(2)

February 1987 Report No. 108

AD-A177 359

Representing System Behaviors and Expert Behaviors for Intelligent Tutoring

Douglas M. Towne
Allen Munro
Quentin A. Pizzini
David S. Surmon

BEHAVIORAL TECHNOLOGY LABORATORIES
Department of Psychology
University of Southern California

Sponsored by

Navy Personnel Research and Development Center

and

Office of Naval Research
Cognitive Science Research Programs

Under Contract No N00014-85-C-0040
ONR NR 535-001

USC

February 1987 Report No. 108

Representing System Behaviors and Expert Behaviors for Intelligent Tutoring

Douglas M. Towne
Allen Munro
Quentin A. Pizzini
David S. Surmon

BEHAVIORAL TECHNOLOGY LABORATORIES
Department of Psychology
University of Southern California

Sponsored by

Navy Personnel Research and Development Center

and

Office of Naval Research
Cognitive Science Research Programs

Under Contract No. N00014-85-C-0040
ONR NR 535-001

ADA177359

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION: Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT: "Approved for Public Release: Distribution Unlimited"
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S): Technical Report No. 108	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Behavioral Technology Laboratories; 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION: Personnel and Training Research Programs Office of Naval Research (Code 1142PT)
6c ADDRESS (City, State, and ZIP Code): 1845 South Elena, 4th Flr. Redondo Beach, CA 90277	7b ADDRESS (City, State, and ZIP Code): 800 N. Quincy St. Arlington, VA 22217-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: N00014-85-C-0040
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO 61153N; PROJECT NO RR04206; TASK NO RR04206-01; WORK UNIT ACCESSION NO NR-535-003

11 TITLE (Include Security Classification): Representing System Behaviors and Expert Behaviors for Intelligent Tutoring.

12 PERSONAL AUTHOR(S): Douglas M. Towne, Allen Munro, Quentin A. Pizzini, David S. Surmon

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Interim	FROM 11/84 TO 2/87	87/02/09	52

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Artificial Intelligence, Graphical Simulation, Troubleshooting Expertise, Simulation Training, Representing Device Behavior
05	09	08	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Simulation-based software tools that can infer system behaviors from a deep model of the system have the potential for automatically building the semantic representations required to support intelligent tutoring in fault diagnosis. The Intelligent Maintenance Training System (IMTS) is such a resource, designed for use in training troubleshooting skills and in conducting research into intelligent instruction. The IMTS incorporates a generalized model of an expert diagnostician, termed Profile, to evaluate student performance and to recommend improved fault isolation strategies.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT: [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL: Dr. Susan Chipman	22b TELEPHONE (Include Area Code): (202) 696-4318; 22c OFFICE SYMBOL: ONR 1142PT

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

Block . 19 - Abstract - Continued

The equipment expert uses domain-independent editing tools to construct the simulation using previously defined generic objects. As the diagrams are interactively assembled, an underlying representation of system content and structure is automatically produced, allowing the graphical simulation to change in response to student actions during training sessions.

The first application of the IMTS will be as a trainer of fault isolation skills for the Bladefolding system of the SH-3H helicopter. In this application, the IMTS is coupled to the Generalized Maintenance Training System (GMTS), a videodisc-based simulator that displays high-resolution color views of the controls and indicators as they are manipulated by the student.

ABSTRACT

Simulation-based software tools that can infer system behaviors from a deep model of the system have the potential for automatically building the semantic representations required to support intelligent tutoring in fault diagnosis. The Intelligent Maintenance Training System (IMTS) is such a resource, designed for use in training troubleshooting skills and in conducting research into intelligent instruction. The IMTS incorporates a generalized model of an expert diagnostician, termed Profile, to evaluate student performance and to recommend improved fault isolation strategies.

The equipment expert uses domain-independent editing tools to construct the simulation using previously defined generic objects. As the diagrams are interactively assembled, an underlying representation of system content and structure is automatically produced, allowing the graphical simulation to change in response to student actions during training sessions.

The first application of the IMTS will be as a trainer of fault isolation skills for the Bladefolding system of the SH-3H helicopter. In this application, the IMTS is coupled to the Generalized Maintenance Training System (GMTS), a videodisc-based simulator that displays high-resolution color views of the controls and indicators as they are manipulated by the student.

ACKNOWLEDGEMENTS

Development of the Intelligent Maintenance Training System (IMTS) is being supported by the Cognitive Science Research Programs of the Office of Naval Research and by the Navy Personnel Research and Development Center. We thank Susan Chipman and Michael Shafto of ONR and Vern Malec of NPRDC for their support of this work.

The Profile model of expert diagnostic behavior was originally developed under funding from the Engineering Psychology Group of the Office of Naval Research. We thank Martin Tolcott and Gerald Malecki for their support of this work.

William Johnson of Search Technology and Ronald Renfro of ManTech Mathetics Corporation have provided subject-matter expertise for the first application of the IMTS. William Johnson has also participated in the preliminary evaluations of the simulation authoring facilities and has contributed to the student interface design.

We thank Richard Burton of Xerox PARC for his interest and assistance in exploring techniques for efficient simulation computation.

TABLE OF CONTENTS

LIST OF FIGURES

SECTION I. INTRODUCTION

Background

This report describes the Intelligent Maintenance Training System (IMTS), under development at Behavioral Technology Laboratories, University of Southern California, since early 1985. Our first report on this work (Towne, Munro, Pizzini, and Surmon, 1985) set out the underlying instructional principles fundamental to the design of an intelligent maintenance training system, the training characteristics sought for IMTS, the desired environment for constructing domain-specific simulation and training scenarios, and generalizable techniques for assessing and supporting human diagnostic performance. For completeness, this report will reiterate some of the issues which influenced major aspects of the system design. However, the primary objective of the report is to provide a relatively comprehensive account of the processes used in IMTS for representing system behavior and for generating expert diagnostic behavior. A companion report (Towne, Munro, Pizzini, and Surmon, in preparation) will present the instructional functions of IMTS.

The nature of fault diagnosis allows some relatively specialized consideration of the ways computer-generated intelligence can contribute to training effectiveness. Much of the IMTS design attempts to take advantage of the character of fault diagnosis. Readers should understand that approaches followed in the IMTS may not be entirely applicable in domains outside of simulation-based maintenance training.

Objectives of the Work

One premise of the development project is that there have been numerous research projects in the area of intelligent training and in maintenance training which have yielded useful results, principles, and techniques in relatively restricted, isolated, and sometimes abstract environments, and that the time has arrived to begin trying to interpret and apply those experiences and findings in a functioning system.

A major objective of the IMTS project is to attempt to construct a cohesive maintenance training system largely of these concepts and techniques. This process has identified the areas of instruction which are well supported by research and those which are not. In areas where there has been a substantial amount of research, applying the findings in a direct manner can be a very difficult matter, either because it is difficult to interpret the work in an operational way, or because findings and principles from different sources seem to be in partial or complete conflict. We have found, however, that most of the apparent conflict can be resolved by carefully considering the setting in which the research was conducted and some of the unstated assumptions or objectives.

The second major objective is to produce an operational maintenance training system which can be used by instructors to meet a wide range of pressing training needs. To meet this objective the IMTS must 1) be sufficiently flexible that it can be set up to simulate the function of many types of devices, and 2) be easily embedded within a suitable range of curricula and training environments to assist the instructor in meeting the students' learning needs. The first application of the IMTS will be in training corrective maintenance of the SH-3H helicopter's bladefolding subsystem. In this setting the IMTS will be interfaced to the Generalized Maintenance Training System (GMTS), a simulator set up to simulate the surface behaviors of the Bladefold system using video disc and graphic overlays. The GMTS system will have its own simulation database for the helicopter Bladefold subsystems. The role of the IMTS in this environment will be to assess student performance on the GMTS, to intervene when necessary, and to provide supporting guidance in performing the diagnostic activities.

