutor-dialog-actions. Most tutor-dialog-actions are taken with the cursor in the shape of a "?".

Measuring Troubleshooting Performance on TURBINIA-VYASA

Although your ultimate goal is to identify the failed component responsible for abnormal system behavior, your performance is affected by other factors. This section will discuss these factors so that you have a better feel for what is expected of you.

Correct diagnosis: Successful fault diagnosis is the most important measure of your troubleshooting ability. However, since the problems are tough, your inability to solve problems has to be evaluated in conjunction with other factors.

Troubleshooting time: The total amount of time taken for troubleshooting is an important performance measure for those who successfully solve the problem. Those who solve the problems in less time have a better performance rating.

Number of relevant actions: Even though every informative action has some informational content, some have more relevance than others for the failure being investigated. Also, there is a minimum number of relevant informative actions necessary to diagnose each failure. The number of relevant informative actions past this minimum number taken to solve a problem is a measure of diagnostic performance. Smaller number of relevant informative actions required to correctly diagnose the fault implies better performance.

Number of irrelevant actions: The informative actions that have no relevance to the current problem are said to be irrelevant. The larger the number of such irrelevant informative actions during your troubleshooting exercise, the worse is the diagnostic performance.

Number of incorrect diagnosis: You are penalized any time you make an incorrect diagnosis. However, the penalty depends upon the component incorrectly identified as failed. At any stage during the troubleshooting process there are likely candidates for failed component based on the observed abnormal system states. The likelihood that a component may have failed increases or decreases as you conduct more diagnostic tests. Selecting a likely component as the cause of abnormal system behavior does not penalize you as much as picking a component that cannot have failed. Therefore, even though your performance is adversely affected by an incorrect diagnosis, it is considered worse if the suspected component cannot have failed based on the symptoms at the time you express the diagnosis.

Investigation of unaffected schematics: For each failure, there are only few schematics, subsystems and fluid paths that are affected. Affected schematics are those schematics that have gauges with abnormal readings. Investigating components in schematics that are unaffected by the failure reflects the inability on your part to relate the symptoms to the correct structural location of the power plant. Thus, investigating components in unaffected schematics reduces your performance rating. Continuing to investigate unaffected schematics in spite of the guidance provided by the tutor makes the performance worse.

Investigation of unaffected subsystems: Like the schematics, investigating components in subsystems unaffected by failure reduces your performance rating and repeating the mistake in spite of the tutor's guidance makes it worse.

Investigation of unaffected fluid paths: Once again, like the previous two factors, investigating components in unaffected fluid paths harm your performance and repeating the mistake in spite of the tutor's guidance makes it worse.

This manual has guided you through your first session, made you familiar with TURBINIA-VYASA, and has described how your performance will be measured during the experiment. From the next session, you will begin your formal training. VYASA will help you to learn to troubleshoot marine power plants efficiently.

Since it is vital for my experimental results, you are requested not to discuss any aspect of this experiment with other subjects.

GOOD LUCK!

OPERATOR INSTRUCTIONS FOR TURBINIA-VYASA (ACTIVE MODE)

TURBINIA-VYASA is an instructional system that trains operators to troubleshoot marine power plants. TURBINIA is the name of the simulated marine power plant used in the instructional system and VYASA is the computer-based tutor that teaches the troubleshooting task using TURBINIA. As a naval trainee, you will be trained to diagnose some common failures in a marine power plant using this instructional system. This instructional manual will describe your interaction with both TURBINIA and VYASA following a brief description of a typical marine power plant and its control system.

Introduction

A marine power plant is a collection of components configured to produce mechanical work from thermal energy. This energy transformation takes place in components called the *turbines*. A ship that uses steam as a medium to carry the thermal energy to the turbines is said to be steam-driven. In a steam-driven ship the source of thermal energy is usually fossil or nuclear fuel. This section describes the functioning of a fossil fuel-oil fired, steam-driven marine power plant.

The process of producing mechanical work in a steam-driven marine power plant can be decomposed into several stages. Each stage is associated with one of the four phases in the steam cycle: generation, expansion, condensation and feed.

Steam Generation

Figure 1 shows the configuration of components in the generation phase of the steam cycle. This phase of the steam cycle takes place in the boiler. The boiler is comprised of *tubes* and a *steam drum*. The boiler tubes contain water that is heated by flue-gases resulting from the fuel burned in the *furnace*. This heat transfer is by conduction through the tube walls. Heating of water in the tubes produces steam. This steam accumulates over the water surface in the steam drum and is called *saturated steam*. Saturated steam is sometimes also referred to as wet steam because of its moisture content.

Continuous steam generation in the boiler increases the steam pressure in the drum. Boilers are rated by the steam pressure they can handle in the drum. In a 1200-psig boiler, for example, the maximum steam pressure permitted is 1200-psig. A safety valve is activated to release pressure whenever it exceeds the maximum value.

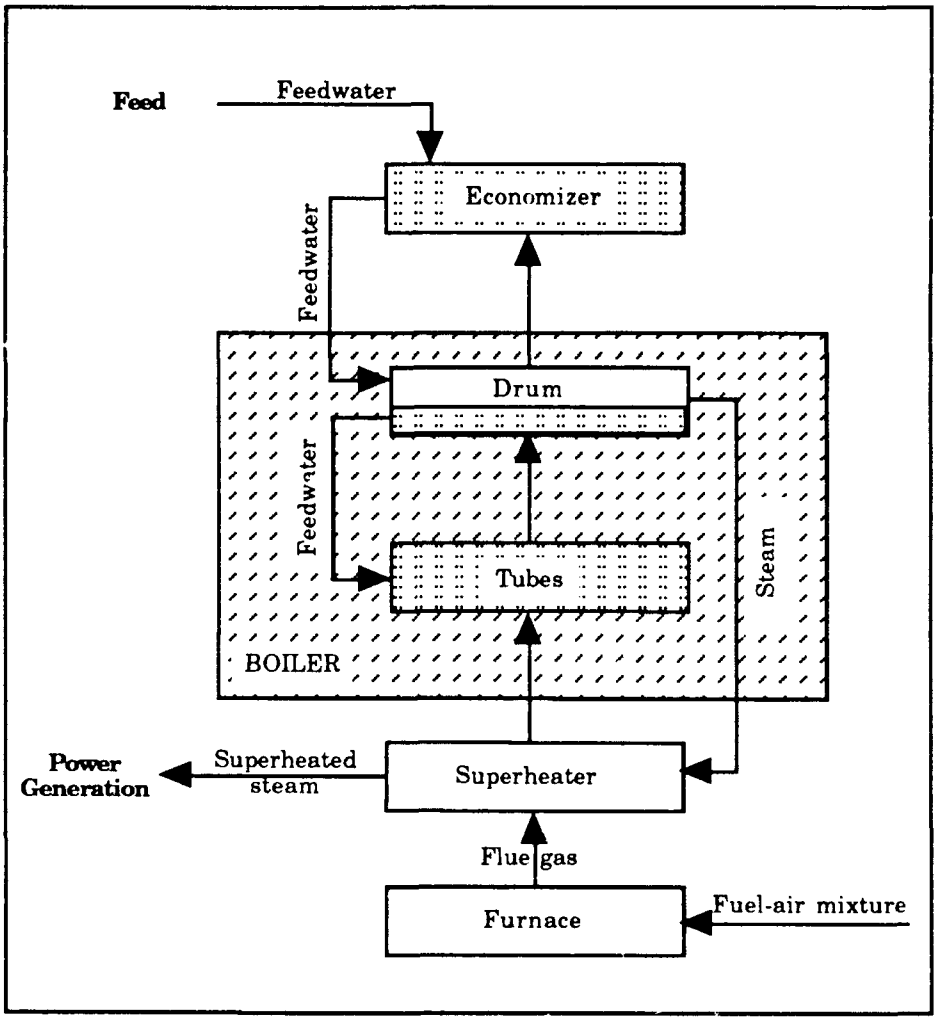


Figure 1. Steam Generation

The steam pressure in the drum controls the temperature at which the water boils in the drum. Since the temperature of saturated steam accumulating above the water surface is the same as the temperature of the water, the saturated steam temperature depends upon the steam pressure. This temperature at which the water and saturated steam coexist in the drum is called the saturation temperature. The highest possible saturation temperature is attained in a boiler when the boiler operates at its maximum rated pressure. Since the thermal energy of steam in the drum is proportional to its saturation temperature, the heat content of the saturated steam is maximum at the highest boiler operating pressure.

Even though the boilers are designed for high operating pressures, the saturated steam does not contain enough thermal energy to operate the turbines at their best efficiency. Thermal energy of steam is increased by passing it through tubes in the section of the boiler closest to the furnace. This section of the boiler is commonly known as the *superheater*. The superheater is responsible for adding heat to saturated steam at constant pressure. The heat added to saturated steam in the superheater is called sensible heat. Sensible heat increases

the temperature of the steam beyond the saturation temperature and makes it drier. The steam from the superheater is called superheated steam and the increase in steam temperature in the superheater measures the degree of superheat.

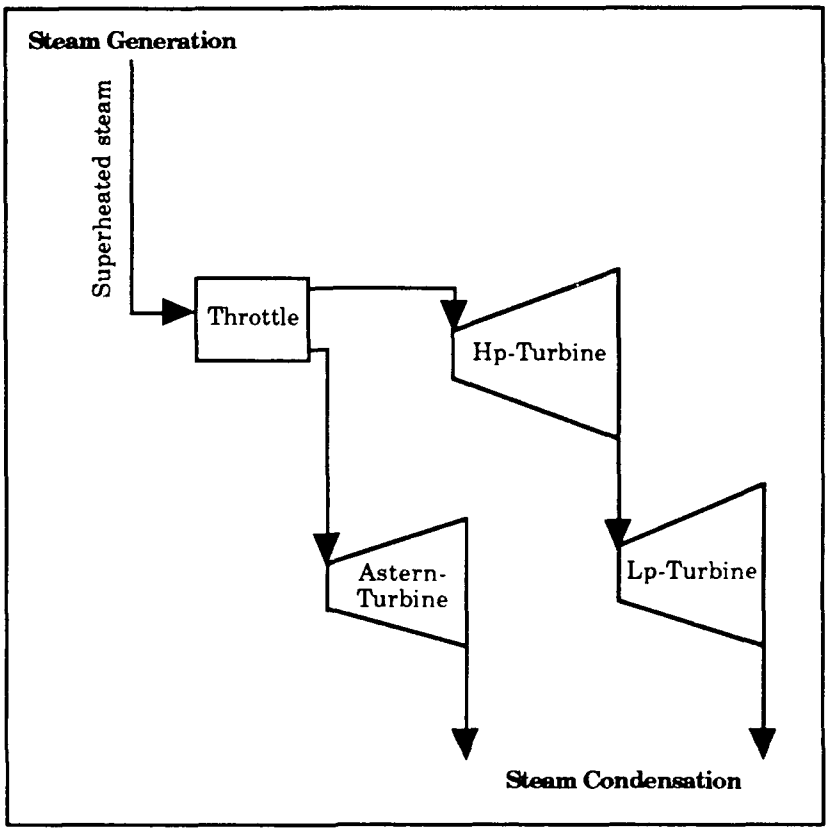


Figure 2. Steam Expansion

Steam Expansion

The second phase of the steam cycle takes place in two steps. First, the superheated steam from the boiler expands in a *high pressure turbine* to convert thermal energy to mechanical work. Then, since the steam still contains a considerable amount of thermal energy, it is expanded further in a *low pressure turbine* connected to the exhaust of the high pressure turbine. Figure 2 shows the arrangement of low and high pressure turbines in a power plant.

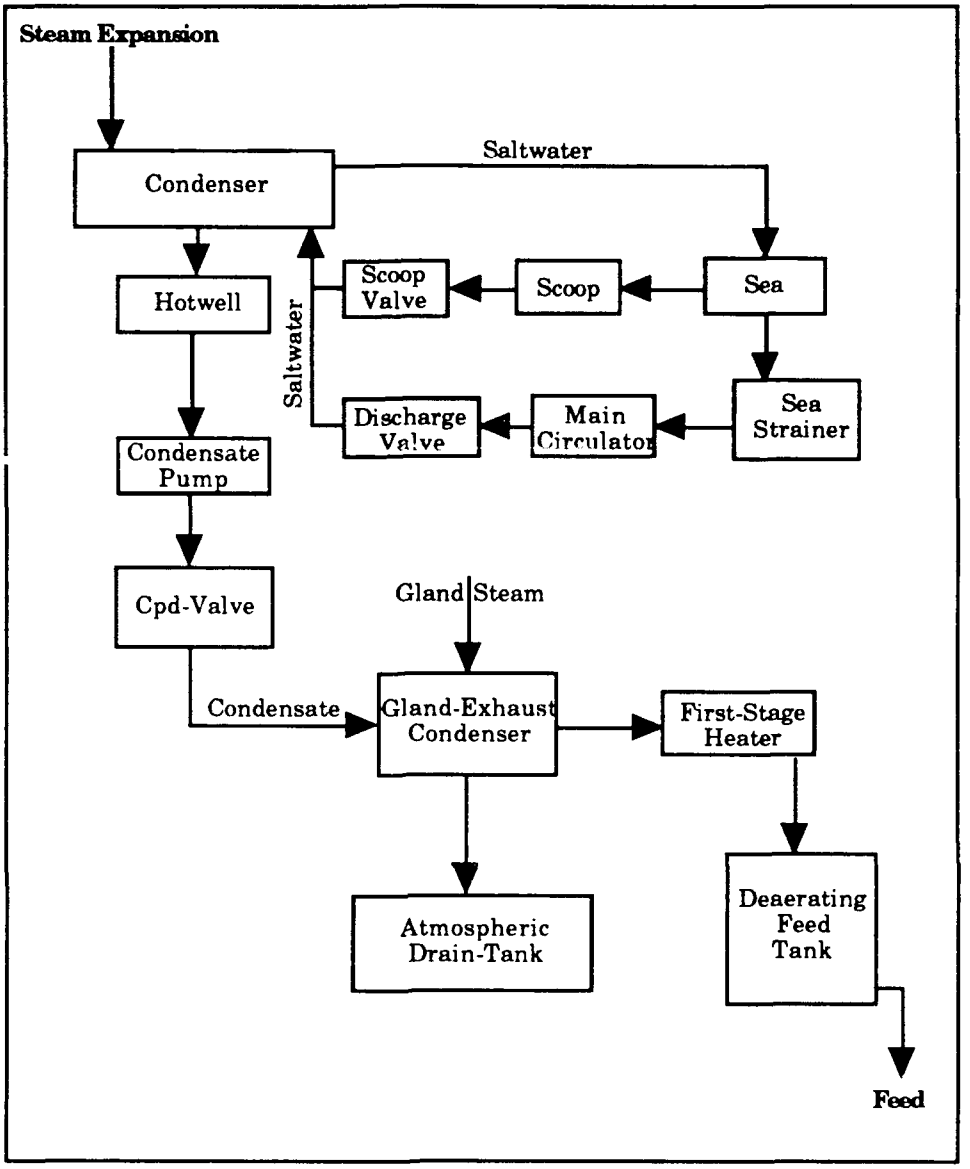


Figure 3. Steam Condensation

Steam Condensation

After expansion, the third phase of the steam cycle is steam condensation which takes place in the main *condenser* (Figure 3). The condenser is a sealed container with tubes that carry cold sea water. When the steam passes over these tubes it loses latent heat to the cold water. When sufficient latent heat is withdrawn from the steam, it changes phase and turns back into water, called *condensate*.

Steam pressure at the turbine exit is low and steam can flow into the main condenser only if the pressure in the condenser is lower. Since the condensate occupies less volume than the same amount of steam and because the condenser is a sealed container, condensation creates a vacuum in the condenser shell. This vacuum in the condenser shell helps maintain a continuous flow of steam from the turbines to the condenser.

As the steam from the turbines turns into condensate, it flows into a collecting tank called the *hotwell*. The *condensate-pump* then pumps the condensate to the *deaerating-feed-tank* via the *gland-exhaust-condenser*. In the gland-exhaust-condenser, the condensate from hotwell serves as the cooling medium to condense steam from the turbine glands. While the condensate from the gland-exhaust-condenser flows to the deaerating-feed-tank, the condensed gland steam is returned to the condensate system by way of *atmospheric-drain-tank*.

Feed

Feed, the last phase of the steam cycle, begins at the deaerating-feed-tank. The deaerating-feed-tank is a storage tank for feedwater. It also contains apparatus to remove dissolved oxygen entrained in the condensate. The other major components in the feed phase are the main feed pump, the feed water regulator and the economizer. These components are shown in Figure 4. The main feed pump is responsible for pumping water to the boiler. The feed water regulator regulates feedwater into the economizer enroute to the boiler. The economizer is a heat exchanger that preheats the feedwater.

Each of the four phases of the steam cycle perform an important system function. The collection of components responsible for the function constitute a functional subsystem. Thus, **steam generation**, **steam expansion** or **power generation**, **steam condensation**, and **feedwater preheating** are also essential *subsystems* of a marine power plant. In addition, a power plant typically has subsystems that perform other functions necessary for its operation. **Combustion**, **auxiliary steam use**, **control air**, **lube oil**, and **saltwater service** are examples of such subsystems.

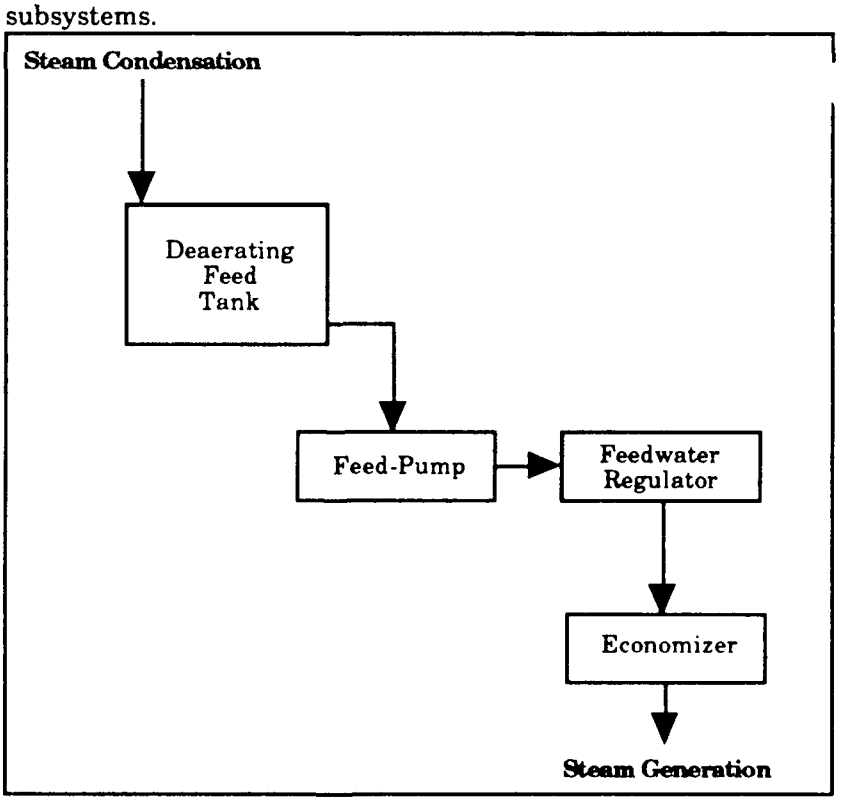


Figure 4. Feed

Combustion involves burning the fuel-air mixture prepared in the *burner*. The thermal energy released during combustion is used to heat water in the boiler. The components that make up the combustion subsystem are shown in Figure 5. These components lie along two fluid paths: *combustion air* and *fuel-oil*.

Combustion air is supplied to the burner by a forced draft fan operated by either a steam turbine or an electric motor. Fuel-oil is supplied to the burner by pumping fuel from a *settling-tank*. For proper combustion, both the combustion air and the fuel-oil need to be at the proper pressure and temperature. Furthermore, for complete combustion the mass of air required is fourteen times the mass of fuel-oil.

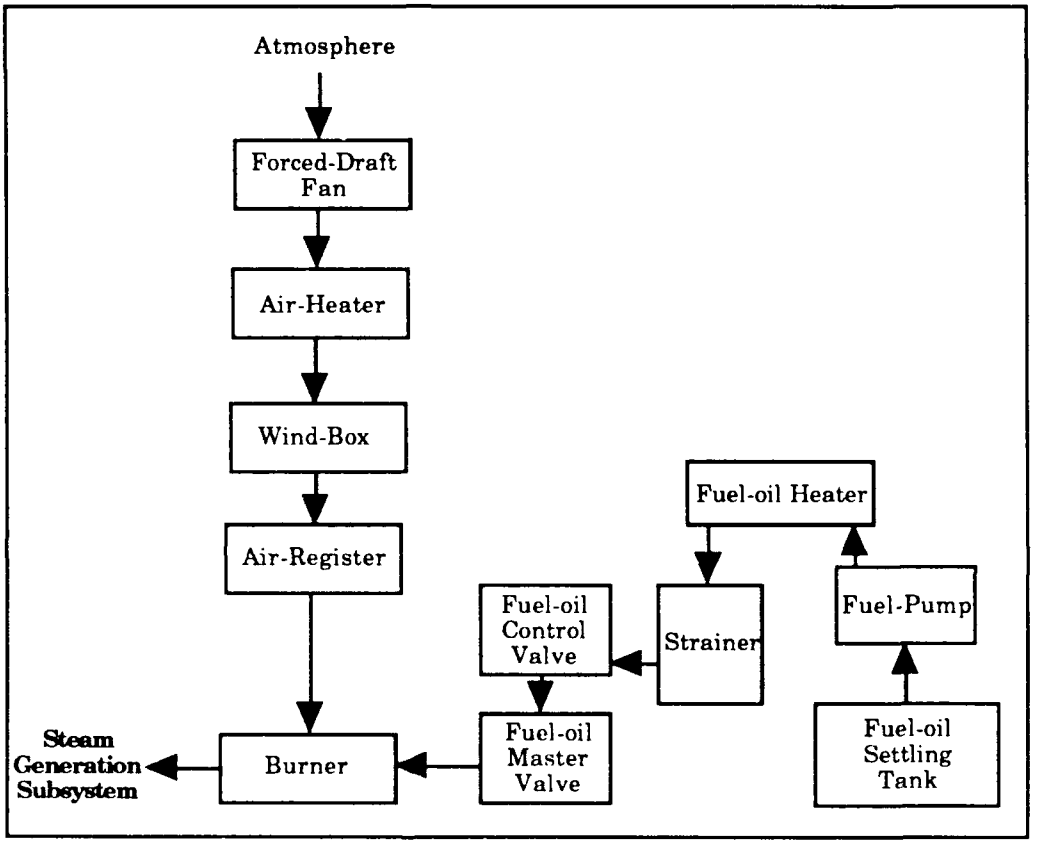


Figure 5. Combustion Subsystem

Incorrect quantity or improper heating of either the combustion air or fuel-oil causes combustion problems. Inadequate quantity or insufficient heating of combustion air and excessive flow of fuel-oil or insufficient heating of it causes incomplete combustion. Incomplete combustion causes dark smoke in the boilers. On the other hand, excess quantity of combustion air in the fuel-air mixture either due to increased flow rate of air or reduced flow rate of fuel-oil extinguishes the flame in the furnace. Excessive preheating of either the combustion air or the fuel-oil causes yet another combustion problem called preignition. In preignition, the fuel starts to burn before it reaches the burner.

The auxiliary steam use subsystem shown in Figure 6 uses *desuperheated steam* for various purposes. Desuperheated steam, unlike the superheated steam, is low pressure steam. Desuperheated steam is obtained by passing superheated steam through the *desuperheater*. Low pressure desuperheated steam is used (1) by the auxiliary power units

to run equipment such as the *feedwater-pump, fuel-pump, saltwater-service-pump* and the *forced-draft -fan*; (2) to preheat the fuel-oil and the feedwater in the deaerating-feed-tank; and (3) by the the gland seals to prevent leakage of air into the turbine casings and steam leakage out of the casings.

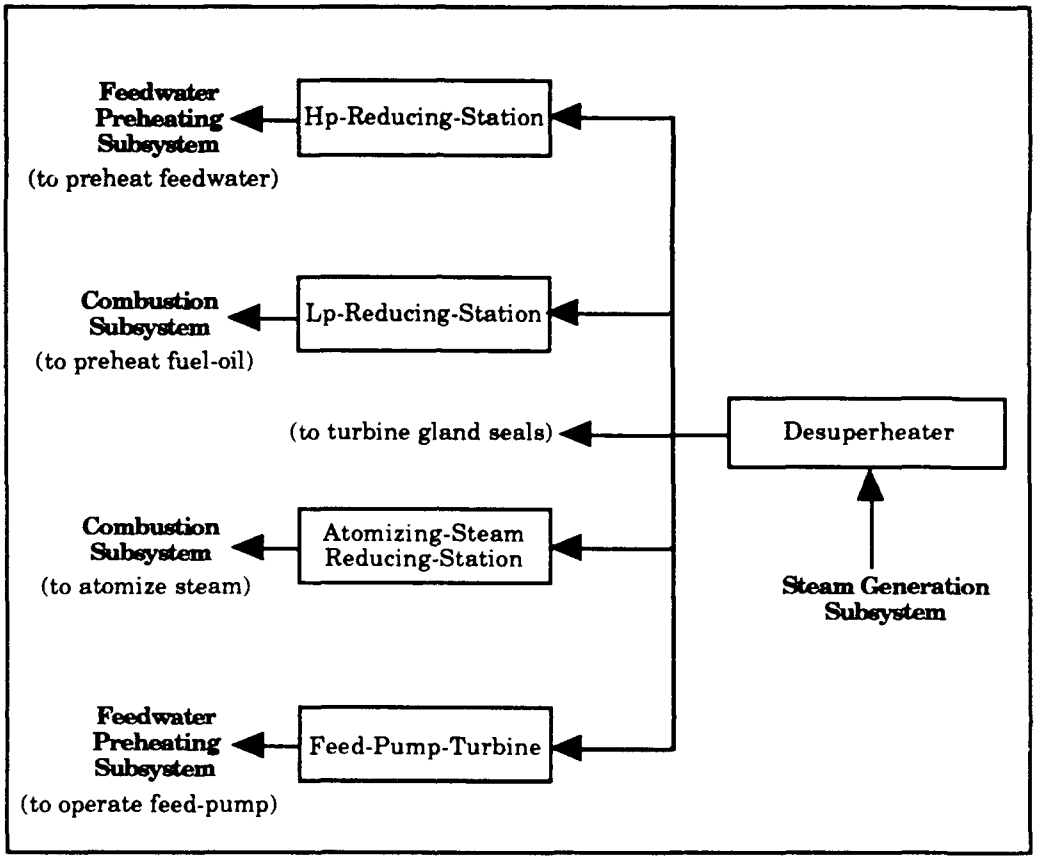


Figure 6. Auxiliary Steam Use Subsystem

The control air subsystem is responsible for distributing control air to many valves and regulators. These valves and regulators are operated by the control air. Figure 7 shows the components that constitute the control air subsystem of a marine power plant.

The lubrication subsystem has the primary purpose of lubricating moving parts and removing the heat produced by friction. The subsystem consists of a pump that draws lube-oil from the *oil-sump* and distributes it to those components that need lubrication. Figure 8 shows the components in the lubrication subsystem.

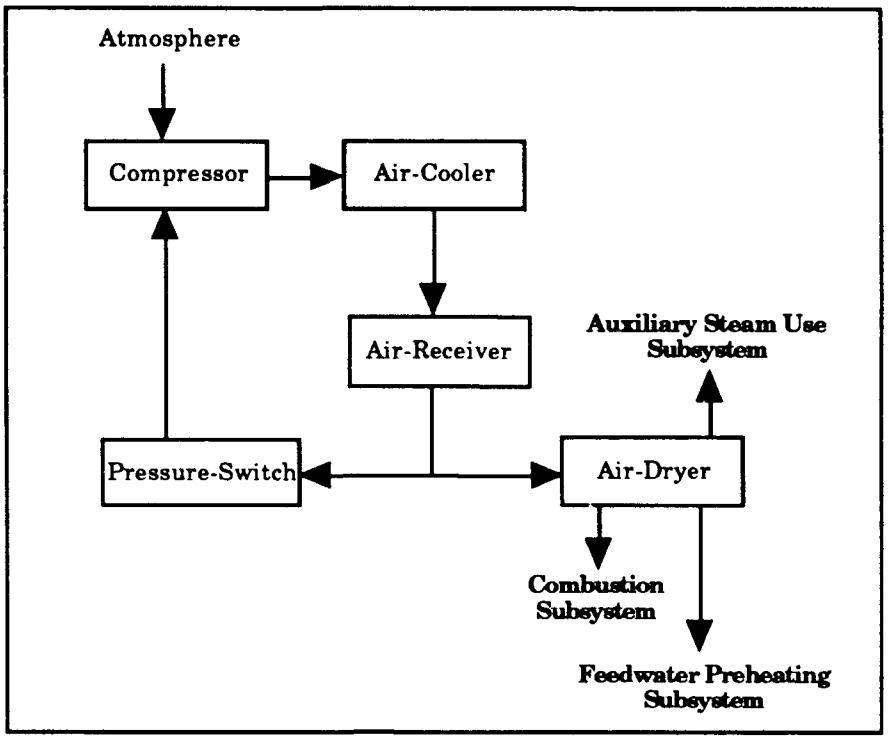


Figure 7. Control Air Subsystem

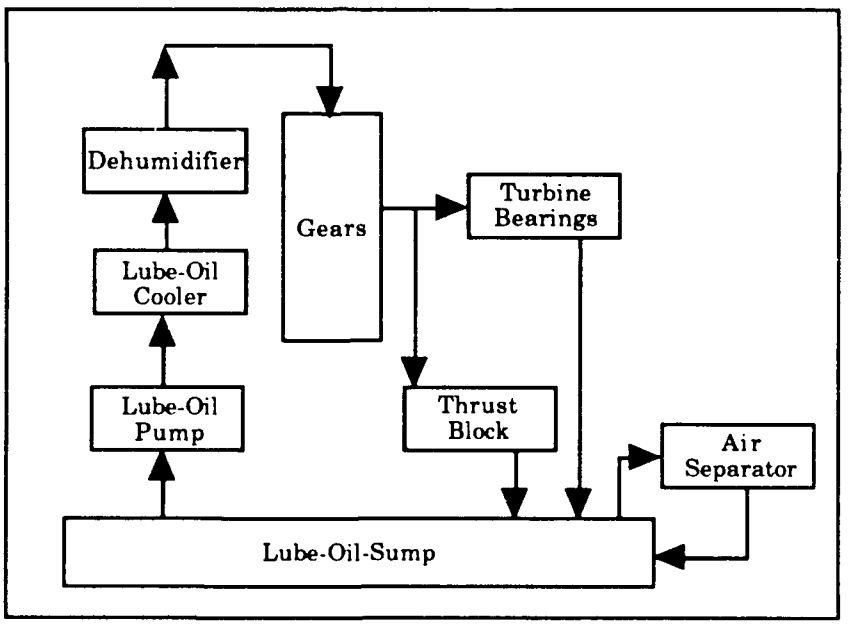


Figure 8. Lubrication Subsystem

The saltwater service subsystem distributes the cold sea water to remove heat from units dissipating heat. Figure 9 shows sections of the power plant cooled by the saltwater.

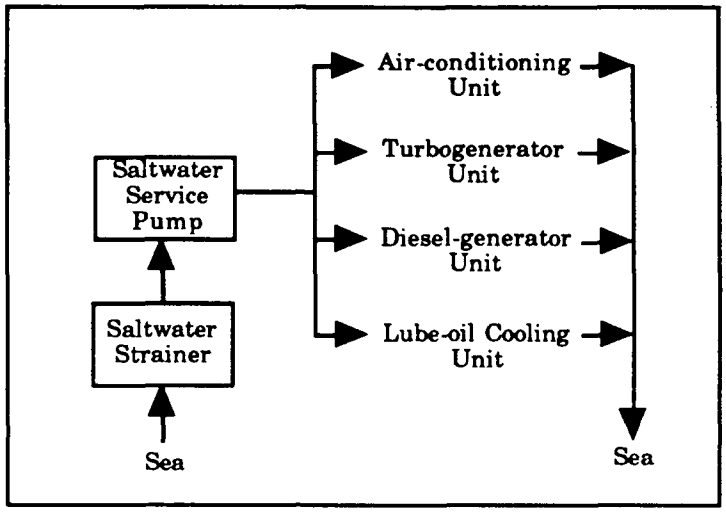


Figure 9. Saltwater Service Subsystem

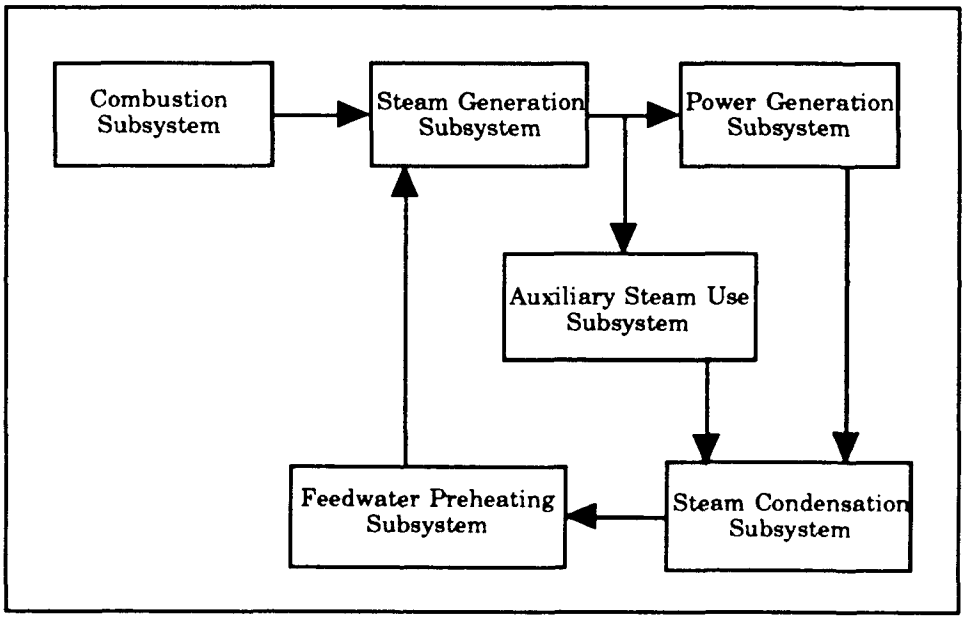


Figure 10. Interacting Subsystems of Marine Power Plant

This section described the decomposition of a marine power plant into nine functional subsystems. It also described the role each subsystem plays in achieving the overall goal of producing power. The interaction between the subsystems to produce power is summarized in Figure 10. For constant power supply, the operating conditions for these subsystems can be set to safely meet the demand. However, the demand for power in a ship is never constant but varies with load. Control systems manage the power plant so that it can satisfy the changes in the demand for power due to fluctuating load. The boiler control system is the most important among all control systems in a marine power plant. The boiler control system of most modern Navy vessels is sophisticated and needs minimum human intervention. Some components of the automatic boiler control system (ABC) are described next.