System Overview

The instructional process performed by the IMTS involves the following major steps: 1) it selects malfunctions within the target system which will most effectively exercise the individual students at their current stage of understanding and proficiency, 2) it inserts the malfunctions into the simulations of the target system and allows the students to manipulate the simulation much as they would manipulate the real system, 3) it simulates the response of the system to student actions, providing an opportunity to practice diagnostic tasks, 4) it provides 'within-problem' support, as necessary, to ensure that students proceed to problem completion in a productive manner while exercising their problem-solving skills as much as possible, and 5) it provides 'between-problem' support, as necessary, to resolve more general deficiencies. Thus the IMTS plays the role of the instructor and, in a stand-alone configuration, it simulates the actual equipment.

Organization of the Report

Section II outlines the premises and objectives which shaped the design of the IMTS. Section III describes the simulation authoring process implemented in IMTS. Section IV describes the underlying system model and the techniques for inferring system behavior from that model. Section V presents a description of Profile, the subsystem within the IMTS which models an expert troubleshooter. It is this process which allows the IMTS to demonstrate expert diagnostic strategies, to evaluate student performance, and to assist learners in completing practice problems. Section VI presents conclusions and a brief discussion of future planned developments.

SECTION II.
PRINCIPLES UNDERLYING DESIGN OF THE TRAINING SYSTEM

This section is concerned with briefly summarizing the issues which influenced design of the IMTS.

Hardware Issues

A number of characteristics of the IMTS were determined by issues related to the costs and capabilities of computer hardware and systems for user interaction. The IMTS was envisioned as a system which would operate upon off-the-shelf hardware which was manufactured in quantity and sold and maintained commercially. This also implied that general-purpose media would be used for student-computer interactions, i.e., no special hardware would be constructed to emulate characteristics of particular equipments being taught. It was also a design objective to employ media which would allow instruction developers to execute, check, and revise their emerging simulations as a continuous and integral part of the development process.

Instructional Issues

From the beginning the IMTS was planned to be simulation-based and responsive to student actions, i.e., the IMTS would emulate the behavior of the real equipment as students carried out diagnostic functions upon the simulation. The instruction would be presented primarily in response to student actions, rather than being scheduled in a frame-based manner. This approach addresses a critical need in the services to provide individual students an opportunity to practice diagnosing faults in a self-directed rehearsal of field conditions, yet be supported in a manner which identifies and resolves learning deficiencies and minimizes unproductive practice time.

Another decision made early in the planning phase was to place some operating characteristics of the IMTS under the control of the instructor. This decision was based upon three realities. First, instructors often know of time constraints, problems with training in prerequisite areas, and entering class characteristics. The instructor should be able to adjust the characteristics of the training system to meet local needs, rather than subjecting students to inappropriate training while the instructional system is adapting to the conditions. Second, since the IMTS is not designed to automatically adjust its processes based upon experience with *previous* students, the instructor is a vital mechanism in the control loop.

Finally, as a practical matter, instructors are more receptive to a training device in their classroom if they can have some control over its behavior. Because excessive or persisting requirements for instructor input can produce other types of resistance, the instructor control actions are set up to be simple and entirely optional.

Like all the previous computer-based training systems we have developed, the IMTS is not intended to eliminate the human instructor. Instead, it is viewed as a potentially powerful and intelligent aid to the instructor which can take over a massive workload in dealing with individual students as they work exercises, thereby freeing the human instructor for preparation and presentation of other instructional material, and for dealing with unusual problems. As a research goal it is challenging to attempt to automate as many of the instructor's functions as possible. It is clear, however, that there remain a number of critical instructional functions which are currently performed effectively only by a skilled human instructor.

The design of the IMTS is also based upon an intensive analysis of corrective maintenance performance and diagnostic expertise, involving detailed observation and analysis of nearly 600 diagnosis and repair sequences for 87 different technicians (Towne, Johnson, and Corwin, 1982, 1983). This research played an important role in forming basic concepts of the IMTS design, and it provided the data upon which to construct a model of expert diagnostic performance, called Profile. The Profile model (see section V) forms the central resource for evaluating and remediating student performance.

Alternatives for System Simulation

There are many methods for constructing interactive graphic simulations for training, differing from each other in such qualities as authoring difficulty, accuracy of the constructed simulations, simulation maintainability, and others. Figure 1 roughly evaluates some different approaches to building simulations using six criteria.

	Programerless Interactive Interface	Generated System Behaviors	Object Behavior Editing Tools	Graphics Editing Tools	Direct Manipulation of Objects	Can Be Object Oriented
Programming Language (e.g., Lisp, C, Pascal)						√
Expert System Shell (e.g., OPS5)		Partial	Partial			√
Interface Editor (e.g., STEAMER)	√		√	√	√	√
CAE (e. g., Mentor Graphics)		√		Partial	Partial	Partial
Surface Simulation (e.g., BMTS, ESAS)	√	√			Partial	
Deep Simulation Tools (e.g., IMTS)	√	√	√	√	√	√
Animation Tools (e.g., Videoworks)	√	√		√		

Figure 1. Features of Tools for Composing Interactive Simulations for Training

Programming Languages

The most complex approach to simulation composition is to write a new computer program for each new application that is to be developed, using a general-purpose programming language such as Pascal, C, or Lisp. This approach is extremely laborious and may result in simulations that are difficult to maintain. When it is done very well, however, it can result in more accurate and responsive simulations than can be produced by any other means. The greatest advantage of custom programmed simulation is that there are few restrictions on the equipments or processes which may be handled.

Expert System Shells

Expert system shells offer partial support for the automatic generation of system behaviors. A domain expert can enter rules describing the behavior of the elementary objects of a simulation, and the shell will generate an application that applies these rules in response to changes in the states of the objects. Execution of these rules can cause effects to propagate, simulating the behavior of the system simulation. Conventional expert system shells, however, lack the graphic object editing tools as well as the scene construction tools that automatically generate system topology in the course of scene composition. Building such a user interface using conventional expert system tools would be difficult and expensive.

Interface Editor

STEAMER (Hollan, 1983; Hollan, Hutchins, & Weitzman, 1984) provides an example of a system that does not require computer programming skills to produce the user interface (that portion of a simulation/training system that allows users to manipulate the states of simulated components, to change the displayed views, and to access pedagogical features). STEAMER is a user interface editor that allows the construction of graphical objects which are then linked to values in an existing simulation program. Graphic editing tools are used to create, modify, and position depicted objects.

STEAMER lacks the ability to generate behaviors of the simulated system. A separate program is required to determine the correct values which feed the displayed STEAMER objects during a simulation. While the development of the simulation program can be more difficult, it allows the use of STEAMER not only to build simulation interfaces, but also to construct user interfaces to other programs. Experimental STEAMER interfaces have been constructed for a number of UNIX utilities, for example.

CAD/CAE Systems

Computer Aided Design (CAD) and Computer Aided Engineering (CAE) systems provide a graphical interface for the composition of scenes, but there is typically no graphically depicted interaction with an end user that is automatically provided as a consequence of the composition activities. Commercial CAE (computer-aided engineering) systems generate system behaviors for electrical and electronic systems, using precise formulas to determine values at nodes in the system topology. Their primary limitations are that they are generally restricted to such limited technologies as electronics or air conditioning, and they are not configured to be highly interactive.

A CAE system typically provides an object library that includes code modules defining the behavior of each object. When CAE users want to add new objects similar to an existing type they can essentially program a new version of the type by specifying a number of parameters. When the new object is not related to an existing type, or when they want to create a more efficient "black box" module to replace an assembly that includes many standard components, they must write new code modules (usually in C or Pascal), compile them, and link them into the object library or a special user-defined library.