Automatic Boiler Control System

Navy vessels typically have the following three ABC systems: automatic combustion control (ACC), feedwater control (FWC), and makeup and excess feed control systems. These control systems perform the functions of measuring, comparing, computing and correcting. In each control system, a state value of interest is measured; compared to a desired value; a new operating condition, if necessary, is computed; and finally a correction made in the operating conditions to reduce the deviation between the measured and the desired value of the state.

Automatic Combustion Control System. The function of the automatic combustion control system is to maintain the boiler drum pressure at a constant value during steady and changing load conditions. The ACC system accomplishes this task by

(1) constantly measuring the steam drum pressure and combustion air flow;
(2) comparing the steam drum pressure to the specified designed value;
(3) computing the amount of change, if any, in the furnace combustion; and
(4) correcting furnace combustion as needed.

When the steam demand on the boiler is increased, the steam drum pressure decreases because the rate of steam withdrawal from the drum becomes greater than the rate of steam production in the boiler. This pressure drop is sensed by the ACC system and an increase in furnace combustion is computed to meet the increase in the demand for steam. Computing the increase in furnace combustion involves computing the increase in the supply of combustion air and a proportionate increase in the supply of fuel-oil to assure complete combustion. The ACC system controls the combustion air flow by regulating the supply of steam to the forced draft fan turbine and controls the fuel-oil flow by positioning the main-fuel-oil-control-valve. The measurement of air flow provides the ACC system with the feedback necessary to perform this function.

Feedwater Control System. The function of the feedwater control system is to maintain a constant water level in the steam drum. The FWC system automatically does this by

(1) measuring the steam drum water level and the feedwater flow rate to the boiler;
(2) comparing the measured water level in the drum to a designed value;
(3) computing the required change, if any, to the rate of feedwater flow; and
(4) correcting the feedwater flow rate as needed.

When the load is steady, the feedwater flow rate into the boiler equals the rate of steam consumption and the water level in the steam drum is normal. But, when the load changes, so does the demand for steam. Any change in this demand is detected and the feedwater flow rate is increased or decreased to equal the steam flow rate out of the boiler. The actual control of feedwater flow is accomplished by adjusting the air-operated diaphragm of the feedwater regulator between the feed pump and the boiler.

Makeup and Excess Feed Control System. Operation of a steam-driven power plant often requires the addition or removal of water from the steam cycle. The makeup and excess feed control system is responsible for doing this and for maintaining a specified level of feedwater in the deaerating-feed-tank.

Whenever the level in the deaerating-feed-tank deviates from the specified value, water is either withdrawn from or added to the deaerating-feed-tank. In both cases the process is facilitated by two standby tanks. The two standby tanks are the atmospheric-drain-tank

and the distillate-tank. When the feedwater level in the deaerating-feed-tank falls below normal, the *makeup-feed-regulator* is adjusted by the control system to increase flow from the standby tanks. Increased flow into the deaerating-feed-tank compensates for the loss in the feedwater level. Similarly, a *deaerating-dump-regulator* is activated by the control system to withdraw excess feedwater from the deaerating-feed-tank when the level in the tank rises above the normal value.

In addition to the automatic boiler control system, a power plant has several other controls which are not discussed here because they are not relevant to your task. However, knowledge concerning some common modes of failure in components of a power plant is useful for diagnosing faults and is described next.

Common Modes of Failure

A mechanical component in a physical system like the marine power plant can fail in more than one way. The four most common modes of failure for components of TURBINIA are: (a) blocked-shut, (b) stuck-open, (c) leak-in, and (d) leak-out. Faults in components fit one or more of these four mode types.

A blocked-shut component offers greater than normal resistance to the flow of fluid for the desired operating condition. A valve that cannot position its vane to a larger opening demanded by the new operating condition or a strainer that is clogged with dirt are examples of the blocked-shut mode of failure.

A stuck-open component offers less than normal resistance to the flow of fluid for the desired operating condition. A valve that refuses to position its vane to a smaller opening on command is an example of the stuck-open mode of failure.

A component failed in leak-in mode allows undesirable or excess flow of fluid *into* it, while a leak-out mode of failure causes undesirable passage of fluid *out* of the component. A vacuum tank that allows air to leak in from outside and a ruptured piping that allows the fluid it carries to leak out from it are examples of leak-in and leak-out modes of failure respectively.

Each failure mode is responsible for a system behavior that manifests in the form of a typical pattern of abnormal state values. During diagnostic problem solving, it is often helpful to identify the failure mode from system behavior and confine the search to components that fail in the identified mode. The typical system behavior associated with a fault also depends upon the phase of the fluid in the affected path. The following set of examples explains the abnormal system behavior for each of the four modes of failure in liquid and gas paths.

A blocked-shut mode of failure in a liquid path causes the liquid *level* downstream to be lower than normal and the level upstream higher than normal. A similar blocked-shut mode of failure in a gas path, on the other hand, decreases the downstream gas *pressure* and increases the upstream pressure.

A stuck-open mode of failure in a liquid path causes the liquid *level* downstream to be higher than normal and the level upstream lower than normal. A similar stuck-open mode of failure in the gas path increases the downstream gas *pressure* and decreases the upstream pressure.

When a container that stores liquid allows more of the same liquid to leak in, the *level* of the liquid in the container increases. When the same container stores gas and allows more of it to leak in from the high pressure surroundings, the *pressure* in the container becomes abnormally higher.

A ruptured component that allows liquid to leak out causes a drop in the liquid *level* upstream as well as downstream from the place of leakage. A similarly ruptured component carrying gas causes a drop in *pressure* upstream and downstream from the place of leakage.

Although there is a typical system behavior associated with each mode of failure, it is not always easy to observe the abnormal behavior in a real system. This is due to the limited number of available gauges. Therefore, pressures, temperatures, and flows cannot be measured across every component. Furthermore, certain components can prevent propagation of expected abnormal behavior past them. For instance, a source-sink such as a deaerating-feed-tank located downstream in the blocked-shut condensate path prevents further propagation of low level downstream from the tank. The deaerating-feed-tank imposes such a behavior on the system because it is an infinite source of feedwater which temporarily compensates for any loss of water level. A summary of typical system behavior associated with the four failure modes is shown in Table 1.

Failure Mode	Fluid	State	Abnormal Behavior Upstream	Abnormal Behavior Downstream	Propagation Limited By Upstream	Propagation Limited By Downstream
Blocked-Shut	Liquid	Level	High	Low	Infinite Sink	Infinite Source
	Gas	Pressure	High	Low	Safety Valve	Infinite Source
Stuck-Open	Liquid	Level	Low	High	Infinite Source	Infinite Sink
	Gas	Pressure	Low	High	Infinite Source	Safety Valve
Leak-In	Liquid	Level	High	High	Infinite Sink	Infinite Sink
	Gas	Pressure	High	High	Safety Valve	Safety Valve
Leak-Out	Liquid	Level	Low	Low	Infinite Source	
	Gas	Pressure	Low	Low		

Table 1. Typical Abnormal System Behavior

This completes a description of a typical marine power plant, its control systems, the four common modes of failure in components of the power plant, and the typical abnormal system behavior associated with each of the four failure modes. The next section describes TURBINIA's interface.

The Interface

TURBINIA- the marine power plant simulator, and VYASA- the computer-based tutor have been developed on a dual screen Apple Macintosh II workstation. The dual screen configuration consists of one 19" color monitor and a 13" color monitor. In this set up, the larger monitor is the left screen and the smaller monitor is the right screen. A single button computer mouse that can point to all locations on both screens is the only input device. You will use this mouse to interact with the direct manipulation interface of both TURBINIA and VYASA. Almost all your actions involve moving the mouse cursor to a desired location and clicking on the mouse button. All valid user actions have appropriate response while invalid actions are ignored by the system. Valid actions at the joint TURBINIA-VYASA interface are described in detail later.

The interface to TURBINIA-VYASA consists of seven schematic windows, a schematic menu, a requests menu, a hypothesis menu, a communication dialog, multiple levels of hierarchically organized passive tutor dialogs, a symptom display dialog and several error dialogs.

The seven *schematics* display the physical connections between the components of the power plant. You will use these schematics to investigate components and probe gauges attached to these components.

The *schematic menu* displays seven icons each representing one of the seven schematics. You will use these icons to access the schematics.

The *requests menu* has three icons. You will use the first icon to request for an opportunity to diagnose the fault, the second to temporarily halt the simulation and the third to resume the simulation.

The *hypothesis menu* has four items. You will use the hypothesis menu to communicate with your computer-based tutor VYASA. The first item "View" is used to review the failure hypotheses that you have provided to the tutor. "Add" and "Delete" are used to modify hypotheses. "Advice" provides assistance from the tutor.

The *communication dialog* is used by VYASA to provide instructions.

The *passive tutor dialogs* establish your communications with VYASA when you seek knowledge concerning the structure, function and behavior of the subsystems and the components. You will use the passive tutor dialogs to explore the tutor's knowledge-base.

The *symptom display dialog* shows the initial symptoms observed at the time you begin your troubleshooting task.

The *error dialogs* convey appropriate messages when you make a mistake.

The display of *error dialogs*, *symptom display dialog*, or new text on the *communication dialog* is accompanied by a beep.

A more detailed description of the interface and valid forms of your interaction with the system follows. You will now be given a guided tour of your first session with the instructional system.

A session with TURBINIA-VYASA

Welcome to your first session with TURBINIA-VYASA. You will soon be troubleshooting a simulated failure in a marine power plant. You will be aided in your task by the computer-based tutor VYASA. This computer-based tutor functions in two modes: passive and active. In the passive mode, the tutor only responds to your queries while in the active mode it intervenes on its own to provide you with instructions. Your first session has been designed to make you familiar with the joint interface of TURBINIA and VYASA. This first session will be short containing a single problem. Subsequent sessions will be of 45 minutes each and will require you to solve three problems. Use the instructions in this section to guide yourself through the first session.

At the beginning of every session, the dual screen Apple Macintosh II workstation displays three menus on the large screen and two dialog boxes on the small screen. The three menus on the large screen are the *schematic menu*, the *requests menu* and the *hypothesis menu*. A *communications dialog* is displayed on the bottom edge of the small screen and an *output file path dialog* is displayed above the communications dialog. This display of the two screens at the start of every session is also shown in Figure 11. If you are starting your first session now, make sure that the screens in front of you look like Figure 11.

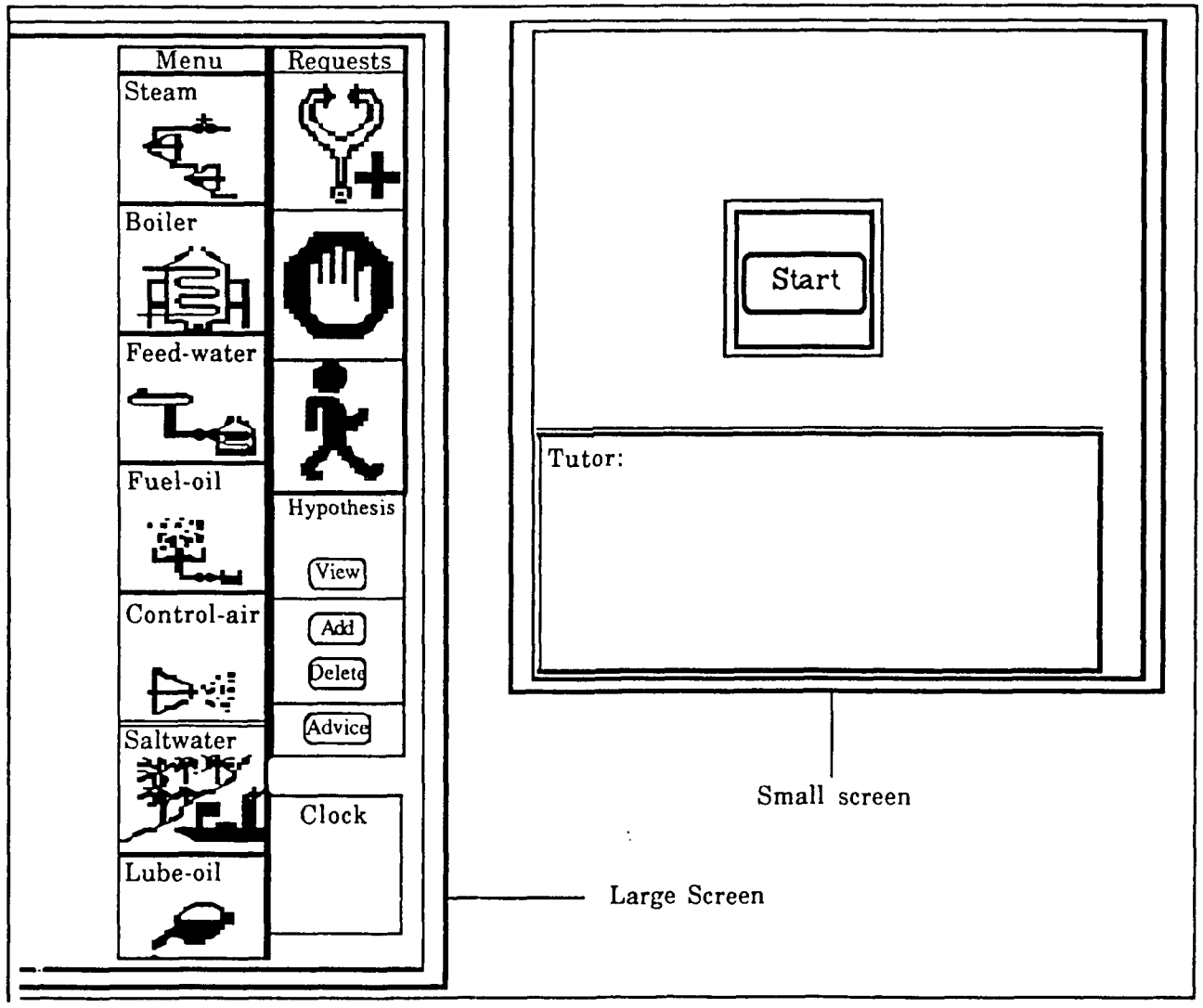


Figure 11. Configuration of Screens

Schematic menu:

The schematic menu on the large screen and also shown in Figure 12, displays seven icons. Each icon represents one of the seven schematics of the simulated power plant. The names of the seven schematics are also provided in the textual form above each icon. The seven schematics are the steam, boiler, feed-water, fuel-oil, control-air, saltwater and lube-oil. You can access any of the seven schematics by clicking on the icon representing the schematic.

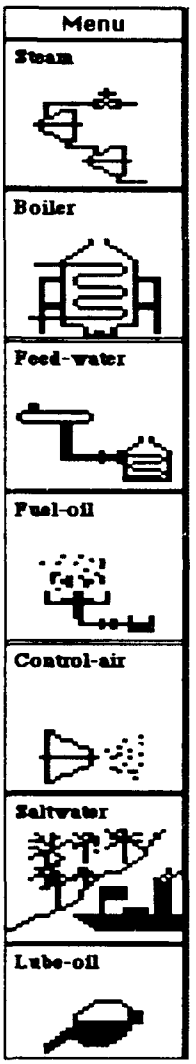


Figure 12. Schematic Menu

Requests menu:

The requests menu adjacent to the schematic menu displays three icons (Figure 13). The first is the diagnose icon. You click on diagnose icon when you have sufficient evidence to confirm your failure hypothesis. By clicking on the diagnose icon, you indicate to the instructional system your intention to identify the component responsible for the observed

abnormal behavior. You are later provided with an example of how to use the diagnose icon.

The second icon on the requests menu is the stop icon. A click on the stop icon can halt the simulation putting you in a mode to interact with the passive tutor. Your interaction with the passive tutor is later described in detail.

The third icon on the requests menu is the resume icon. The resume icon is used to restart simulation after it has been halted to communicate with the passive tutor. Since you are currently not interacting with the passive tutor, the resume icon is shown disabled. All disabled icons in this application have an inverted-gray or tan colored background as compared to enabled icons that are shown in gray.

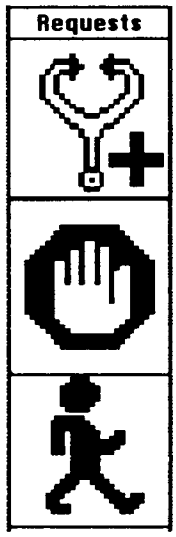


Figure 13. Requests Menu

Communication dialog:

The computer-based tutor VYASA communicates with you through textual messages and instructions presented on the communication dialog. This communication dialog is displayed on the bottom edge of the small monitor (Figure 14). All communications through this dialog box are accompanied by a beep.

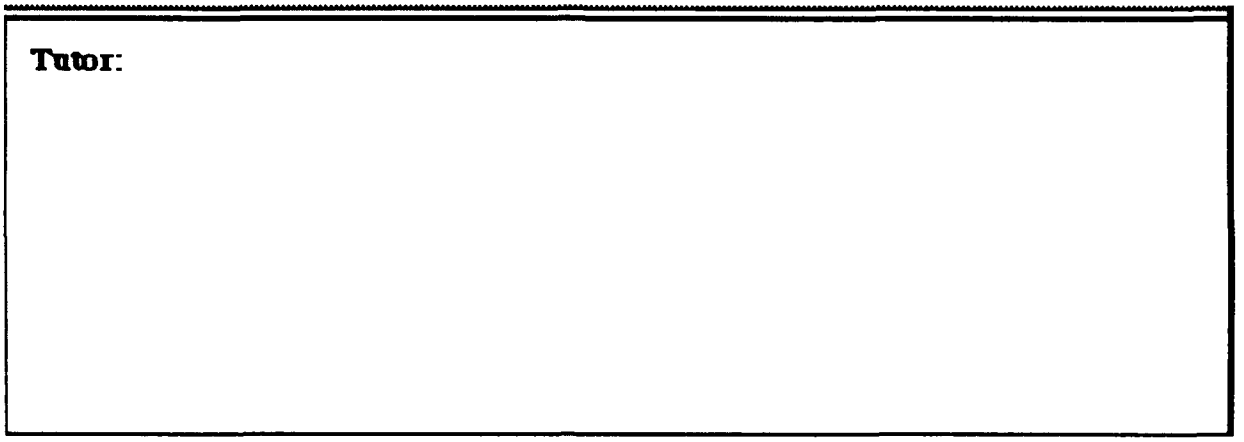


Figure 14. Communication Dialog

Output file path dialog:

Output file path dialog is displayed above the communication dialog on the small screen at the beginning of each new session. This dialog is also shown in Figure 15. Output file path dialog expects you type in a name of the file to store your performance data. You begin every session by typing in your last name to create this file. As you type in, you should see the characters appear in the editable text region bounded by a rectangle in the output file path dialog. When you are done typing in your name, check the spelling. If you have made a spelling error, use the delete key on your keyboard to erase characters and make the corrections. When you have your last name spelt correctly, hit the return key.

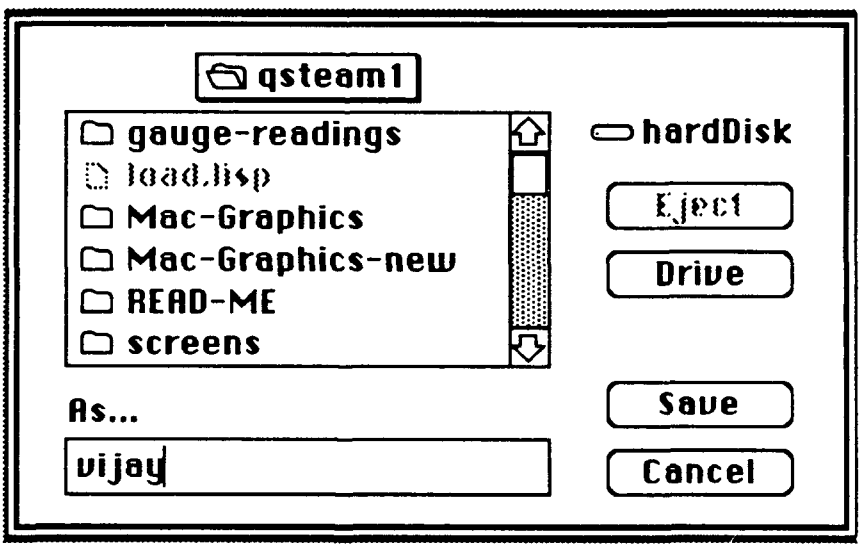


Figure 15. Output File Path Dialog

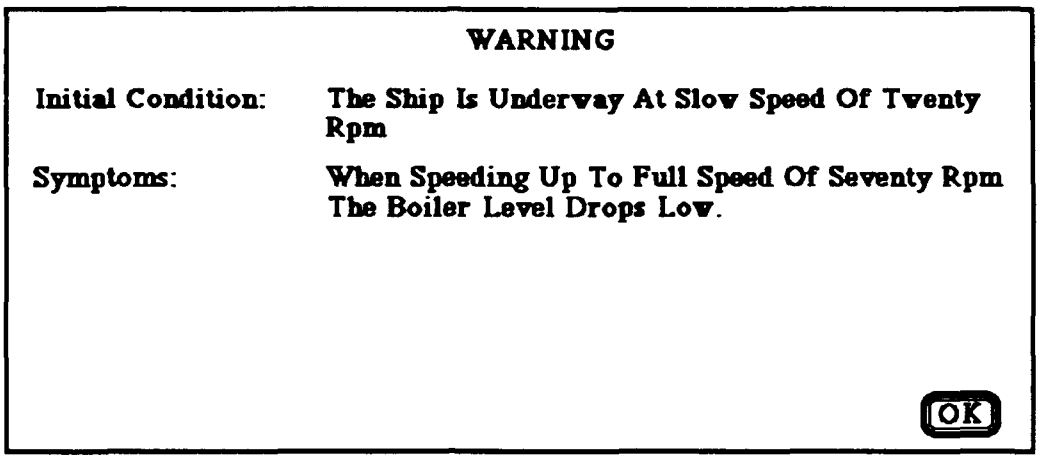


Figure 16. Symptom Display Dialog

Symptom display dialog:

After you have hit the return key, you should see the symptom display dialog appear with a beep in the center of the large screen. The symptom display dialog should look like the one shown in Figure 16. The symptom display dialog shows you the simulated ship's initial operating condition and the first symptoms that indicate the existence of a problem. For this first session, VYASA has picked a failure that has caused the feedwater level in the boiler to fall. This information is conveyed to you through the symptom display dialog currently displayed in front of you on the left screen. Your task is to identify the failed component responsible for this abnormal system behavior.

Since the time taken to solve problems is also important, each problem is simulated for 15 minutes. There is, however, no cascading of failure in this 15 minute period. Therefore, your task is confined to identifying a single component responsible for abnormal system behavior. During the next 15 minutes, you may have to make several investigations before you can accomplish your task. This guided tour of your first session will familiarize you with the actions that are necessary to achieve your goal.

You do not have to memorize the ship's initial operating conditions or the symptoms displayed in the symptom display dialog because you are provided with the facility to recall this information. However, remember that the symptom display dialog is a special dialog used by the application which deactivates your mouse for regions outside the dialog box. Only when you complete interaction with such a dialog, does the mouse get activated again for regions outside the dialog box. This instructional system has several of these special dialogs called the *modal dialogs*. These dialogs respond with a beep if you click the mouse elsewhere without first completing the interaction with them. As an example, try clicking the mouse with the cursor on one of the icons in the schematic menu while the modal symptom display dialog is still visible on the screen. The normal system response to clicking on a schematic menu icon is to display the schematic associated with the icon selected. But this response is currently suppressed by the open modal symptom display dialog. Instead, the system sounds a "beep" to remind you to first finish interacting with the open modal dialog.

You should click on "OK" button in the symptom display dialog to proceed further. Clicking on "OK" completes your interaction with the symptom display dialog and the modal dialog is closed.

Schematics:

Schematics are pictorial representations of the simulated marine power plant. Each schematic presents a view into the structure of the system. A schematic shows the sequence in which components and the gauges appear in the system. If you now click on the boiler icon in the schematic menu, the boiler schematic will be displayed. The boiler schematic is shown in Figure 17. Although the boiler schematic has been chosen as an example to

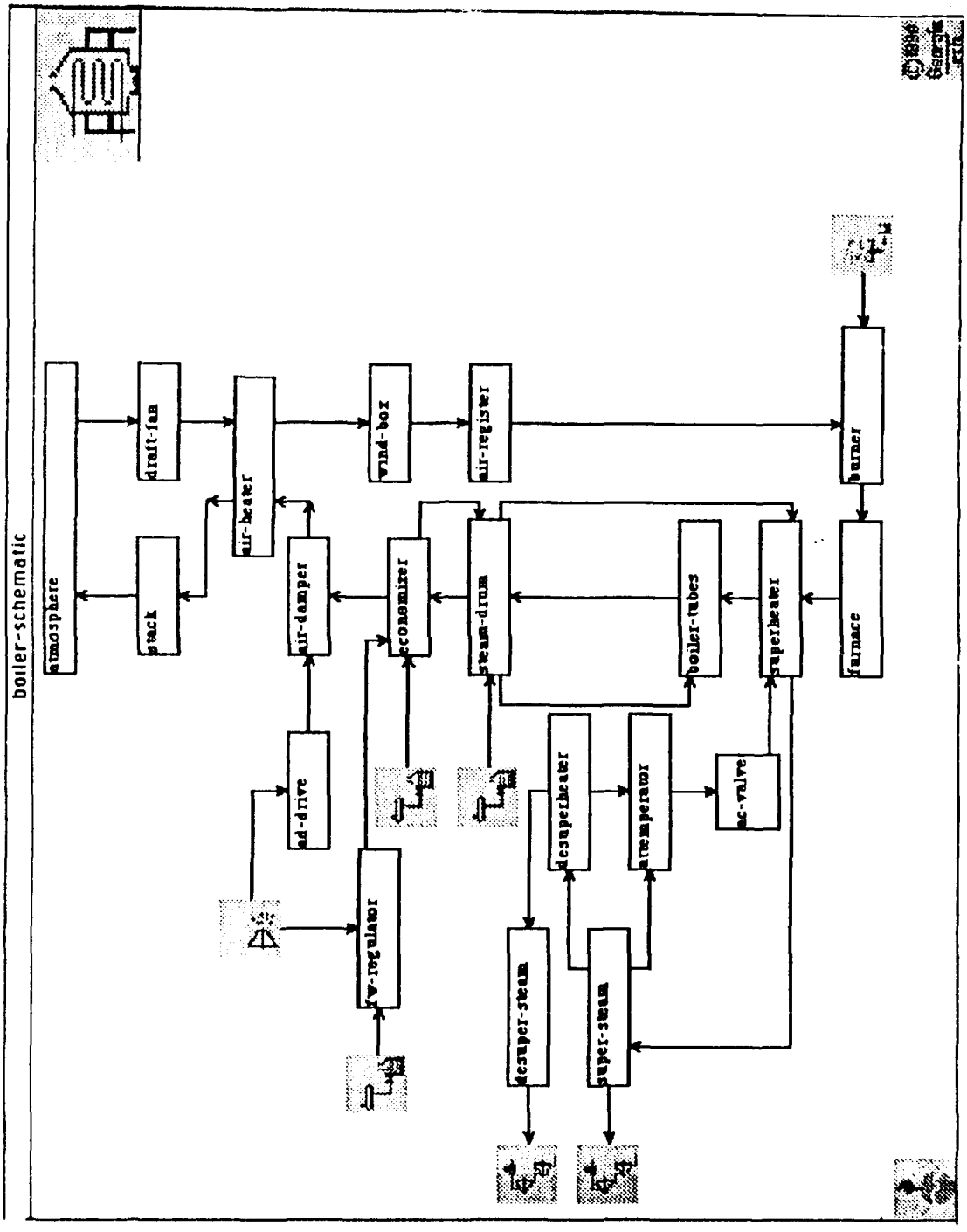


Figure 17. Boiler Schematic

explain the various features of the schematic interface, we could as well have selected any other schematic for this purpose.

All the components in the boiler schematic have been represented by rectangles. The connections between components are shown by firm lines connecting the components. These firm lines are known as connectors. The direction of flow of fluid from one component to another is shown by the arrow head on these connectors. For example, the economizer and the drum have a two way connection. The connection from the economizer to the drum represents the flow of feedwater while the connection in the reverse direction represents the flow of flue-gases.

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

Now click on the feedwater icon at the end of the input connector of feedwater-regulator and see what happens. You should notice two things. First, the display switches to feedwater schematic. Second, the boiler icon on output connector from the hp-heater is highlighted with a red band around it. The highlighted boiler icon helps you establish the physical connection between the hp-heater and the feedwater-regulator. Click on the highlighted boiler icon to get back to the boiler schematic. Notice that the boiler schematic now has two feedwater icons highlighted, one connected to the feedwater-regulator you clicked on earlier, and the other to the economizer. This simply means that the hp-heater is connected to both the feedwater-regulator and the economizer in the boiler schematic.

Most components of TURBINIA are uniquely represented in one of the seven schematics. However, there are a few that have multiple representations. For example, the condenser and the hp-heater appear in both the steam and the feedwater schematics. Switch to the steam schematic by selecting the steam icon in the schematic menu and locate the condenser and the hp-heater. Multiple representation of these components in schematics is indicated by feedwater icons adjacent to these components. Notice that these icons do not have a connector attached to them. Click on any one of these two icons and your display will switch to the feedwater schematic. The rectangular boxes marked condenser and hp-heater, in this feedwater schematic, are another representation of the same condenser and hp-heater you saw in the steam schematic.

Troubleshooting for failure indicated by the symptoms at the beginning of the session involves gathering information about system states. You collect information concerning system states using a two-action sequence. The first action of the sequence is called the investigative action. Investigative action enables you to display gauges attached to a component, if any. The second action is the informative action that allows you to access the actual gauge reading. The investigative and the informative actions are now explained with an example.

In the current session you have been asked to detect the failure responsible for decreasing water level in the boiler. It is therefore reasonable to investigate components near the boiler. Click on the boiler icon in the schematic menu to view the portion of the power plant with abnormal behavior. As a part of the process to confirm the symptom indicated, move the cursor over the drum and click on it. You have now taken an investigative action and all gauges attached or relevant to the drum are displayed as a result of this action. This action also highlights the last investigated component, the drum, in blue.

There are three types of gauges in TURBINIA: pressure, temperature, and flow-or-level gauges. These three gauges are represented by icons with letters P, T and L inscribed in them to indicate pressure, temperature, and level respectively. Although there is no visible distinction between the flow and the level gauges, it may be helpful to remember that level gauges are attached to tanks such as the deaerating-feed-tank, fuel-oil-settling-tank, atmospheric-drain-tank, distillate-tank, hotwell, and drum. Furthermore, there is just one gauge that measures flow and is located in the fuel-oil path across the strainer.

The drum you are investigating has all three types of gauges attached to it. There are two pressure gauges, one on the flue-gas connector to the economizer and the other on the steam drum. There is a temperature gauge on the feedwater connector from the economizer and a level gauge on the steam drum. The pressure gauge on the steam drum measures the saturated steam pressure in the drum itself and as such is not located over a connector.

To view the reading of a displayed gauge, you have to click on it. This action of probing a gauge is called an informative action. Select the displayed level gauge on the drum and click on it. First, the drum is lowlighted in yellow-brown and then an icon appears near the gauge. This icon is a qualitative representation of the current level. TURBINIA uses five different qualitative representations of state values. These five are normal, low, slightly-low, slightly-high and high, each represented by an icon as shown in Figure 18.

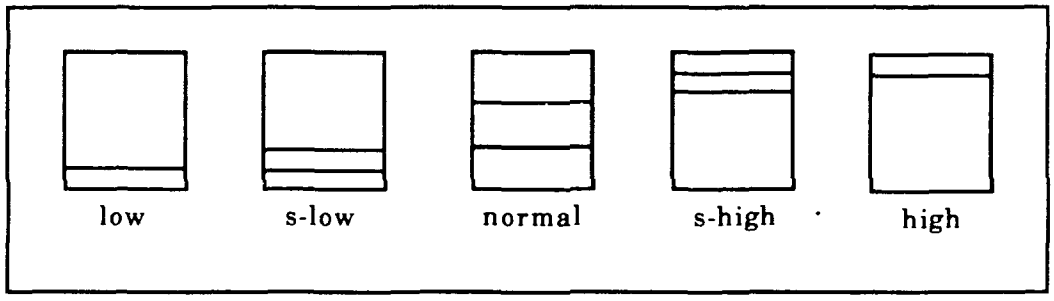


Figure 18. Qualitative State Representation

When you click on the level gauge attached to the drum you should see an icon indicating low level appear at the bottom of the gauge. However, if you see an icon that indicates a slightly-low or normal reading, do not be alarmed. Remember that TURBINIA starts simulating the failure condition at the beginning of the session and if the failure is located far away from the drum, the failure effects will take time to propagate to the drum. Therefore, the low level symptom at the drum, indicated at the beginning of the session, may not yet be visible. You will, however, be able to observe a low level reading during the course of the session.