Surface Simulation Systems

Surface simulation systems do not incorporate specific models of the propagation of effects among simulated objects. Instead, they are designed to reflect the surface effects of simulated element manipulations. Even quite remote effects may be directly referenced in a surface simulation system. Some surface simulation systems, such as the Generalized Maintenance Training Simulator (GMTS), may be scene oriented rather than object oriented. GMTS is a static-scene simulation system based on a simulation construction system developed at Behavioral Technology Labs in the late 1970s. Under funding from the Navy Personnel Research and Development Center, this simulation composition and presentation tool was modified by developers at Cubic Corporation, ManTech/Mathetics, and Systems Engineering Associates. Several versions are now distributed by Cubic Corporation. GMTS simulates the behavior of systems through the presentation of static videodisk images in response to student touch inputs. A medium-resolution graphics overlay system is used to provide textual annotation and simple simulation graphics on the video display screen.

Altering the setting of a control in a scene-oriented surface simulation system results in a replacement of the entire depicted scene, even though only a single object needs to be changed. GMTS is a system that emphasizes surface simulation fideliiy. Its color video displays of actual equipment panels and other system components help to ensure that students will be able to transfer the material they learn to the actual equipment.

Not all surface simulation systems are scene oriented, however. ESAS, the Equipment Simulation Authoring System (Towne & Munro, 1984) is a simulation composition system that is object oriented at the user interface. Independent graphic objects can be manipulated by students and are displayed and altered individually on the screen.

These objects, however, do not propagate their effects in any manner which approximates the propagations in the real equipment. Instead, rules for determining the surface appearance of all potentially affected objects must be evaluated whenever students manipulate an object.

Deep Simulation Tools

IMTS, like many CAE systems, automatically constructs simulations using the behaviors of the object components selected from a library, together with the topology of the described system. Unlike most of these systems, it also allows users to add new objects to the library without writing programming language code to describe the behavior of those objects.

For a general-purpose simulation construction system, such as IMTS, it is not practical to try to provide a comprehensive library of object elements that can be used to compose simulations of any equipment system, whether it be electrical, electronic, mechanical, or hydraulic. One constraint is that the number of object types that would be required is simply too large. Another is that the level of object description that is appropriate for training simulations is usually quite a bit higher than that which is appropriate for CAE. This means that many equipment simulations may require the definition of idiosyncratic objects, which could not be expected to pre-exist in the library.

Animation Tools

An example of a software system that can be used to create simple graphic simulations is the Videoworks™ application on the Apple Macintosh™ computer. This type of simulation construction is relatively easy, but the "simulations" constructed are inflexible and do not permit exten;ive user interactions. A simple animation application such as Videoworks may be the appropriate tool for the simple equipment simulations that do not call for intensive interactive training. For military and commercial training purposes, accurate high-quality simulation training is essential.

Authoring and Instructional Features

Graphic simulation systems such as STEAMER and IMTS provide a user interface which allows the user to manipulate depicted objects directly and then immediately observe graphical consequences in other objects.

In order to avoid bringing a computer programmer into the simulation construction process using IMTS, an object behavior editor was developed. This software module lets the subject matter expert describe the behavior of a new graphic object using a simple data entry form. The author uses this editor to specify the input/output ports of the object, to describe the object's transformations on port values, and to describe the conditions under which the object changes state (and possibly appearance). Other methodologies can provide some of the features of object behavior editing without programming that IMTS does. For example, STEAMER makes it possible to use object graphics editing tools to specify the correspondences between certain simulation values and aspects of the appearance of an object. IMTS extends the non-programmer's control over simulation objects by allowing the specification of value transformations and underlying state changes as well as object appearance changes.

Graphics editing tools. An ideal simulation composition system should provide two kinds of graphic editing capabilities. First, authors need an object appearance editor to build the possible appearances of new object types. Second, a scene composition editor is required to construct simulation scenes from instances of the object types and to add text and background graphics features to the scene. CAE packages typically provide the latter type of editing capability but not the former; STEAMER and IMTS provide both.

Direct manipulation of objects. Both authors and students should be able to manipulate objects and scenes through direct manipulation activities, such as pointing at the screen with a finger or selecting with a mouse, rather than through less direct, more symbolic actions such as typing commands. (See Norman & Draper, 1986, for a thorough discussion of the qualities of direct manipulation user interfaces.)

Object orientation. Many simulation composition systems, including IMTS, STEAMER, and CAE have an object-oriented user interface, one in which individual objects can be independently manipulated and moved. The IMTS emphasizes understanding and remediating student misconceptions about how the target equipment system works and how to troubleshoot it.

Because GMTS is inherently a surface simulation system, there is no underlying representation of cause and effect in the behavior of the complete system. In an IMTS simulation, on the other hand, surface (or system-level) behavior is automatically derived from what is known about the behavior of elements and how they are connected. This permits the automatic generation of more insightful and intelligent instruction than is possible in a purely surface simulation. Authoring is more straightforward in a deep simulation system such as IMTS, because the author does not have to independently consider all the possible combinations of control settings and their effects on indicators in the system. Because all global behavior is derived from local effects, a more modular approach to simulation authoring is possible, with attendant benefits in building, documenting, and maintaining a simulation.

In addition to having an object-oriented user interface, a deep simulation system should also have an object-oriented implementation of equipment behavior. At the implementation level, a STEAMER simulation may not be fully object-oriented, since the custom simulation program it interacts with may not use separate modules to compute the effects of the depicted objects. IMTS simulations are object-oriented at this implementation level, since the simulations are automatically constructed using the behavior modules of the objects included in that simulation's scenes. Surface, system-level, behavior is automatically derived from what is known about the behavior of elements and how they are connected. This permits the automatic generation of more insightful and intelligent instruction than is possible in a purely surface simulation. Authoring a complete simulation is more straightforward in a deep simulation system, because the author does not have to independently consider all the possible combinations of control settings and their effects on indicators in the system. Because all global behavior is derived from local effects, a more modular approach to simulation authoring is possible, with attendant benefits in building, documenting, and maintaining a simulation.

Quantitative Precision in Simulation

The different tools for composing interactive training simulations described above provide differing degrees of quantitative precision in the simulations. Systems that are not object-oriented, such as animation tools and surface simulators, do not compute simulated values at all, but simply display indicator values as they have been pre-authored. Among the simulation systems that interactively compute simulated object values, there is a great range of precision in those computations. CAD/CAE systems with simulators, such as **Spice**™, compute all simulated values to a high degree of precision, using very accurate computational models of the objects provided in the CAE library and simulation algorithms specific to analog or digital electronics. On the other hand, some simulators used for training, such as that developed by Govindaraj (in press), do not compute precise quantitative values at all. This type of simulator uses a qualitative approach to the representation of simulated values.

Qualitative models have received a good deal of attention in studies of reasoning about physical systems (Davis, 1984; De Kleer, 1984; De Kleer & Brown, 1984; Forbus, 1984), but they are less commonly used to represent the behavior of physical devices for simulation training.

There is a continuum of precision in simulation computations. The IMTS simulator falls in the middle of this continuum of simulation precision. The object rules in IMTS may be cast in precise quantitative terms if necessary in order to compute exact outputs. While the object rules in the Bladefold application are quantitative, they generally reflect rather broad ranges of values which dictate object behavior. For example, a typical object rule states that the object enters a particular state if a value at an input port exceeds some threshold value, and a typical output specification states that the output is some fixed value. Alternatively, the output rule could be a complex function of the input values.

While the IMTS object rules can be of high precision, the simulator routine is a general-purpose algorithm which does not perform global computations, as systems like Spice™ do. Thus IMTS simulations will be completely accurate only for systems in which local object rules and system topology are sufficient to fully characterize the processes. In spite of the complexity of the Bladefold system, it meets this requirement and the simulation is correct. Moreover it appears that most digital, mechanical, hydraulic, and electrical systems can be simulated with sufficient precision to meet training requirements.

On the other hand it is clear that the IMTS simulator will not produce highly accurate results for complex analog systems involving parallel circuits. One possible solution to this limitation is to add some special-purpose simulation algorithms for dealing with such important system characteristics.

SECTION III. SIMULATION AUTHORING

Philosophy

One approach to providing consistently high quality training is to provide instruction developers with tools that produce accurate interactive simulations without requiring programming skills, fault-evaluation skills, or sophisticated pedagogical expertise. Much of our efforts have been devoted to providing extensive tools for extracting device-specific knowledge from authors who are neither computer programmers nor training specialists. To the largest extent possible, instructional interactions are automatically guided by the reliable, factual data base so extracted. The same underlying data base is used to drive several different types of instructional presentations and interactive environments, thereby increasing the payoff for the simulation authoring time expended by the device expert. For example, the expert troubleshooting model, Profile, obtains the data it needs by automatically inserting possible failures into the simulated system and observing their effects, as determined by the IMTS simulator routine.

IMTS simulations are constructed using an editor that incorporates a *direct manipulation interface* which allows 'drawings' of the system architecture to be created in terms of graphical objects. The authoring process is therefore similar to that employed in CAE systems. Like a CAE system, the behavior of a complete simulation is determined by the behavior rules of the individual objects and by the topology of the system, and does not require the authoring of system-specific simulation rules. Unlike CAE systems, however, the IMTS simulation responds graphically, as well as computationally, to actions upon it. Thus users can observe the responses of the system to their actions, rather than having to analyze more abstract representations of system behavior such as timing diagrams or table of node values.

The approach used in the IMTS for relating the graphical appearance of an object to its role and state within a particular system was heavily influenced and inspired by work on STEAMER (Hollan, 1983; Hollan, Hutchins & Weitzman, 1984). STEAMER allows experts to construct interfaces between existing simulations of particular systems to graphical 'objects' which display their response to system conditions. When attached to a particular point in a system by a content-expert, the generic objects are able to determine their reactions and appearances under specific input conditions. As a student alters the system configuration by setting switches, the intelligent objects respond by changing their appearances appropriately. IMTS extends the STEAMER concept by providing tools for constructing the underlying simulation based on graphical elements and object behavior rules, avoiding the need for a separate simulation programming step.

While the more interesting research question concerns approaches for minimizing the extent of human intelligence required to support instruction, practical training requirements also demand that the IMTS provide a means foı capturing and communicating that portion of human knowledge about systems which cannot yet be produced artificially. Simple authoring tools have therefore been created for adding more customized instructional content to the problem-solving environment created automatically. These tools make it possible for the IMTS to 1) present instructional or admonitory texts under particular situations anticipated by a human device expert, and 2) present an interactive demonstration and explanation of a procedure, following the steps of a human device expert. The quality of these instructional products depend heavily upon the skills of the author, and a greater variation in the quality of the instructional materials built with these tools can be expected.

The Simulation Authoring Process

Simulation/training is produced for a new target system by describing its architecture, i.e., its components and connections. From this specification the IMTS infers the system's behaviors under whatever conditions the student produces by manipulating the switches and controls, it determines how each malfunction will affect the system indicators and test points, and it computes diagnostic sequences which it can demonstrate to the student or employ when the student needs assistance.

The configuration-editing system used to create the specification (Figure 2) is composed of four main units: 1) an *object construction editor* , for defining the graphical appearance and rules of operation of generic objects, 2) a *generic object library*, for storing object specifications, 3) a *system construction editor* for assembling the generic objects into single-screen views of subsections of the target system, and 3) a *fault simulator* capable of determining the behavior of the total system under any mode and fault condition. The simulator is included in the configuration editing system to allow authors to execute, check, and revise simulations without leaving the simulation- construction mode. Detailed simulation authoring procedures are provided in the IMTS Users' Guide (Towne, Surmon, Pizzini, Penrose, & Munro, 1987).

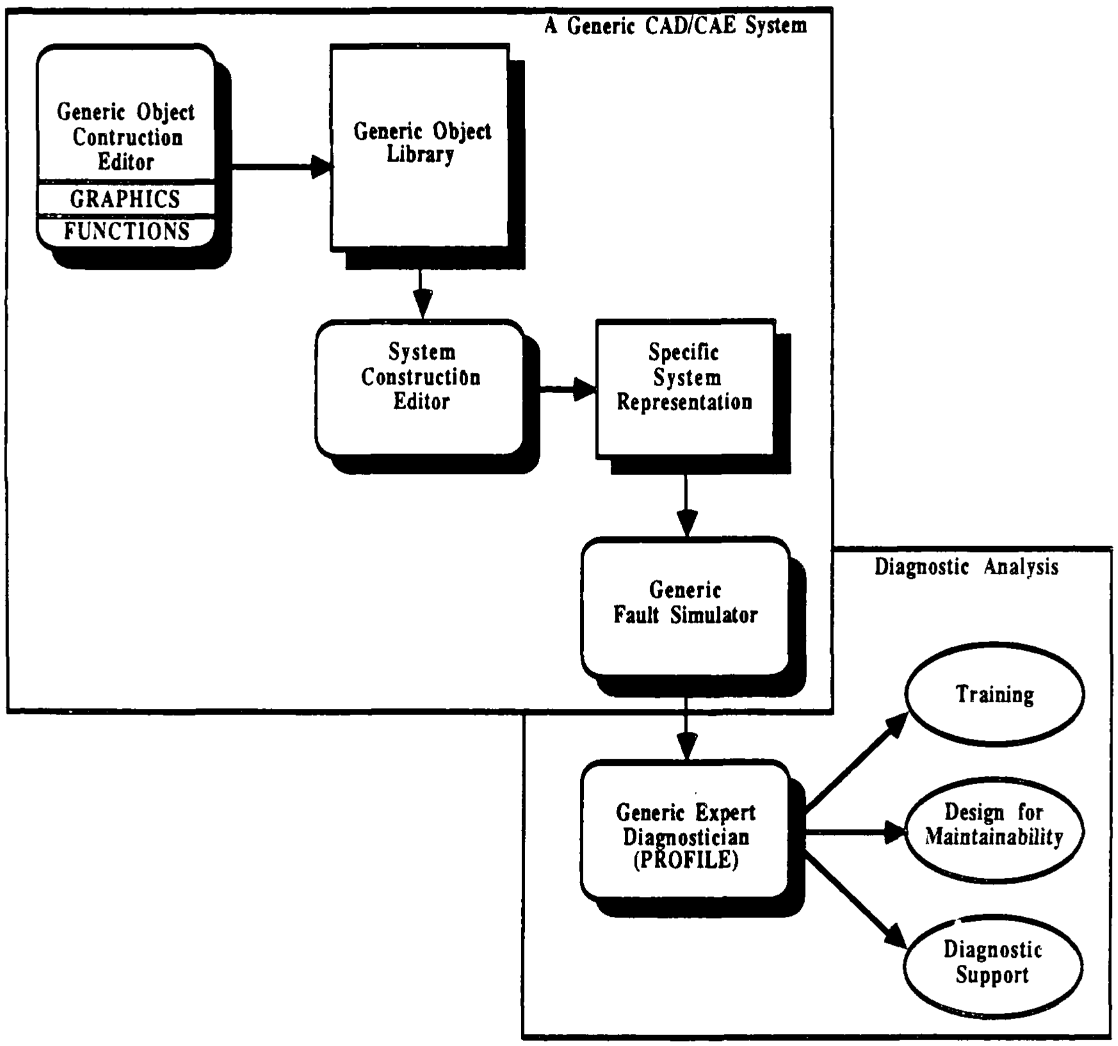

A Generic CAD/CAE System
Generic Object Contruction Editor
GRAPHICS
FUNCTIONS
Generic Object Library
System Construction Editor
Specific System Representation
Generic Fault Simulator
Diagnostic Analysis
Generic Expert Diagnostician (PROFILE)
Training
Design for Maintainability
Diagnostic Support

Figure 2. Components of the IMTS System

Creating a New Graphical Object

If the simulation author determines that the existing library of generic objects lacks a required object, he constructs it, first using an editor for describing the object's possible graphical appearances. This involves constructing on the screen that part of the object which does not change, called the *static* part, then entering the graphics which change according to the state of the object, termed the *state-dependent* part. Figure 3 shows an object in its two states.