You should never assume that a gauge reading will be the same at all times after you have observed it. While the gauge readings may change with time, the displayed gauge readings are not dynamically updated. Therefore, you must take an informative action when you need to see the current gauge reading. Thus, if you did not find the drum level low earlier, keep repeating the informative action of selecting the drum's level gauge and you will eventually find it to be low.

You can access any displayed gauge or a gauge reading only until the time you take a new investigative action. Click on the superheater to see what this means. You will discover that all gauges attached to the drum and the probed gauge readings that were visible

disappear. Instead, two new gauges, one pressure and the other temperature attached to the superheater are now displayed in the superheated steam path. Along with the appearance of the gauges, the newly investigated superheater is highlighted in blue.

There are certain components that do not have gauges attached to them. The air-damper is a good example of such a component in the boiler schematic. If you click on the air-damper, all visible gauges and gauge readings on the schematic disappear. Also, the last investigated component is lowlighted and the air-damper is highlighted. However, no new gauges are displayed because there are none attached to the air-damper.

The troubleshooting task typically involves several investigative and informative actions in one or more schematics. Assume that you have conducted several tests and now you have enough evidence to support hypothesis about the failure. Your next valid action, under these circumstances, is to submit a request for conveying the diagnosis. To make this request you click on the diagnose icon in the requests menu. Go ahead and click on the diagnose icon to see how VYASA prepares to accept your diagnosis.

When you click on the diagnose icon, VYASA asks you to select the failed component. This message is conveyed to you through a text appearing in the communications dialog at the bottom of the small screen. You then select the component that, in your opinion, is responsible for the abnormal system behavior. Selecting a failed component is an action identical to investigative action. However, this time, when you click on a component, no gauges are displayed. Instead, if your diagnosis is correct, a message congratulating you appears on the communication dialog. Otherwise, an error dialog accompanied by a beep is displayed over the schematic. This error dialog is shown in Figure 19.

As an example of erroneous diagnose, click on the economizer. The economizer is not the failed component responsible for the current abnormal system behavior. When you complete the click, an error dialog appears over the boiler schematic. This error dialog is a modal dialog. You can close this error dialog by selecting one of the two options available. If you choose "try again" you remain in the diagnose mode and can revise your diagnosis. On the other hand, if you choose "investigate", you are back in the troubleshooting mode where selecting a component displays the gauges attached to it. Click first on "investigate" in the error dialog and then on the drum and you will notice that you are out of the diagnose mode and the four gauges attached to the drum reappear on the screen.

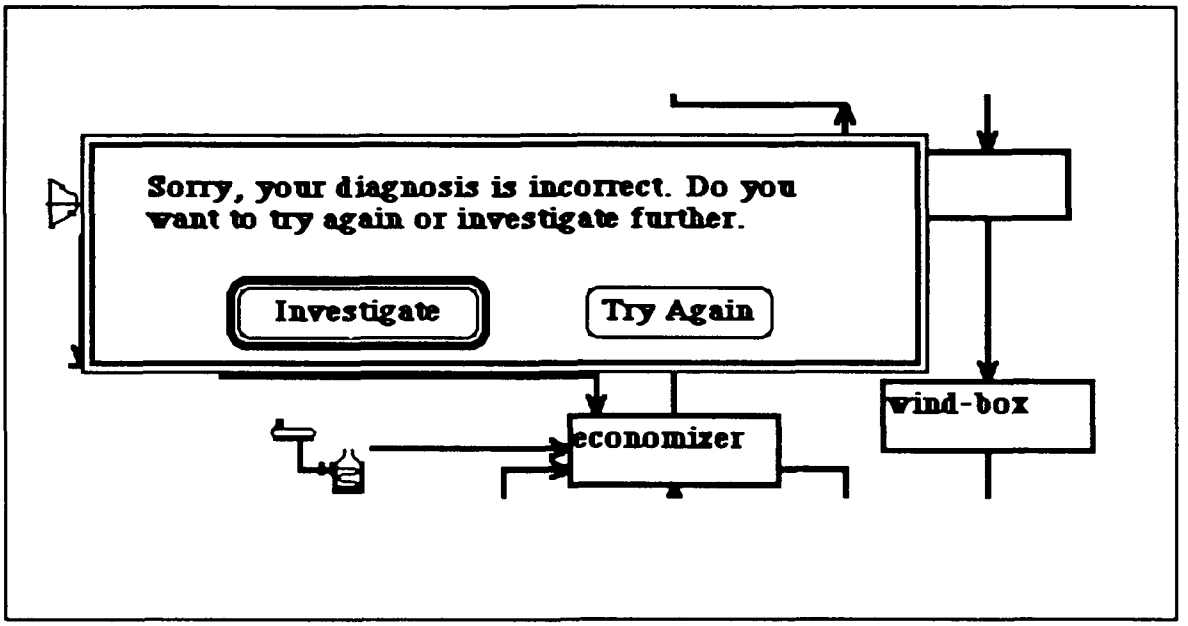


Figure 19. Incorrect Diagnosis

On all schematics you will observe the presence of two other icons that do not represent a schematic and have not also been discussed as yet. One icon appears on the right top corner and the other on the right bottom corner of all schematics. The icon on the right bottom corner is a Georgia Tech. copyright icon. This icon is disabled and has no response. The icon at right top corner is a symptom icon and is used to recall the initial symptoms. Click on this icon and you will see the symptom display dialog reappear. Thus, you can access the ship's initial operating conditions and the initial symptoms at any time. You can once again close the modal symptom display dialog by clicking on the "OK" button.

In this session you have not yet had the opportunity to interact with VYASA- your computer-based tutor. VYASA can function in two modes, passive and active. In the passive mode, you are solely responsible for initiating the communications. In the active mode, VYASA takes the initiative and provides instructions when it identifies a possible misconception based on your actions. When both the passive and active modes function simultaneously VYASA is said to operate in *dual-mode*. In all your sessions, you will find VYASA operating in dual mode. You will now explore and experience the capabilities of your computer-based tutor in both the modes.

VYASA in Passive Mode:

Click on the stop icon in the requests menu to halt the simulation and invoke VYASA in passive mode. Notice how the stop and the resume icons change their background colors. The stop icon background changes to tan indicating that it has been disabled. At the same time, the background of resume icon turns gray indicating that it has been enabled. Also notice that the cursor changes shape and turns into a "?". All these changes indicate that you have now invoked the passive tutor and temporarily halted the simulation.

After you have clicked on the stop icon you will also see a *help-levels passive tutor dialog* appear in the top left corner of the large monitor. This dialog box is also shown in Figure 20. This dialog has seven buttons of which two are enabled. The two highlighted buttons indicate the levels of help that the tutor can provide in the passive mode. Starting from the "failure" and "system" buttons, you can explore the entire knowledge-base of the tutor.

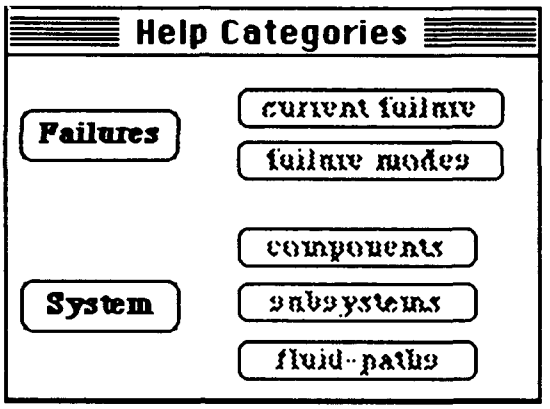


Figure 20. Help-Levels Passive Tutor Dialog

You choose the system button in the help-levels dialog to access knowledge about the system. When you click on the system button, the "components", "subsystems", and "fluid-path" buttons are enabled. These three buttons provide you with further options to select the type of system knowledge description you want to access. When you click on any one of these three buttons, a new passive tutor dialog associated with the selected button appears next to the help-levels dialog. This new dialog also contains several selectable items. You can, by selecting items in the passive tutor dialogs, explore the entire knowledge-base of the tutor at the component, subsystem and fluid-path levels.

Although interacting with passive tutor dialogs is intuitive, it is helpful to remember the following five aspects of interaction:

(1) Clicking on highlighted items (buttons, icons, and text) are valid actions and each action has an associated response.

(2) The response from the tutor usually involves one of the following:
   (a) the appearance of a new dialog box with certain items highlighted;
   (b) the highlighting of lowlighted buttons in the same dialog box to present further options;
   (c) an answer to your query as text in the dialog box or as graphics in the schematics.

   Both (a) and (b) enable you to make your query more specific.

(3) You can click on any highlighted item in any of the displayed dialog boxes to initiate communication with the tutor. Only those boxes that are relevant to the current query are kept open by the tutor.

(4) The context of the information contained in any passive tutor dialog, if unclear, can be gathered from its parent dialog appearing to its left.

(5) Finally, as long as you interact with the passive tutor your cursor will continue to be in the shape of a "?" and investigative actions in the schematics will not be possible. Although an investigative action in this mode will highlight the component, no gauges will be displayed. This response to an investigative action does not necessarily imply that there are no gauges attached to this component.

After you have explored system knowledge to your satisfaction, come back to the help-levels dialog and click on the failures button. You will notice that the tutor closes all the dialogs to the right of help-levels dialog since they are relevant to the system knowledge and not the failures. Furthermore, the two buttons in the help-levels dialog associated with failures are highlighted and the buttons associated with the system button are lowlighted. The two highlighted buttons are the "current-failure" and "failure-modes" buttons.

Using the failure-modes button you can access information concerning typical system behavior associated with each mode of failure in the liquid and gas paths. Along with the abnormal behavior, the circumstances under which the propagation of abnormal behavior may be curtailed is also provided. Go ahead and click on the failure-modes button and access the tutor's knowledge concerning the four failure modes.

After exploring the tutor's knowledge of the failure modes, click on the current-failure button. The current-failure button brings up a clipboard that extends to the smaller screen on the right. This clipboard presents a summary of observed results from your diagnostic actions. For example, based on the observed gauge readings, the clipboard displays the schematics, subsystems and fluid-paths that contain the affected gauges. The clipboard also displays the most likely mode of current failure if it can be inferred from the tests conducted. The extended portion of the clipboard on the smaller screen displays some of the gauges probed along with their gauge readings. These are mostly those gauges that the tutor considers critical for your diagnostic task.

In the current session you have investigated the steam drum in the boiler-schematic and have found the feedwater level to be low. Therefore, your current clipboard shows the boiler-schematic, the steam-generation subsystem, and the steam path as the affected schematic, subsystem, and fluid-path respectively. The location of the drum's level gauge along with the gauge reading observed are also displayed on the extended clipboard.

Sometimes you may notice that the extended clipboard has a blue colored marker next to the gauge reading. This marker indicates that the reading of that particular gauge has changed since it was last viewed. Once you re-investigate the gauge, the marker disappears and the clipboard is updated to contain the latest information. If you notice a marker appear and then disappear on its own, it simply means that the gauge reading has not stabilized or is perhaps oscillating. You will get a chance to see an example of this a little later.

By now you should be familiar with the capabilities of your computer-based tutor in the passive mode. But since your goal of identifying the failed component has not yet been accomplished, you need to continue investigating. Furthermore, since the time available to solve the problem in a session is limited, you do not want to spend all of it interacting with the tutor. After all, it is your ability to solve problems that will earn you credit and not your ability to query the tutor. Therefore, do not waste any more time and click on the resume icon to get back to the troubleshooting mode. Notice that the background colors of resume and stop icons have reverted to their original colors and the cursor is back in the shape of an arrow. However, all the passive tutor dialogs last displayed are still visible on the screens. Since the clipboard was the last passive tutor dialog displayed in your case, it is still visible even though you are back in the troubleshooting mode.

After returning to the troubleshooting mode, call up the feedwater-schematic and investigate the distillate-tank. You will once again be able to view gauges attached to the investigated component which was not possible when you were interacting with the tutor. Now if you probe the level gauge on the distillate-tank and find it to be abnormal you will notice that this information is posted on the still visible extended clipboard. Even if you

find that the reading is normal, you will notice that your diagnostic findings get recorded on the extended portion of the clipboard. This happens because VYASA considers this test as a significant troubleshooting clue. With the passage of time, you will notice that a red marker appears next to the distillate-tank's level gauge reading on the extended clipboard. This marker will disappear after some time only to reappear a little later. Such a behavior is an indication that the level gauge reading on the distillate-tank is fluctuating.

You can also obtain useful information concerning the current mode of failure from the clipboard. When you have conducted enough diagnostic tests that match the typical abnormal behavior associated with a particular mode of failure, the tutor posts a message on the clipboard. You are informed of the mode of failure that you should suspect based on the test results obtained by you thus far. For example, if for the current failure you investigate the deaerating-feed-tank and find its level to be high, you will notice that "blocked-shut" appears as the most likely mode of failure on the clipboard. This mode of failure is inferred from the two gauge readings you have observed thus far: a high feedwater level in the deaerating-feed-tank and a low feedwater level in the steam drum. Your knowledge of blocked-shut mode of failure should help you infer that the failed component lies between the deaerating-feed-tank and the steam-drum in the feedwater path.

If you are not sure of the components that lie between the tank and the drum in the feedwater path you may want to inquire about it from the tutor. Fluid path information is part of system knowledge. You can seek this information from the tutor by directly selecting the system button in the displayed help-levels dialog instead of invoking the tutor via the stop icon. However, notice that your action has the same effect on the cursor shape and the background colors of stop and resume icons as you would expect when you invoke the tutor via the stop icon. After you have found the fluid path information you were seeking you can get back into the troubleshooting mode by clicking on the resume icon.

VYASA is currently functioning in the active mode but since you have not yet given any reason to suspect a misconception it has not intervened in your troubleshooting process. Active intervention from VYASA occurs only when it infers possible misconceptions and none has been indicated thus far by your actions. To demonstrate some of the capabilities of VYASA in the active mode, we will now deliberately mislead you into taking actions that indicate some serious misconceptions.

VYASA in Active Mode:

VYASA in the active mode will often intervene to communicate with you. It does this through instructions presented on the communication dialog, accompanied by a beep. These instructions are displayed for a short period of time and therefore you must read them immediately. If your future actions indicate that you have not followed the instructions, the tutor will continue to provide you with more instructions.

To see some of the instructions VYASA is capable of providing, switch to the control-air-schematic and conduct an investigative action. VYASA will intervene and tell you that you are investigating a schematic unaffected by the failure and therefore you are not likely to obtain any useful diagnostic information from this schematic. If you continue to investigate more components in the control-air-schematic, the tutor will keep repeating the same instructions.

If you now switch to the steam-schematic and investigate the hp-regulator and the lp-regulator, the tutor no longer tells you that you are in the wrong schematic. But if you

investigate the lp-reducing-station, lp-dump-regulator and the ip-extraction-regulator which are all in the auxiliary steam use subsystem, the tutor informs you that you are investigating an unaffected subsystem. This information should help you concentrate your search for the failed component elsewhere.

In addition to guiding you with instructions as described, the tutor is capable of helping you with your hypotheses. When the tutor finds that you have conducted enough investigative and informative tests it will ask you for a list of components that you suspect are responsible for the abnormal system behavior. If you desire, you may even provide the tutor with hypotheses even before it asks for it. In both cases, you can then ask for help from the tutor on a specific hypothesis. All communications with VYASA concerning failure hypotheses is carried out via the hypothesis menu.

Hypothesis Menu

Hypothesis menu, shown in Figure 21, appears below the requests menu on the large monitor. It has four buttons. The "View" button is used to view the list of hypotheses you have provided the tutor. The "Add" button is used when you want to specify a new hypothesis. The "Delete" button is used to remove a hypothesis from a list of hypotheses. Hence the delete button gets highlighted only after you have provided your first set of hypotheses. Finally, the "Advice" button is used to seek help concerning a particular hypothesis.

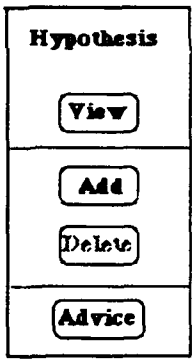


Figure 21. Hypothesis Menu

Click on the "Add" button to learn how you communicate your failure hypotheses to the tutor. After you have clicked on the add button, you are required to select the component that you suspect is responsible for the current abnormal system behavior and also pick the mode in which you suspect the component to have failed. You first select the component and then you will be prompted to select a failure mode. When adding hypothesis, either on request from the tutor or on your own, the action of selecting the suspected component is identical to the investigative action in the troubleshooting mode. You remain in the mode of adding hypothesis until you click on the "Done" button in the communication dialog. After you click on the done button you get back to the troubleshooting mode. It is, however, considered an error to click on the "Done" button without providing a single hypothesis if you have been asked for your hypotheses by the tutor.