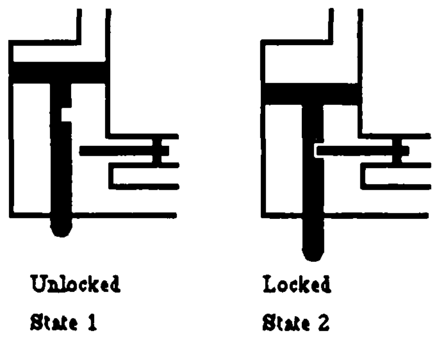


Figure 3. An Object in its Two States.

Neither the graphics editor nor IMTS place restrictions upon the form of the graphics which can be used to represent a component. While Figure 3 illustrates a 'schematic' representation of a component, the author could just as easily use a form which is physically representative of the part. Thus a toggle switch could either be represented in a schematic form or it could be drawn to look like a toggle switch (in two possible positions).

The generic object library currently contains all the components necessary to simulate the Bladefold system. Bladefold is a moderately complex, electrically controlled hydraulic system which controls the position, movement, and orientation of the blades of the SH-3H helicopter after landing. While we have defined only those objects required to simulate the Bladefold system, such as wires, switches, indicator lights, meters, valves, relays, and pipes, these objects would take authors a long way toward simulating many new systems. The examples which follow all relate to this particular application of the IMTS.

Getting Objects to Behave

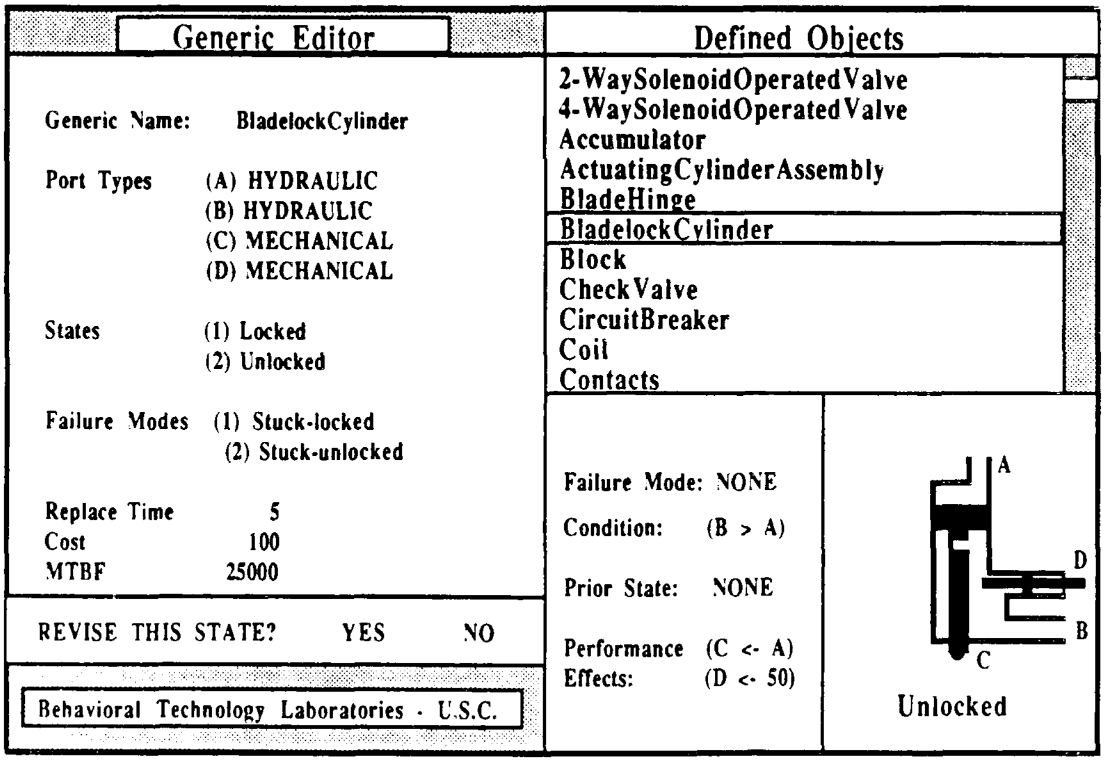
Once a new object is defined graphically it must be provided its rules of behavior. This is done within a special generic-object behavior editor, as shown in Figure 4. This editor provides four windows as follows:

a. The Defined Objects window, in the upper right-hand part of the screen, lists the names of all objects whose graphics have already been created; the user scrolls through this window and selects the name of the object previously created in the graphics editor.

b. The Object Display window displays whatever object has been selected; as the user steps through each object state the display cycles to the proper state.

c. The Constant Object Specification window, on the left of the screen, prompts the user for information about the object which is not state-dependent. The user identifies the types of inputs and outputs the object processes, such as hydraulic versus electrical or mechanical, as well as information about the object which is important to the Profile troubleshooting model (replacement time, spares cost, and mean-time-between-failures, MTBF).

d. The State Definition window, at the lower center of the screen prompts the user for information about each state of the object. Here are entered the rules governing what processes the object performs upon its inputs in each state, and the conditions under which the object enters each state.



Generic Editor
Defined Objects
2-WaySolenoidOperatedValve
4-WaySolenoidOperatedValve
Accumulator
ActuatingCylinderAssembly
BladeHinge
BladelockCylinder
Block
CheckValve
CircuitBreaker
Coil
Contacts
REVISE THIS STATE? YES NO
Behavioral Technology Laboratories · U.S.C.
A
B
C
D
Unlocked

Generic Name: BladelockCylinder

Port Types (A) HYDRAULIC
(B) HYDRAULIC
(C) MECHANICAL
(D) MECHANICAL

States (1) Locked
(2) Unlocked

Failure Modes (1) Stuck-locked
(2) Stuck-unlocked

Replace Time	5
Cost	100
MTBF	25000

Failure Mode: NONE

Condition: (B > A)

Prior State: NONE

Performance Effects: (C <- A) (D <- 50)

Figure 4. The Object Behavior Editor.

The object shown in Figure 4, called a BladelockCylinder, has only two graphical states, but it has four different behavior states. The first two states, Locked and Unlocked, are normal states. Two other possible states are failure conditions: Stuck-locked and Stuck-unlocked. The functioning of this object in the normal Locked state is identical to its function in the abnormal Stuck-locked state, however the conditions under which the states occurs differ. Thus if IMTS encounters a normally operating BladelockCylinder in its simulation, it will evaluate the values at the object's input ports to determine what state the object enters and what outputs it produces. If, on the other hand, it encounters a failed BladelockCylinder, it finds that the input port values are irrelevant, and sets the output values which result from the malfunction.

Virtually all the components of the Bladefold system are elements which exist in a few discrete states, such as extended/retracted, locked/unlocked, or on/off. While these happen to be common in the Bladefold system, the IMTS is *not limited to simulating discrete-state elements*. The pressure meter, for example, is an object which performs the simple function of sensing the pressure in a pipe and reflecting that reading via a needle. One could just as easily define an amplifier whose function is to output the square of its input.

There may be objects in systems whose current state depends in part upon their prior state (like a flip-flop, for example). The generic object editor offers the object-definer the means for including an object's previous state into its definition of current state.

Creating Object Functions

The main task of the object behavior editor is to create Lisp functions for each object described. Figure 5 displays a *normal* RotorBrakeCaliper (like a car's disk brakes) in its two possible states, along with the rules which accompany each state. If the pressure at port A is less than 200 psi or if some other object is exerting a force greater than 50 pounds directly on the brake calipers at port B, then the RotorBrakeCaliper object enters the BrakeOff state. This IF-THEN expression is termed the System Condition. In this state the performance effect, which is propagated to adjacent objects, is that no force is exerted upon them.

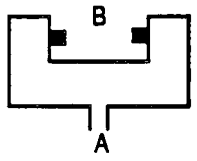
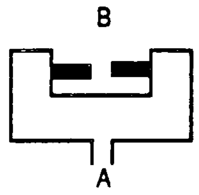
State Name	BRAKEOFF	BRAKEON
Graphic		
System Condition	((A < 200) OR (B > 50))	((A >= 200) AND (B < 50))
Performance Effects	(B ← 0)	(IF (A >= 200) THEN (B ← 50) ELSE (B← 5))

Figure 5. The Object Behavior Rules for the Normal RotorBrakeCaliper

Figure 6 displays the Lisp code which the generic object editor generated from the rules of Figure 5. The object functions employ fixed indices to array structures to reduce compute time, thus there are many integers involved in the functions. Additional object functions are automatically produced for each failure mode for the object.

```
(LAMBDA NIL
    (PROG   ((STATE (Get 0))
             (NEWPORTNUM (Get 54))
             NEWSTATE a b)
            (Put 56 NIL)
            (COND
                ((AND (EQP NEWPORTNUM 2)
                       (Get 4))
                 (Reverse 4)))
            (COND
                ((EQ STATE (QUOTE BrakeOff))
                 (if   (OR (NOT NEWPORTNUM)
                       (EQP NEWPORTNUM 2))
                       then (if (NOT  (Get 4))
                             then (SETQ b (QUOTE (A . 0)))
                       elseif (NOT (EQP (CDR (Get 4))
                                        0))
                       then (SETQ b (CONS (CAR (Get 4))
                                                   0))))))
                ((EQ STATE (QUOTE BrakeOn))
                 (if (OR (NOT NEWPORTNUM)
                         (NUMBERP (CAR (Get 2))))
                      then
                      (if (GEQ (CAR (Get 2))
                                200)
                       then (if (NOT (Get 4))
                             then (SETQ b (QUOTE (A . 50)))
                             elseif (NOT (EQP (CDR (Get 4))
                                              50))
                             then (SETQ b (CONS (CAR  (Get 4))
                                                              50)))
                else (if (NOT  (Get 4))
                       then (SETQ b (QUOTE  (A . 5)))
                        elseif (NOT (EQP (CDR (Get 4))
                                         5))
                       then (SETQ b (CONS (CAR (Get 4))
                                              5)))))))
            (COND
                ((OR (AND (NUMBERP (CAR (Get 2)))
                               (LESSP (CAR (Get 2))
                                      200))
                         (AND (NUMBERP (CAR (Get 4)))
                              (GREATERP (CAR (Get 4))
                                        50)))
                (SetState2  (QUOTE BrakeOff)))
                 ((AND (NUMBERP (CAR (Get 2)))
                          (GEQ  (CAR (Get 2))
                                200)
                          (NUMBERP (CAR (Get 4)))
                          (LESSP  (CAR (Get 4))
                              50))
                (SetState2  (QUOTE BrakeOn))))
        (Put 0 (NewState))
        (Put 2 a)
        (Put 4 b)
        (Put 54 2)))
```

Figure 6. Lisp Function for Normal RotorBrakeCaliper

The Library of Generic Objects

When an object has been defined both graphically and behaviorally, it is stored in a general library which can be used as a resource by any simulation author. A portion of the generic library is shown in Figure 7.

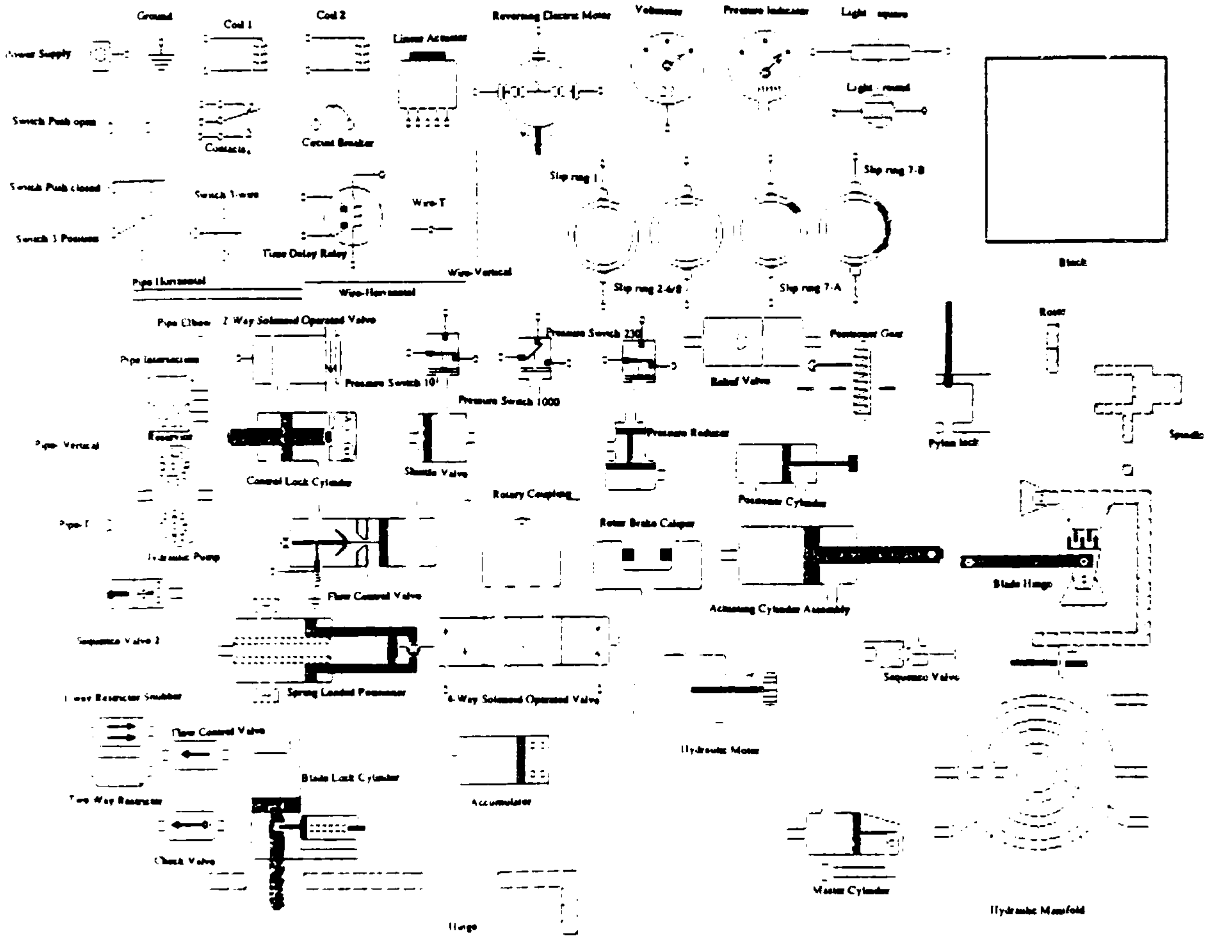


Figure 7. The Generic Object Library

Constructing Simulation Scenes

The content-expert constructs a specific system simulation (and all associated training interactions) by selecting appropriate objects from the library and positioning them on the screen, using a special graphics editor. This is the easiest authoring function. The job of constructing the simulation is primarily one of subdividing a big system into separate screens, or scenes, and then producing each individual scene. Normally, existing technical diagrams serve as an excellent starting point for creating the scenes. It is also common, however, that drawings from technical references are incomplete or incorrect, and not created for ease of understanding. Typically the simulation author will massage the drawings considerably before being satisfied with the clarity and accuracy of the displays.

As the objects are positioned, the scene editor detects the connections between elements, and it retains the connectivity data in a file. The connectivity data are necessary but not sufficient to compute how a system will behave under a current condition. The IMTS uses the connectivity information plus the behavior rules of each object involved to determine the *nature of the signal conversions*, and hence the particular appearance of system elements.

As an aid to verifying the correct operation of scenes, the scene editor includes the IMTS simulation programs. This allows the scene author to manipulate switches and input values in each scene to check for correct responses. Any errors in operation can then be traced either to 1) incorrect definition of a generic object, or 2) incorrect identification and connection of the objects within the scene.