Go ahead and provide your hypotheses concerning failure based on the diagnostic information you have thus far. Click on the done button when finished. Now use the view, advice and delete buttons in that order to become familiar with what they do. The interaction with the tutor to delete a hypothesis or to seek advice is straightforward. The tutor prompts you for every action through the communication dialog and you should not have any problems seeking help from the tutor.

By now you have been introduced to all the features of the instructional system. You may freely interact with it for a while till the session ends. If you have any questions, feel free to ask.

In this session you are not provided with the solution to the problem unless you happen to diagnose it on your own. However, in the rest of the sessions the solution will be provided to you at the end of the session. The solution will be supplemented with explanations for abnormal behavior of individual gauges. The explanation should help you form a causal model of fault propagation. Since these explanations can help improve your performance in subsequent sessions, you are recommended to pay attention to them.

This section has introduced you to all your valid interactions at TURBINIA-VYASA interface. You are now familiar with all the features of the computer-based tutor. You will be using this tutor in subsequent sessions to learn the task of troubleshooting a simulated marine power plant. Your valid interactions with the instructional system described thus far are briefly summarized in the next section. Following the summary is a description of how your performance will be measured.

Summary of Valid Actions

Provided below is a list of actions that you will perform while interacting with TURBINIA-VYASA.

Call-for-schematic-action: This is an action you perform to call a new schematic or switch between schematics. There are two ways this may be done. You can either click on an icon in the schematic menu that represents the schematic you want to view, or, if a schematic is currently displayed, click on a similar icon in the schematic itself. If you are investigating components along a suspected path in a schematic that ends up in an icon, you may prefer to use the icon in the schematic itself to switch to a new schematic. By using the icon in the schematic, a highlighted icon in the new schematic properly orients you to continue investigations along the suspected path.

Investigative-action: During your entire period of interaction with the marine power plant simulator, you are either in troubleshooting or in diagnose mode. When in troubleshooting mode, your action of clicking on the mouse button with cursor on a selected component constitutes an investigative-action. You perform investigative-action to view all gauges attached to the input and output sides of the component being investigated. A new investigative-action always makes the displayed gauges and the gauge readings of the last investigated component disappear from the screen. When no gauges are displayed in response to an investigative-action, it implies that the component investigated has no gauges attached to it.

Informative-action: The gauges displayed following an investigative-action, when probed, display the gauge reading. The action of probing displayed gauges by clicking the mouse on the gauge is called an informative-action.

Diagnose-request-action: This action is performed to switch from the troubleshooting mode to diagnose mode. You perform this action when you are prepared to indicate your diagnosis. Clicking on the mouse button after selecting the diagnose icon in the requests menu constitutes a diagnose-request-action.

Diagnostic-action: Diagnostic-action is performed following the diagnose request action. In diagnostic-action you select the component that you suspect is responsible for the observed abnormal system behavior. In indicating your diagnosis, you select the component in the same manner as you do when investigating the component. Thus, diagnostic-action is an investigative-action in diagnose mode.

Modal-dialog-action: Modal dialogs deactivate the mouse in regions outside the dialog box. Before the mouse button can be reactivated for regions outside the modal dialog box, you are required to terminate interaction with the modal dialog. Terminating interaction with a modal dialog requires selecting a button dialog item. The action of selecting the button dialog item in the displayed modal dialog is called modal-dialog-action. Symptom display dialog and error dialogs are examples of modal dialog that require modal-dialog-actions.

Help-request-action: The action of clicking on the stop icon to halt the simulation and invoke the passive tutor is called the help-request-action. Once the help-levels dialog is displayed on the screen you will probably never need to access the passive tutor through the stop icon. After seeking help from the passive tutor the first time you will be able to

communicate with it again by selecting any of the highlighted items in any of the displayed passive tutor dialogs.

Resume-request-action: Every time you initiate interaction with the tutor it is your responsibility to bring the system back to the troubleshooting mode before you can continue with diagnosis. The action of clicking on resume icon to get back to the troubleshooting mode, after you have completed interaction with the tutor, is called resume-request-action.

Tutor-dialog-action: All actions that involve clicking on highlighted items in passive and active tutor dialogs are called tutor-dialog-actions. Most tutor-dialog-actions are taken with the cursor in the shape of a "?".

Measuring Troubleshooting Performance on TURBINIA-VYASA

Although your ultimate goal is to identify the failed component responsible for abnormal system behavior, your performance is affected by other factors. This section will discuss these factors so that you have a better feel for what is expected of you.

Correct diagnosis: Successful fault diagnosis is the most important measure of your troubleshooting ability. However, since the problems are tough, your inability to solve problems has to be evaluated in conjunction with other factors.

Troubleshooting time: The total amount of time taken for troubleshooting is an important performance measure for those who successfully solve the problem. Those who solve the problems in less time have a better performance rating.

Number of relevant actions: Even though every informative action has some informational content, some have more relevance than others for the failure being investigated. Also, there is a minimum number of relevant informative actions necessary to diagnose each failure. The number of relevant informative actions past this minimum number taken to solve a problem is a measure of diagnostic performance. Smaller number of relevant informative actions required to correctly diagnose the fault implies better performance.

Number of irrelevant actions: The informative actions that have no relevance to the current problem are said to be irrelevant. The larger the number of such irrelevant informative actions during your troubleshooting exercise, the worse is the diagnostic performance.

Number of incorrect diagnosis: You are penalized any time you make an incorrect diagnosis. However, the penalty depends upon the component incorrectly identified as failed. At any stage during the troubleshooting process there are likely candidates for failed component based on the observed abnormal system states. The likelihood that a component may have failed increases or decreases as you conduct more diagnostic tests. Selecting a likely component as the cause of abnormal system behavior does not penalize you as much as picking a component that cannot have failed. Therefore, even though your performance is adversely affected by an incorrect diagnosis, it is considered worse if the suspected component cannot have failed based on the symptoms at the time you express the diagnosis.

Investigation of unaffected schematics: For each failure, there are only few schematics, subsystems and fluid paths that are affected. Affected schematics are those schematics that have gauges with abnormal readings. Investigating components in schematics that are unaffected by the failure reflects the inability on your part to relate the symptoms to the correct structural location of the power plant. Thus, investigating components in unaffected schematics reduces your performance rating. Continuing to investigate unaffected schematics in spite of the guidance provided by the tutor makes the performance worse.

Investigation of unaffected subsystems: Like the schematics, investigating components in subsystems unaffected by failure reduces your performance rating and repeating the mistake in spite of the tutor's guidance makes it worse.

Investigation of unaffected fluid paths: Once again, like the previous two factors, investigating components in unaffected fluid paths harm your performance and repeating the mistake in spite of the tutor's guidance makes it worse.

This manual has guided you through your first session, made you familiar with TURBINIA-VYASA, and has described how your performance will be measured during the experiment. From the next session, you will begin your formal training. VYASA will help you to learn to troubleshoot marine power plants efficiently.

Since it is vital for my experimental results, you are requested not to discuss any aspect of this experiment with other subjects.

GOOD LUCK!

APPENDIX B

SUBJECT CONSENT FORM

HUMAN SUBJECT CONSENT AND RELEASE FORM

Intelligent Tutoring for Diagnostic Problem Solving
in
Complex Dynamic Systems

1. I understand that the following procedure is to be observed: I will troubleshoot a failure in a simulated marine power plant using the diagnostic information I gather from the dual screen Apple Mac II workstation. In order to gather the diagnostic information concerning the system and its state I will call up various schematics, click on objects displayed in the schematics and read messages. I will interact with the simulated system using only a single button mouse. I will occasionally hear a "beep" to draw my attention to some significant events. I may have a computer-based instructor aid me in understanding the domain and/or the task in some of the sessions. When I am told that the computer-based instructor is present in the passive mode alone, I will be responsible for taking the initiative to interact with it via the available menus. I may, in some sessions, also have the computer-based instructor evaluate my misconceptions based on my actions. When my actions indicate a possible misconception, the computer-based instructor may intervene to provide me with instructions or advice.

2. I understand that the risks due to participation in this experiment do not exceed those of normal office work or those encountered in using computers as part of courses at Georgia Tech.

3. I understand that the expected benefits of the experiment is to evaluate the design of a computer-based instructional system for diagnostic problem solving in complex dynamic domains.

4. I volunteer to participate in twelve experimental sessions lasting approximately one hour each. The twelve experimental sessions may last four to five weeks. I understand that I will be paid $3 per session for the sessions completed. I understand that I can resign at any time during the course of the experiment and get paid for any time spent participating in the experiment. If I complete all twelve sessions, I will receive a total of $72, including a bonus of $36 for participation in the entire experiment, upon completing the twelfth session.

5. I understand that there are no alternative procedures that would be advantageous to me.

6. I understand that any reference to my performance will identify me by subject number rather than by my name. If the protocol requires it, I give my permission for the release of these records.

7. I understand that I may make inquiries concerning this procedure. The person to contact is Dr. T. Govindaraj in Room 333, ISyE building, phone x4-3873, or Vijay Vasandani in Room 337, ISyE building, phone x4-4322.

8. Reports of injury should be made to the principal investigator. I understand that neither the Georgia Institute of Technology nor any Georgia Tech personnel has liability and neither has made provision for payment of costs associated with any injury resulting from participation in this study.

9. I have read and understand the procedures involved and hereby consent to volunteer to participate in this study.

Subject's Signature Date

Investigator's Signature Date

SSN:_____-___-_____ P.O.B._____

APPENDIX C

SURVEY FORM

Survey Form

Name: _____ **SSN:** _____

P. O. Box: _____ **Phone:** _____

1. Your current class status (i.e. Junior, Senior etc): _____

2. Have you had some training on ship? Yes __ No __

   Ship type: Oil-fired ___ Nuclear ___ Other ___

3. Have you had some introductory course in Thermodynamics? Yes __ No __

4. If you have had more than 1 course in Thermodynamics, list how many: _____

5. Have you ever used a PC for word processing, programming or spreadsheet applications?
   Yes ___ No ___

If your answer to 5 was yes, please continue. Otherwise skip to question 8.

6. How often and for what do you use the computer?

Word-processing	_____	hours/week
Programming	_____	hours/week
Spreadsheet	_____	hours/week
Other	_____	hours/week

7. How do you interact with your applications?

Mouse or other pointing device	Yes ___	No ___
Keyboard	Yes ___	No ___
Both mouse and keyboard	Yes ___	No ___

8. Please rank your experience as a computer user:
   Novice ___ Occasional ___ Frequent ___ Hacker ___

APPENDIX D

QUESTIONNAIRES

Questionnaire 1 (and 2)

Subject ID: _____

1. Name the 4 stages of the closed loop steam cycle in a marine power plant

_____

_____

2. What is the function of each of the following in a marine power plant

boiler: _____

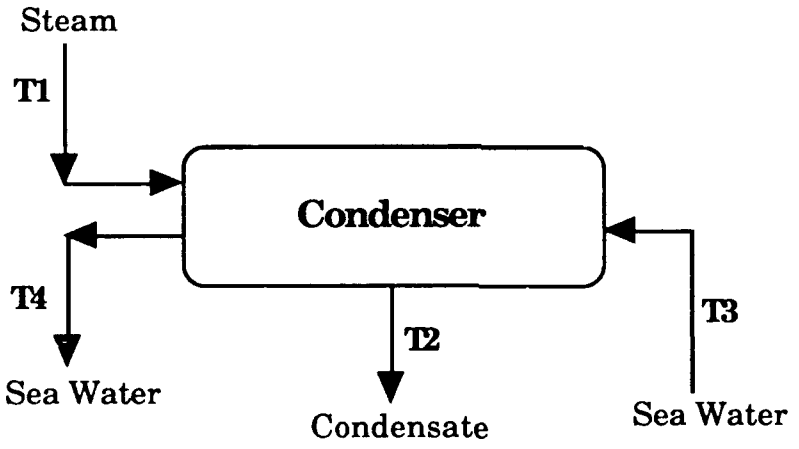
economizer: _____

condenser: _____

superheater: _____

turbine: _____

3. What is the relationship between the following sets of temperatures in the heat-exchanger shown below. Circle the appropriate equality/inequality.



T1	**<**	**>**	**=**	**T2**
T3	**<**	**>**	**=**	**T4**
T3	**<**	**>**	**=**	**T2**
T4	**<**	**>**	**=**	**T2**

where, **<** : less than; and **>** : greater than

4. Which out of the following are heat-exchangers:

boiler	Yes ___	No ____
condenser	Yes ___	No ____
turbine	Yes ___	No ____
superheater	Yes ___	No ____
economizer	Yes ___	No ____

5. Make 6 valid pairs choosing one element of each pair from column X and the other from column Y. For example, a valid pair can be expressed as (f) (6).

Column X	Column Y	Valid Pair
(a) excess combustion air	(1) black smoke	
(b) insufficient heating of fuel-oil	(2) incomplete combustion	
(c) rich fuel-air mixture	(3) white smoke	
(d) excess fuel-oil	(4) pre-ignition	
(e) overheated fuel-oil	(5) black smoke	
(f) boiling water	(6) steam	

6. Apart from power generation, name a few other uses of steam is a marine power plant.

---

7. Name a one major difference between saturated and superheated steam.

---

8. What is latent heat?

---

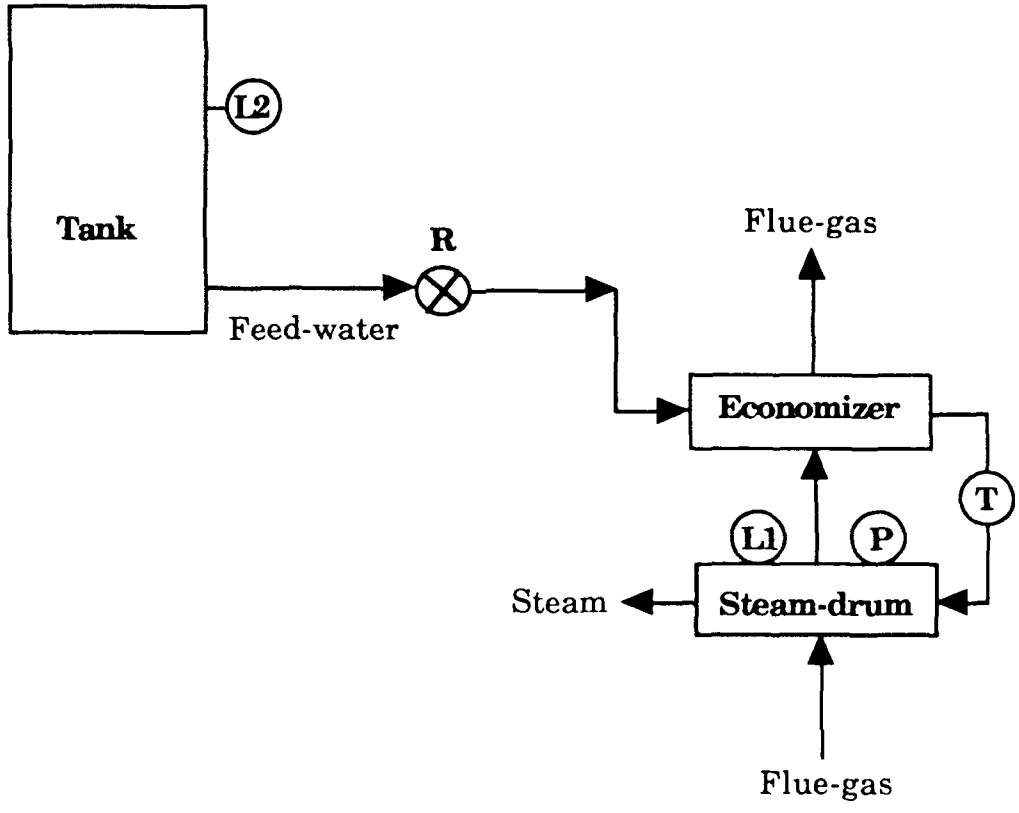
9. Name the fluids flowing through a boiler?