Producing the Top-level Diagram

When all the individual scenes have been created, it is necessary to 1) complete any connections between scenes, and 2) construct a diagram which represents the entire target system. Connections between scenes are made by identifying the ports in one scene which are connected to ports in another scene, in a manner similar to that used to connect ports within scenes. A simulation composition editor is provided for this purpose. After the individual scenes have been connected, the final step is to construct a simple block diagram which reflects the general organization of scenes and to identify the scene which corresponds to each block. Figure 8 illustrates the top-level diagram for the Bladefold system. To view or manipulate a different scene of a large simulation, the student selects the block representing the scene of interest. The selected scene then appears in the main IMTS simulation window with all objects shown in their current states.

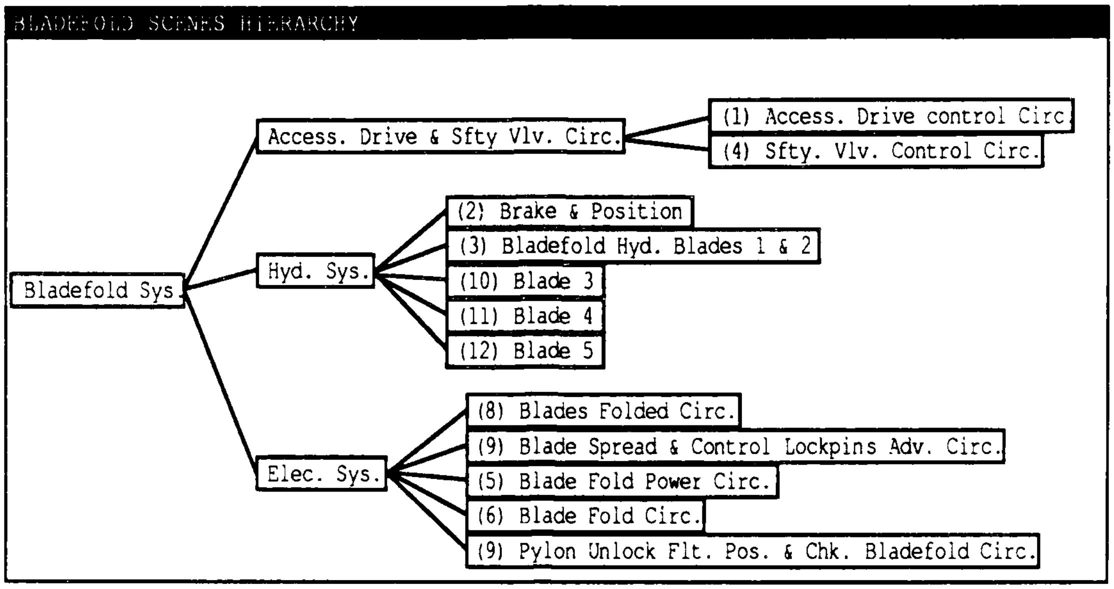

BLADEFOLD SCENES HIERARCHY
Bladefold Sys
Access. Drive & Sfty Vlv. Circ.
(1) Access. Drive control Circ
(4) Sfty. Vlv. Control Circ.
Hyd. Sys.
(2) Brake & Position
(3) Bladefold Hyd. Blades 1 & 2
(10) Blade 3
(11) Blade 4
(12) Blade 5
Elec. Sys.
(8) Blades Folded Circ.
(9) Blade Spread & Control Lockpins Adv. Circ.
(5) Blade Fold Power Circ.
(6) Blade Fold Circ.
(9) Pylon Unlock Flt. Pos. & Chk. Bladefold Circ.

Figure 8. Top Level Diagram for Bladefold

Methodologies for Authoring Procedures Training

Although IMTS is capable of generating a great deal of instruction from the simulation data provided by authors, it cannot automatically construct all of the kinds of simulation-based instruction that may be desired. The IMTS generic expert does a good job of guiding equipment troubleshooting practice using the simulator, but there are other kinds of instruction that should be provided for maintainers. One such type of training is *fixed procedures training*. In many military and industrial environments, there are prescribed sequences of activities that equipment maintainers are expected to carry out. Examples include periodic maintenance activities, including adjustments and calibrations, and fixed checkout procedures which may involve actions performed for safety, security, or other reasons which cannot be anticipated entirely from the design of the system.

Many of the same considerations that apply to methodologies for authoring equipment simulations must also apply to the authoring of fixed procedures training. Authoring such training should not require programming and should make good use of direct manipulation methods in the authoring process. One way to avoid programming and to make use of direct manipulation is to use a procedures-authoring process that makes use of the existing equipment simulation in an authoring-by-example mode. That is, the author of a fixed procedure should be able to put a simulation in *record* mode and simply carry out the procedure while adding explan: tory remarks. When students study a fixed procedure, a *playback* mode uses the recorded sequence as a template for the student's actions.

SECTION IV. SIMULATING EQUIPMENT BEHAVIORS

Variables Affecting Equipment Behaviors

In general, the behavior of a time-invariant, real-world system is a function of the malfunction state, of settings of controls, and of previous states, as described below.

Malfunction State

The simulation process is basically the same when IMTS simulates a normal, fail-free system and when it simulates a system that contains a malfunction. When the problem-selection routine identifies the malfunction which will provide the best practice to the student, it simply changes the type-name of the failed part in the target system from its standard generic name to the name of the particular failed part. For example if the Main Power Switch in a system is to be failed in an open condition, its type name would be changed from ToggleSwitch-2Wire to ToggleSwitch-Open. When the IMTS simulator wishes to evaluate the MainPower Switch it would then be led to evaluate the behavior rules for an open switch rather than a properly functioning switch.

Since the failed version of the part contains rules of behavior in exactly the same form as normal parts, the simulator does not need to distinguish between simulating a normal system and a failed system, although other routines in the IMTS do note whether symptoms seen by the student are normal or abnormal. When the student calls for replacing the part which the IMTS has failed, the IMTS simply restores the internal generic name of the part to that of the properly-operating generic type. Some attractive implications of this are:

- students may request the introduction of failed components, to explore their effects.
- a wide range of failure modes may be specified for a component, allowing selective simulation of abnormalities.
- multiple failures may be introduced with virtually no complication in the simulator or in the authoring process.
- 'cascading' failure effects, in which a failure in one component or an improper equipment mode causes a failure in a second component, can be simulated correctly (with a slight modification to the object behavior editor)

In addition to accommodating virtually any object failure, including opens in wires and pipes, the IMTS also allows either the student or the surrogate instructor routines to produce and simulate short circuits between any two points in the system. Such failures involve topological change rather than failure of a component to perform its normal functions. This same capability is also used when the student wishes to use jumper wires to delete a section of the system by connecting a source signal directly to some distant point in the system.

Settings of Controls

Changing the setting of a single control can drastically alter the set of elements of a system that are involved in system operation and the way in which they are interconnected. In the Bladefold application, a single switch change can trigger mechanical movements which actuate other microswitches, which in turn trigger other mechanical, hydraulic, or electrical functions. In the real SH-3H aircraft the responses to a single switch change can go on for over thirty seconds. For these reasons the simulator functions in the IMTS are extremely compute-bound and far more complicated than originally expected.

Previous States

Many complex systems, including Bladefold, operate in ways which are history-dependent, i.e., the state they enter is affected by previous states as well as current switch settings and malfunction conditions. The student/user might operate the simulated system in such a manner that some parts lock, for example, and then attempt to change configuration without unlocking the necessary parts. In this case the IMTS simulator recognizes the locking condition and it recognizes when the necessary unlocking actions have occurred. Thus the simulation is capable of responding correctly to exceedingly complex sequence constraints on operational steps.

While the IMTS simulation accurately reflects the behavior of such systems, it cannot do so unless it continually updates the simulation in response to each student action. If this were not the case, the interactions between student and IMTS could be made more rapid, for the simulation update could be deferred until the student has made all the switch settings desired to enter a different mode of operation.