---

10. When the speed of the ship is increased, which of the following will increase, decrease or remain unchanged:

feed-water flow into boiler	increase	decrease	unchanged
combustion air flow to burner	increase	decrease	unchanged
steam flow out of boiler	increase	decrease	unchanged
feed-water level in steam drum	increase	decrease	unchanged
fuel-oil flow into burner	increase	decrease	unchanged

11. When the speed of the ship is increased, will the regulator "**R**" be opened or closed with respect to its current setting:



Opened ____

Closed ____

12. If the command to open/close regulator "**R**" is not followed when the ship's speed is increased, the following gauges will show abnormal/normal readings:

Feedwater level gauge **L1**	above normal	normal	below normal
Feedwater level gauge **L2**	above normal	normal	below normal
Steam pressure gauge **P**	above normal	normal	below normal
Feedwater temperature gauge **T**	above normal	normal	below normal

13. If rate of flow of steam **Qs** is greater than rate of flow of water **Qw** from the steam drum in the figure above, what happens to the pressure gauge (**P**) reading:

rises ____ falls ____ remains unchanged ____

Questionnaire 3

Subject ID _____

1. When vacuum in the condenser is low, does the lp-turbine exit temperature get affected? If so, how and why?

---

2. Which of the following can cause black smoke:

   (a) insufficient fuel

   (b) inadeqate heating of combustion air

   (c) excess air

   (d) inadeqate heating of fuel oil

   (e) insufficient air

   (f) overheated fuel

   (g) excess fuel

3. What causes white smoke?

---

4. If fuel oil temperature is low, it could be because of a malfunctioning

   (a) fuel-oil-heater

   (b) fuel-oil-heater-steam-regulator (fohs-regulator)

   (c) fuel-oil-heater-steam-trap (fohs-trap)

5. What is the function of

   (a) attemperator

   (b) atomizing steam reducing station (asr-station)

   (c) atmos-drain-tank-level-transmitter (lt)

   ---

6. In which schematics are the following components:

(a) ac-valve _____

(b) condenser _____

(c) hp-heater _____

(d) fp-turbine (feed-pump-turbine) _____

7. When saturation pressure in the steam drum decreases, what does the automatic boiler control mechanism do?

_____

_____

8. Provide 3 expected abnormal gauge readings when

(a) condensate recirculation valve is stuck open

_____

_____

_____

(b) sea strainer in the main-condenser-cold-fluid-path is blocked shut

_____

_____

_____

9. Provide 2 expected abnormal *level* gauge readings when

(a) feed-water-regulator is blocked shut

_____

_____

(b) boiler-tubes are leaking

_____

_____

10. Does a blocked shut condensate pump explain the following

(a) high level in hot-well Yes _____ No _____

(b) low vacuum in condenser Yes _____ No _____

(c) normal cpd-valve pressure Yes _____ No _____

If you were in Group I answer only questions 11 through 16. Others may skip questions 11-16.

11. What percentage of time were you sure about a failure before submitting the diagnosis and what percentage of time were you guessing?

12. What was the average number of guesses made per problem?

13. Have you become a better troubleshooter after participating this training program? Why or why not?

14. Did the training program teach you something about the system? If so what?

15. Describe the troubleshooting strategy you developed?

16. Were you satisfied with your training program? If not, what was missing?

17. How often did you use the tutor during training?

little				a lot
1	2	3	4	5

18. Did the tutor help you solve problems?

hindered		neutral		helped
1	2	3	4	5

19. What percentage of problems in the training sessions did you solve with the aid of the tutor?

less than 25% 25-50% 50-75% above 75%

20. Were some aspects of the tutor more helpful than the others? Which ones and why?

21. Did the tutor teach you the functions of the components in the power plant?

22. Were the explanations at the end of the session helpful? If yes how? If not why not?

23. Did the tutor confuse you? If so, what was the confusing aspect of the tutor?

24. Describe the troubleshooting strategy you developed?

25. What percentage of time were you sure about a failure before submitting the diagnosis and what percentage of time were you guessing?

26. What was the average number of guesses made per problem?

27. Did the training program teach you something about the system? If so what?

28. Have you become a better troubleshooter after participating this training program? Why or why not?

29. Did the tutor help you discriminate/test your hypothesis?

30. Based on your performance in the test sessions, would you say that the training made you very dependent on the tutor to solve the problems.

APPENDIX E

SAMPLE COMPUTATIONS OF TEST STATISTICS - I

Sample Computations

In this Appendix, computations of test statistics used for data analysis are shown with the help an example. These sample computations will give readers a feel for the steps involved in analyzing data using a mixed model.

Data for one performance measure is analyzed here. The performance measure chosen for this purpose is **Percentage of Guesses** (i.e., % of incorrect diagnoses that were considered guesses). The Table below shows the mean values for percentage of guesses obtained from the experimental data.

Training Condition	Seen Status			Mean
	Seen Once (20 Problems)	Seen Twice (30 Problems)	Unseen (50 Problems)	
Simulator	76.20%	66.00%	71.35%	71.40%
Passive-Tutor	43.50%	12.19%	40.00%	35.23%
Active-Tutor	17.64%	14.28%	38.77%	29.50%
Mean	52.20%	32.90%	52.16%	

Type III Expected Mean Squares Expressions computed by SAS (General Linear Model) for a mixed model with seven sources of variations, three fixed and four random, are shown in Table A.

Table A.

Source of Variation	Type III Expected Mean Squares Expression
Cond (Fixed Factor)	Var (error) + 3.086937 Var (Subj*Seen(Cond)) + 10 Var (Cond*Prob(Seen)) + 9.19524309 Var(Subj(Cond)) + Q (Cond, Cond*Seen)
Seen (Fixed Factor)	Var (error) + 3.1 Var (Subj*Seen(Cond)) + 10 Var (Cond*Prob(Seen)) + 30 Var (Prob(Seen)) + Q (Cond, Cond*Seen)
Cond*Seen (Fixed Factor)	Var (error) + 3.1 Var (Subj*Seen(Cond)) + 10 Var (Cond*Prob(Seen)) + Q (Cond*Seen)
Subj(Cond) (Random Factor)	Var (error) + 2.9032258 Var (Subj*Seen(Cond)) + 8.709677 Var (Subj(Cond))
Prob(Seen) (Random Factor)	Var (error) + 10 Var (Cond*Prob(Seen)) + 30 Var (Prob(Seen))
Cond*Prob(Seen) (Random Factor)	Var (error) + 10 Var (Cond*Prob(Seen))
Subj*Seen(Cond) (Random Factor)	Var (error) + 3.1 Var (Subj*Seen(Cond))

The ANOVA table as computed by SAS for percentage of guesses is shown below

Source of Variation	df	MS	F
Cond	2	1.672	18.09
Seen	2	2.326	25.17
Cond*Seen	4	0.070	0.76
Subj(Cond)	27	0.199	2.16
Prob(Seen)	7	0.287	3.11
Cond*Prob(Seen)	14	0.040	0.43
Subj*Seen(Cond)	54	0.093	1.01
Error	189	0.092	

The F statistics computed in the ANOVA table by SAS are all based on the error term. However, Table A shows that not all mean squares expressions computed by SAS are independent. Therefore, these statistics could not be used to determine the significance of effects considered in the experimental design. Computations of test statistics that were used to determine the significance of main and interaction effects on the percentage of guesses provided by the subjects is shown below.

Computation of test statistics:

Random effects

(1) To test if variance due to Subj*Seen(Cond) effect is significant

The statistic to test this hypothesis using the expected mean squares expression in Table A is

$$F = MSSUBJ*SEEN(COND) / MSERROR = 0.0933 / 0.09244 = 1.01$$

Therefore we can accept the hypothesis that Var (Subj*Seen(Cond)) = 0

Substituting Var (Subj*Seen(Cond)) = 0 in Expected Mean Squares Expression in Table A we get Table B for the remaining factors.

Table B.

Source of Variation	Type III Expected Mean Squares Expression
Cond (Fixed Factor)	Var (error) + 10 Var (Cond*Prob(Seen)) + 9.19524309 Var(Subj(Cond)) + Q (Cond, Cond*Seen)
Seen (Fixed Factor)	Var (error) + 10 Var (Cond*Prob(Seen)) + 30 Var (Prob(Seen)) + Q (Cond, Cond*Seen)
Cond*Seen (Fixed Factor)	Var (error) + 10 Var (Cond*Prob(Seen)) + Q (Cond*Seen)
Subj(Cond) (Random Factor)	Var (error) + 8.709677 Var (Subj(Cond))
Prob(Seen) (Random Factor)	Var (error) + 10 Var (Cond*Prob(Seen)) + 30 Var (Prob(Seen))
Cond*Prob(Seen) (Random Factor)	Var (error) + 10 Var (Cond*Prob(Seen))

(2) To test if variance due to Cond*Prob(Seen) effect is significant

The statistic to test this hypothesis using the expected mean squares expression in Table B is*

$$F = MSCOND*PROB(SEEN) / MSERROR = 0.0399 / 0.09244 = 0.43$$

Therefore we can accept the hypothesis that Var (Cond*Prob(Seen)) = 0

* Note that eliminating the Var (Subj*Seen(Cond)) term from the expected mean squares expressions when MSSUBJ*SEEN(COND)/MSERROR is 1 or less than 1 can only lead to a more conservative estimate of F statistic.

Substituting Var (Cond*Prob(Seen)) = 0 in Expected Mean Squares Expression in Table B we get Table C for the remaining factors.

Table C.

Source of Variation	Type III Expected Mean Squares Expression
Cond (Fixed Factor)	Var (error) + 9.19524309 Var(Subj(Cond)) + Q (Cond, Cond*Seen)
Seen (Fixed Factor)	Var (error) + 30 Var (Prob(Seen)) + Q (Cond, Cond*Seen)
Cond*Seen (Fixed Factor)	Var (error) + Q (Cond*Seen)
Subj(Cond) (Random Factor)	Var (error) + 8.709677 Var (Subj(Cond))
Prob(Seen) (Random Factor)	Var (error) + 30 Var (Prob(Seen))

(3) To test if variance due to Prob(Seen) effect is significant

The statistic to test this hypothesis using the expected mean squares expression in Table C is

$$F = MSPROB(SEEN) / MSERROR = 0.0287 / 0.09244 = 3.11$$

and

$$F(7,189, 0.05) = 2.0$$

Therefore we have to reject the hypothesis that Var (Prob(Seen)) = 0

$$\begin{aligned} Var(Prob(Seen)) &= MSPROB(SEEN) / 30 - MSCOND*PROB(SEEN) / 30 \\ &= 0.287 / 30 - 0.0399 / 30 \\ &= 0.0082 \end{aligned}$$

(4) To test if variance due to Subj(Cond) effect is significant

The statistic to test this hypothesis using the expected mean squares expression in Table C is

$$F = MSSUBJ(COND) / MSERROR = 0.199 / 0.09244 = 2.16$$

and

$$F(27,189, 0.05) = 1.53$$

Therefore we have to reject the hypothesis that Var (Subj(Cond)) = 0

$$\text{Var (Subj(Cond))} = MSSUBJ(COND) / 8.7096 - MSSUBJ*SEEN(COND) / 8.7096$$
$$= 0.199 / 8.7096 - 0.0933 / 8.7096$$
$$= 0.012$$

Fixed effects

(5) To test the hypothesis that means values for percentage of guesses are equal for all levels of Cond*Seen interaction effects

The statistic to test this hypothesis using the expected mean squares expression in Table C is

$$F = MSCOND*SEEN / MSERROR = 0.0701 / 0.09244 = 0.76$$

Therefore we accept the hypothesis that the means are equal and the Cond*Seen effect is not significant

(6) To test the hypothesis that mean values for percentage of guesses are equal for all levels of seen status

The statistic to test this hypothesis using the expected mean squares expression in Table C is

$$F = MSSEEN / MSPROB(SEEN) = 2.3267 / 0.287 = 8.1$$

and

$$F(2,7, 0.05) = 4.74$$

Therefore we reject the hypothesis that the means are equal. That is, mean values of percentage of guesses for problems seen once, problems seen twice and unseen problems are not the same.

By determining confidence intervals for the three means we can do a multiple comparison of means and determine which means are different. The three estimates of means are 52.2% (for problems seen once), 32.9% (for problems seen twice) and 52.16% (for unseen problems).

Although for multiple comparison an estimate of variance of means is required, the analysis done here used the error term as the estimate of variance of means. Confidence intervals for each of the three means was computed using MSERROR.

95% Confidence Interval for mean(seen once) :

$$0.522 - 1.96 * [(0.09244 / 60)] ** 0.5 < \text{mean(seen once)} < 0.522 + 1.96 * [(0.09244 / 60)] ** 0.5$$

where,

$t(0.05, 189) = 1.96$, MSERROR = 0.09244 and number of problems seen once = 60

or $0.522 - 0.0769 < \text{mean(seen once)} < 0.522 + 0.0769$

$0.445 < \text{mean(seen once)} < 0.5989$

95% Confidence Interval for mean(seen twice) :

$$0.329 - 1.96 * [(0.09244 / 90)] ** 0.5 < \text{mean(seen twice)} < 0.329 + 1.96 * [(0.09244 / 90)] ** 0.5$$

where,

t (0.05, 189) = 1.96, MSERROR = 0.09244 and number of problems seen twice = 90

or 0.329 - 0.0628< mean(seen twice) < 0.329 + 0.0628

0.2662 < mean(seen twice) < 0.3918

95% Confidence Interval for mean(unseen) :

0.5216- 1.96 * [(0.09244 / 150)] ** 0.5 < mean(unseen) < 0.5216 + 1.96 * [(0.09244 / 150)] ** 0.5

where,

t (0.05, 189) = 1.96, MSERROR = 0.09244 and number of unseen problems = 90

or 0.5216 - 0.04865 < mean(unseen) < 0.5216 + 0.04865

0.4729 < mean(unseen) < 0.57025

It is clear from the analysis that mean(seen twice) is different from mean(seen once) and mean(unseen).

(7) To test the hypothesis that mean values for percentage of guesses are equal for all three training conditions

The statistic to test this hypothesis using the expected mean squares expression in Table C is

F = MSCOND / MSSUBJ(COND) = 1.6724 / 0.199 = 8.4

and

F (2,27, 0.05) = 3.35

Therefore we reject the hypothesis that the means are equal. That is, mean values of percentage of guesses from subjects in the three training conditions are not the same.

By determining confidence intervals for the three means we can do a multiple comparison of means and determine which means are different. The three estimates of means are 71.4% (simulator), 35.23% (passive tutor) and 29.5% (active tutor).

Although for multiple comparison an estimate of variance of means is required, the analysis done here once again uses the error term as the estimate of variance of means. Confidence intervals for each of the three means was computed using MSERROR.

95% Confidence Interval for mean(simulator) :

$0.714 - 1.96 * [(0.09244 / 100)] ** 0.5 <$ mean(simulator) $< 0.714 + 1.96 * [(0.09244 / 100)] ** 0.5$

where,

t (0.05, 189) = 1.96, MSERROR = 0.09244 and number of problems = 100

or $0.714 - 0.059 <$ mean(simulator) $< 0.714 + 0.059$

$0.655 <$ mean(simulator) < 0.773

95% Confidence Interval for mean(passive tutor) :

$0.3523 - 1.96 * [(0.09244 / 100)] ** 0.5 <$ mean(passive tutor) $< 0.3523 + 1.96 * [(0.09244 / 100)] ** 0.5$

where,

t (0.05, 189) = 1.96, MSERROR = 0.09244 and number of problems = 100

or $0.3523 - 0.059 <$ mean(passive tutor) $< 0.3523 + 0.059$

$0.2933 <$ mean(passive tutor) < 0.4113

95% Confidence Interval for mean(active tutor) :

$0.295 - 1.96 * [(0.09244 / 100)] ** 0.5 <$ mean(active tutor) $< 0.295 + 1.96 * [(0.09244 / 100)] ** 0.5$

where,

t (0.05, 189) = 1.96, MSERROR = 0.09244 and number of problems = 100

or $0.295 - 0.059 <$ mean(active tutor) $< 0.295 + 0.059$

$0.236 <$ mean(active tutor) < 0.354

It is clear from the analysis that mean(simulator) is different from mean(passive tutor) and mean(active tutor).

APPENDIX F

SAMPLE COMPUTATIONS OF TEST STATISTICS - II

Sample Computations

In this Appendix, sample computations for pairwise comparison of proportion of premature diagnosis made by subjects in each of the three training conditions are presented. Details of the method used here to compare proportions are described in Hines and Montgomery (1990). The Table below shows the mean values for premature diagnosis obtained from the experimental data.

Premature Diagnosis (Proportion of of Solved Problems)

Training Condition	Seen Status			Mean
	Seen Once (20 Problems)	Seen Twice (30 Problems)	Unseen (50 Problems)	
Simulator	2/19	2/28	21/46	25/93
Passive-Tutor	0/19	0/30	14/46	14/95
Active-Tutor	0/19	0/28	8/41	8/88
Mean	2/57	2/86	43/133	

Each correct diagnosis provided by the subject can be grouped into one of the three categories: premature, timely or overdue.

Let X1 be the number of premature diagnoses provided by subjects in the simulator group, N1 be the number of problems solved by this group and P1 be the proportion of correct diagnoses that are premature (i.e., X1 / N1).

Let X2 be the number of premature diagnoses provided by subjects in the passive tutor group, N2 be the number of problems solved by this group and P2 be the proportion of correct diagnoses that are premature (i.e., X2 / N2).

Let X3 be the number of premature diagnoses provided by subjects in the active tutor group, N3 be the number of problems solved by this group and P3 be the proportion of correct diagnoses that are premature (i.e., X3 / N3).

We are interested in making pairwise comparison of proportions P1, P2 and P3 to see if they come from the same distribution. To test the hypothesis that P1 = P2, we can use the Z statistic given by

$$Z = P1 - P2 / [P (1 - P) (1 /N1 + 1/N2)]^{**}0.5$$

where,

$$P = (X1 + X2) / (N1 + N2)$$

Substituting the values from the data table we get

$$P = (25 + 14) / (93 + 95) = 0.20$$

and

$$Z = (25 / 93 - 14 / 95) / [0.20 (1 - 0.20) * (1 / 93 + 1 / 95)] ** 0.5$$
$$= 2.07$$

Similarly, the Z statistic for comparing P1 and P3 can be also be computed as follows

$$Z = P1 - P3 / [P (1 - P) (1 /N1 + 1/N3)]^{**}0.5$$

where,

$$P = (X1 + X3) / (N1 + N3)$$

Substituting the values from the data table we get

$$P = (25 + 8) / (93 + 88) = 0.18$$

and

$$Z = (25 / 93 - 8/ 88) / [0.18 (1 - 0.18) * (1 / 93 + 1 / 88)] ** 0.5$$
$$= 3.11$$

And, the Z statistic for comparing P2 and P3 can be also be computed as follows

$$Z = P2 - P3 / [P (1 - P) (1 /N2 + 1/N3)]^{**}0.5$$

where,

$$P = (X2 + X3) / (N2 + N3)$$

Substituting the values from the data table we get

$$P = (14 + 8) / (95 + 88) = 0.12$$

and

$$Z = (14 / 95 - 8/ 88) / [0.12 (1 - 0.12) * (1 / 95 + 1 / 88)] ** 0.5$$
$$= 1.19$$

Now, Z(0.95) = 1.65. Therefore, we can say that at α value of 0.05, P1 is different from P2 and P3 but P2 is not significantly different from P3.

The analysis of other two categories of diagnosis: timely and overdue can also be done in the same manner. Also, the analysis to compare performance in terms of premature, timely and overdue diagnosis for the three levels of seen status can be conducted as described above.

APPENDIX G

SOME COMMENTS FROM SUBJECTS

Some of the comments collected from subjects who participated in the experiment are presented in this Appendix. These comments were gathered during the ten training sessions. Subjects trained on the simulator (Group I) were assigned a code number beginning with 1. Subjects trained with the passive tutor (Group II) and those trained with the active tutor (Group III) were assigned codes beginning with 2 and 3 respectively.

Group I

Subject 101: Doesn't make sense how pressure could be low. Wish it explained why.

Subject 102: Couldn't find the (faulty) component simply because I did not know what the asr station was; I found it almost by chance. I often end up picking the bad component on a guess.

Subject 103: There are no references to pieces of equipment that I don't understand. I wish there was.

Subject 104: I need to know more about some of these components and what the purpose of these components. Sometimes I have a hard time following these fluid paths. All arrows are alike and it gets confusing.

Subject 105: I am unfamiliar with small obscure parts. Can somebody not tell me what they do. I need to pay more attention to what I guess because the solution was a component that I could have sworn I had picked earlier.

Subject 106: Still having problems figuring out the purpose of the component.

Subject 107: I wish I could tell what type of fluid the arrows represented and how they actually flow through the component. Piping systems on Navy vessels are color coded. Often wish for input on the interrelationship between various subsystems. Where is the recirculation regulator. Could not find it. Wish it was possible to view all the affected gauges simultaneously. Would be nice if the solution listed some explanations.

Subject 109: Could be little more descriptive, could give us all possibilities to a certain problem. A glossary explaining briefly the functions of the components would be helpful. I wish there was a way to show all abnormal readings.

Subject 110: It is difficult not knowing what some of the components do. It is also frustrating not to see the schematics when the solution is told. However, the program is very easy to use and I have no problems with the operation of.

Group II

Subject 201: I learned the control mechanisms responsible for much of the secondary effects. I need to keep the tutor in the mode where I can see the affected gauges. It helps me.

Subject 202: Tutor was helpful in showing me what the functions of the components were so I improved my chances of solving the problems.

Subject 203: This set-up is very easy to work with after a couple of days of practice. It helps me understand how one component can affect so many others. It is good that the passive tutor does not lead me to the solution but provides me with a chance to learn about the system on my own. This tutor has familiarized me to the propulsion system.

Subject 205: Current failures is very useful. It eliminates the need to write down the gauge readings. It provides information that is most useful in solving problems.

Subject 206: The tutor became increasingly helpful after I learned the most effective way of using the information it gives. Only complaint is that the highlighted subsystems and fluid paths should remain highlighted after I start investigations again.

Subject 207: The mouse is the best part of the system. Also, the passive tutor is very helpful in the failures mode. It gives me the opportunity to see when the gauges change as the problem propagates through the system.

Subject 208: I use the tutor a lot. It helps me understand the functions of the components. The current failures help me a lot to keep track of affected gauges.

Subject 209: Tutor is very convenient to use. Current failures mode is very useful.

Group III

Subject 301: I like passive tutor. I use it a lot. Like fluid paths, functions and current failures features. I like to check my hypothesis using active tutor.

Subject 303: The explanations at the end of each problem shows step by step what went wrong and provides helpful information that can be used in later problems. When the clipboard displays the message that something is blocked shut it would be nice if it beeped.

Subject 304: Tutor's explanation of the problems and hypothesis testing were very helpful.

Subject 306: Tutor is a great help. Helped me solve many problems. It assisted me in finding out what was not malfunctioning. It also explained the problems. Towards the end of the training sessions I tried to stay away from the tutor.

Subject 307: The tutor helps me evaluate all possible component failures. But, I am afraid I may have become too dependent on the tutor.

Subject 308: The tutor tends to increase your ability to discriminate among choices. It is very helpful in letting you know if your hypothesis is implausible.

Subject 309: Hypothesis testing is a great feature. It always helps me find the fault.

Subject 310: Good index of components and their functions. Ability to see affected gauges in failures mode is useful. Fluid paths in tutor help locate problems. Ability to ask for advice with hypothesis is very useful.

APPENDIX H

SOME RESPONSES TO QUESTIONNAIRE 3

Q: Were some aspects of the tutor more helpful than others? Which ones and why?

Group II

Subject 201: Yes- current failure. Shows which gauges are affected and when a component is blocked shut or stuck open. Also the tutor helped in outlining fluid paths.

Subject 202: I liked the list of affected gauges which helped narrow down my choices of what was going wrong.

Subject 203: The tutor made it very clear when a part was stuck open or closed which made the problems easier. There were some inconveniences however such as not being able to investigate in the tutor mode.

Subject 204: For the most part the different aspects were used equally and were equally helpful.

Subject 205: The ability to show the affected gauges and tell when their reading had changed. The component inquire section where you could ask the function of the parts. This was very helpful because a great many of the parts were new to me. It helped me to better understand the component function and what role it might play in the problem.

Subject 206: The current failures helped me to make a path to investigate and it told me the affected subsystems and fluid paths.

Subject 207: I found the tutor useful in many ways.

Subject 208: The current failure helped me keep track of which gauges I have already looked at and if they changed or not. I also found the system components and fluid paths features to help me locate problems and where to look.

Subject 209: Current failures mode was useful.

Subject 210: failure mode and component function were the most useful.

Group III

Subject 301: Current failures - you could see gauges affected. Fluid paths and functions-helped me see where the affected path lead and what each component was called and what it did. Active tutor stopped me from checking wrong schematics, fluid paths and components therefore saving a lot of time.

Subject 302: The gauges affected, fluid paths and advice. You could see what was affected and not worry about other things.

Subject 303: The clipboard allowed me to see which gauges were affected because of the problem. Also, it would tell me which schematics had nothing to do with the symptoms.

Subject 304: Yes, the gauge readings display helpful for study of problem. Explanations were also helpful. Component functions and fluid paths useful in solving problems.

Subject 305: Affected gauges displayed were a great help. Hypothesis aiding was also very useful.

Subject 306: Yes, the current failure part allowed me to see what the nature of the problem was (i.e., blocked shut, stuck open). The hypothesis advice part helped me see if I was correct in my diagnosis.

Subject 307: The advice portion kept me going in the right direction. Any function of the component could be looked up.

Subject 308: The active advice option was helpful because it lets you know if your hypotheses were valid. The current failure option was helpful in letting you know the extent and characteristics of different faults.

Subject 309: Yes, current failures, hypothesis advice and component functions.

Subject 310: Helped in not making poor guesses and telling you about wrong schematics and fluid paths.

BIBLIOGRAPHY

Anderson, J. R., Boyle, C. F., and Reiser, B. (1985). Intelligent tutoring systems. *Science,* 228(4698), pp. 456-462.

Anderson, J. R. (1988). The expert module. In M. C. Polsen and J. J. Richardson (Eds.), *Foundations of ITS*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Babcox and Wilcox Company. (1978). *Steam: Its generation and use.* (39th Ed.).

Brown, J. S., Burton, R. R., and de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in Sophie I, II and III. In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London.

Brown, J. S., and Burton, R. R. (1986). Reactive learning environments for teaching electronic troubleshooting. In W. B. Rouse (Ed.), *Advances in Man-Machine Systems Research* Vol. III. JAI Press Inc., Greenwich, CT.

Bureau of Naval Personnel (1957). *Engineering Operation and Maintenance.* Prepared by the Bureau of Naval Personnel

Burns, H., Parlett, J. W., and Redfield, C. L. (Eds.) (1991). *Intelligent tutoring systems: Evolution in design.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Burton, R. R., Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman and J. S. Brown (Eds.), *Intelligent Tutoring Systems*, Academic Press, London.

Burton, R. R. (1988). The environment module of intelligent tutoring systems. In Polson, M. C. and Richardson, J. J. (Eds.), *Foundations of intelligent tutoring systems.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Charniak, E., and McDermott, D. (1985). *Introduction to artificial intelligence.* Addison-Wesley, Reading, MA.

Chu, R. W., Jones, P. M. and Mitchell, C. M. (1991). The Georgia Tech Payload Operations Control Center simulation: Design and implementation in C++. *Proceedings of the Society for Computer Simulation Multiconference on Object-Oriented Simulation 1991*, Simulation Series Vol. 23, Number 3, 167-174.

Chu, R. W. (1991). Towards the tutor/aid paradigm: Design of intelligent tutoring systems for operators of supervisory control systems. Doctoral dissertation, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Clancey, W. J. (1987). *Knowledge-based tutoring: The GUIDON program,* MIT Press, Cambridge, MA.

Cohen, P. R., and Fiegenbaum, E. A., (Eds.) (1982). *The Handbook of Artificial Intelligence,* Vol. III, William Kaufmann, Inc., Los Altos, CA.

Department of Navy Sea Systems Command. *Engineering Operational Sequencing System* (EOSS) and *Engineering Operational Procedures* (EOP).

Fath, J. L. (1987). An architecture for adaptive computer-assisted instruction programs. Ph.D. thesis, Technical Report CHMSR 87-3, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

Fath, J. L., Mitchell, C. M., and Govindaraj, T. (1990). An ICAI architecture for troubleshooting in complex, dynamic systems. *IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-20 no. 3,* pp. 537-558.

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

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

Govindaraj, T. (1987). Qualitative approximation methodology for modeling and simulation of large dynamic systems: Applications to a marine powerplant. *IEEE Transactions on Systems, Man, and Cybernetics, vol.SMC-17 no. 6,* pp.937-955.

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

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

Gritzen, E. F. (Ed.) (1980). *Introduction to Naval Engineering.* Naval Institute Press, Annapolis, MD.

Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence, vol. 26 (3),* pp. 251-321.

Henneman, R. L., and Rouse, W. B. (1984). Measures of human performance in fault diagnosis tasks. *IEEE Transactions on Systems, Man, and Cybernetics, vol.SMC-14 no. 1,* pp.99-112.

Hines, W. W., and Montgomery, D. C. (1990). *Probability and statistics in engineering and management science.* (Third Ed.). John Wiley and Sons, Inc. Publishers, NY.

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

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

Jones, P. M. (1991). Human-Computer cooperative problem solving in supervisory control. Doctoral dissertation, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

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

Lajoie, S. P., and Lesgold, A. (1990). Apprenticeship training in the workplace: Computer coached practice environment as a new form of apprenticeship. In Machine-Mediated Learning.

Lesgold, A. (1990a). Tying development of intelligent tutors to research on theories of learning. In H. Mandl, E. De Corte, S. N. Bennett, and H. F. Friedrich (Eds.), *Learning and Instruction: European research in an international context.* Vol. 3. Pergamon, Oxford.

Lesgold, A. (1990b). Intelligent computer aids for practice of complex troubleshooting. FAA symposium on Training Technology.

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

Macmillan, S. A., Emme, D., and Berkowitz, M. (1988). Instructional Planners: Lessons Learned. In J. Psotka, L.D. Massey and S.A. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned,* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Marine Safety International. (1983) Personal communications.

Massey, L. D., de Bruin, J., and Roberts, B. (1988). A training system for system maintenance. In J. Psotka, L.D. Massey and S.A. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned,* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Milliken, G. A., & Johnson, D. E. (1984). *Analysis of Messy Data Volume I: Designed Experiments*. Van Nostrand Reinhold Company, New York.

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

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

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

Mitchell, C. M., and Miller, R. A. (1986). A discrete control model of operator function: A methodology for information display design. *IEEE Transactions on System, Man, and Cybernetics, vol. SMC-16(3)*, pp. 343-357.

Munro, A., Fehling, M. R., and Towne, D. M. (1985). Instruction intrusiveness in dynamic simulation training. *Journal of Computer-Based Instruction, vol. 12 (2)*, pp. 50-53.

Naval Training Command (1973). *Engineering Administration*. United States Government Printing Office, Washington D.C.

Nii, H. P. (1986). Blackboard systems. *AI Magazine, vol. 7-2* and *7-3*.

Nii, H. P., Feigenbaum, E. A., Anton, J. J., and Rockmore, A. J. (1982). Signal-to-symbol transformation: HASP/SIAP case study. Report No. HPP-82-6, Heuristic Programming Project, Heuristic Programming Project, Stanford University, Stanford, CA.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA.

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

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

Rasmussen, J. (1986). *Information processing and human machine interaction: An approach to cognitive engineering*, North-Holland, N.Y.

Rich, E. (1983). *Artificial Intelligence*. McGraw-Hill, New York.

Rickel, J.W., Intelligent Computer-Aided Instruction: A survey Organized Around System Components. *IEEE Transactions on Systems, Man, and Cybernetics, vol 19, No. 1,* pp. 40-57, 1989.

Rouse, W. B. (1982). Models of human problem solving: Detection, diagnosis and compensation for system failures. *Automatica, vol. 19,* pp. 613-625.

Rubin, K. S., Jones, P. M. and Mitchell, C. M. (1988). OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture. *IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-18,* pp. 618-637.

Sheridan, T. B., and Johannsen, G. (Eds.) (1976). *Monitoring behavior and supervisory control,* Plenum, New York.

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

Su, Y.-L. (1985). Modeling fault diagnosis performance on a marine power plant simulator. Doctoral dissertation, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Georgia Institute of Technology

Su, Y.-L., and Govindaraj, T. (1986). Fault diagnosis in a large dynamic system: Experiments on a training simulator. *IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-16(1),* pp. 129-141.

Takahashi, Y., Rabins, and M. J., Auslander, D. M., (1972). *Control and dynamic systems.* Addison-Wesley Publishing Company, Rading, MA.

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

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

Towne, D. M. (1986). The generalized maintenance trainer: Evolution and revolution. In W. B. Rouse (Ed.), *Advances in Man-Machine Systems Research* Vol. III. Greenwich, CT: JAI Press Inc.

Vasandani, V., and Govindaraj, T. (1988). A model of the operator's task in diagnostic problem solving. In *Empirical Foundations of Information and Software Science V,* pp. 237-248.

Vasandani, V., Govindaraj, T., and Mitchell, C.M. (1989). An Intelligent Tutor for Diagnostic Problem Solving in Complex Dynamic Systems. Proceedings of *The 1989 IEEE International Conference on Systems, Man, and Cybernetics, vol. II*, pp. 772-777.

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

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

Vasandani, V., and Govindaraj, T. (1991).Intelligent Diagnostic Problem Solving Tutor: An Experimental Evaluation. Accepted for presentation at The 1991 IEEE International Conference on Systems, Man, and Cybernetics to be held in Charlottesville, VA. Oct, 1991.

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

Wickens, C. D. (1984). *Engineering psychology and human performance*, Charles Merrill, Columbus, Ohio.

Winston, P. H. (1980). *Artificial Intelligence* (Second Ed.). Addison-Wesley, Reading, MA.

Woods, D. D. (1986). Cognitive technologies: The design of joint human-machine cognitive systems. *AI Magazine*, pp. 86-92.

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

Woolf, B. P., and McDonald, D. D. (1984). Building a computer tutor: Design issues. *IEEE Computer, 17(9)*, pp. 61-73.

Woolf, B. P. (1986). Teaching a complex industrial process. *Coins Technical Report 86-24*, Computer and Information Science, University of Massachusetts, Amherst, MA.

VITA

Vijay Vasandani was born on December 4, 1960 in Sindri, India. He received the Bachelor of Science degree in Mechanical Engineering from Punjab Engineering College, Chandigarh, in 1983. He received the Master of Science degree in Mechanical Engineering from North Carolina A & T State University, Greensboro, NC, in 1986. He enrolled in the graduate studies program at Georgia Institute of Technology in 1986 from where he received the Master of Science degree in Industrial and Systems Engineering in 1989.