Simulation Issues

Time Effects

The times during which *objects* transition from one state to another are not represented in the IMTS simulation. Thus an actuator might appear extended prior to a student action, and then appear retracted afterwards, with no animated display of the movement. While individual object transitions are not represented, the IMTS simulation display does show the **system** passing through intermediate discrete states before reaching a final status. For the Bladefold application a student action might first affect electrical connectivity which in turn causes mechanical and hydraulic effects. Often the final status of the simulation is reached when electrical contacts close, and simulated front panel indicators display the state of the system. These effects are displayed as quickly as they can be computed, rather than being based upon the times required by the actual system.

Compute Time

A problem which persists is that the time to respond to a student action is longer than we would wish, for the Bladefold application, since 1) the simulated system is extremely large and complex, and 2) the simulator must update the state of the Bladefold system following each student action. While compute time has been reduced by a factor of nearly twenty since the simulator routine was originally made operational, this speed increase was achieved almost exclusively by replacing high-level Lisp functions operating on lists with more intricately coded low-level operations on fixed arrays and direct memory addressing. This has progressed to a point where there is now almost no 'garbage-collection', or recovery of temporarily-used memory, being done by the system software. The one area where Lisp functions have been crucial is in the object behavior editor (Section III), wherein user entries describing object behaviors are converted into Lisp code.

The second technique which reduced compute time applied a special function to the defined system topology which removes pipes and wires from the underlying data structure representing the target system. While the graphic representation and the apparent operation of the system are unaffected, in reality the underlying data structure regards all object ports as being directly connected to other object ports, rather than via pipes and wires. The speed increase following this step is significant. The penalty for eliminating pipes and wires is the inability to fail these elements of the system for instructional purposes. Fortunately, there are failures in objects which are functionally equivalent to most failures of pipes and wires.

Efforts to speed simulation by preanalyzing the target system have been unsuccessful. One such approach attempted to selectively avoid recomputation of some object states, depending upon the nature of the student action. Unfortunately, the distributed architecture of the Bladefold system works to defeat this approach, as it minimizes the extent to which effects can be localized to system modules. Further Bladefold, like many other real systems, contains objects which will change state even when they become stranded from the main body of the system. Thus a simulator cannot limit evaluation to just those components encountered in a trace of connectivity.

Sequential Realism

A second major difficulty encountered during development of the simulation logic was in determining the order in which object states should be evaluated and displayed. Suppose, for example, that a pipe is connected to a tee, and that the two branches of the tee lead to other subsystems. The question arises as to which of the branches should be evaluated first. Often the order of evaluation is of no consequence, but in some cases the actions of one branch have some impact upon what happens in the other branch. If the evaluation is performed in the wrong order the system behavior can be erroneously determined. A related problem has to do with objects which perform multiple functions. Some of the objects in the Bladefold system alter both hydraulic and electrical ports. We have found that, for the Bladefold system, propagating electrical effects before propagating hydraulic and mechanical effects results in correct simulations (as long as subsequent electrical consequences of hydraulic and mechanical changes are then reconsidered by the simulator).

Locality of Effect

A factor which complicated the simulator is that an object's normal behavior rules are sometimes overruled by other objects. For example one simple object has the rule that it EXTENDs when the pressure at port A exceeds the pressure at port B, else it RETRACTs. In some situations, however, the part cannot extend because an adjacent part is obstructing it. The cause of this may be far from the part that would normally extend. The effects may chain backward through many parts, each of which is being prevented from following its rules because of the offending part. This type of complication is very apparent for mechanical parts, but it is just as serious a concern for electrical and hydraulic effects. The major implications of this effect are that object definitions must account for a wider range of situations than is initially apparent and the simulation routine must be able to backtrack when unexpected conditions are encountered.

SECTION V. THE STUDENT INTERFACE

This section will describe the interactive techniques employed in the IMTS and will outline the instructional functions it will attempt to perform. This section is intended to provide a view of the ways in which the IMTS will provide intelligent tutoring, based upon the simulation and underlying data base of a target system. While the Profile model of expert diagnostic decision-making is operational in Lisp, and the student model and problem selection processes are completed, the entire system has not been integrated at the time of writing this report, and not all instructional processes described below are yet implemented. A detailed account of instructional processes and student modeling techniques will be provided in a technical report when this work is complete.

The IMTS is designed to operate under two possible configurations: 1) a stand-alone configuration, and 2) physically coupled to a videodisc-based simulator. Student interface characteristics which are common to both configurations will be described first. The section ends with a description of the coupled configuration.

Practice Problems

Each practice *problem* consists of a malfunction, an initial equipment configuration (mode), and an operator's complaint (which may be 'none'). A particular malfunction may be involved in a multitude of problems which can differ greatly in difficulty and diagnostic activity required as a result of differing initial conditions. The IMTS first selects a problem which best fits the needs of the student, it inserts the malfunction into the simulation data for the system, it initializes the control settings, and it displays the operator's complaint (sometimes called the 'squawk') in the text display area.

The complaint presents the type of information which an operator might offer to the maintenance technician, such as 'The override light is coming on in standby mode'. In more difficult problems the complaint might not offer any starting information, or it could purposely be authored to present incorrect or inconsistent information. Problems can also involve no malfunction (since this is a common type of diagnostic situation actually encountered in the field), either with or without associated errors in the initial setup of the equipment. All of these real-world possibilities offer useful experience to more skilled students, although students should be informed if the ground rules include the possibility of incorrect initial conditions.

Displays

The responsive graphics produced by the IMTS running on a standard Xerox 1108 or 1186 computer are black and white line drawings. Objects in IMTS may be represented either in a schematic graphical form or in a physically representative form, as shown in Figure 9. The simulation author currently has the choice of representing an equipment in a manner which is physically realistic or in a way which reflects internal functions (at a future time the IMTS might allow this to be student-selected). The former approach provides more realistic operator experience, while the latter supports more informative presentations of underlying system architecture and functional system behaviors.

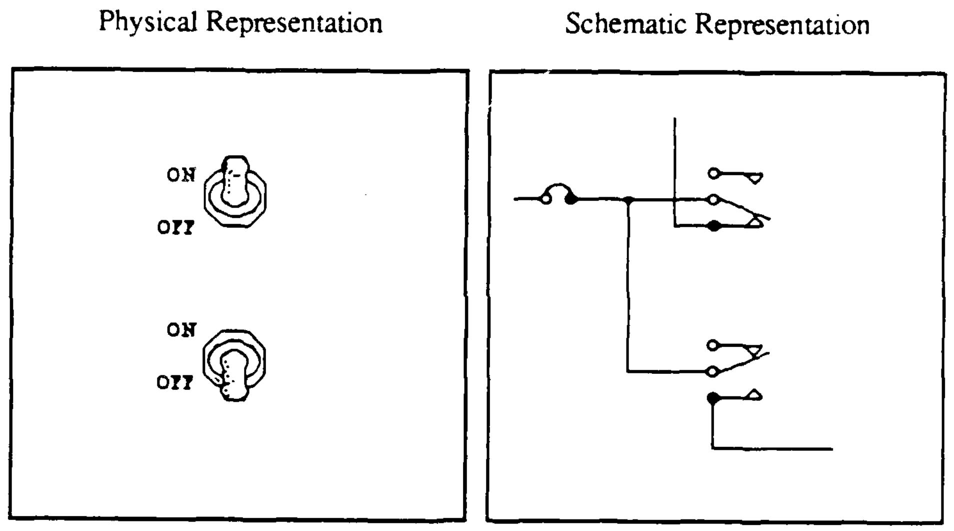


Figure 9. Alternate Representations of a Switch

In addition, static bit-mapped graphic images may be prepared using a video camera and a small digitizing unit. In Figure 10 the user has selected an object and requested a view of its physical appearance. These are the most realistic images which the IMTS can present without use of videodisc equipment.

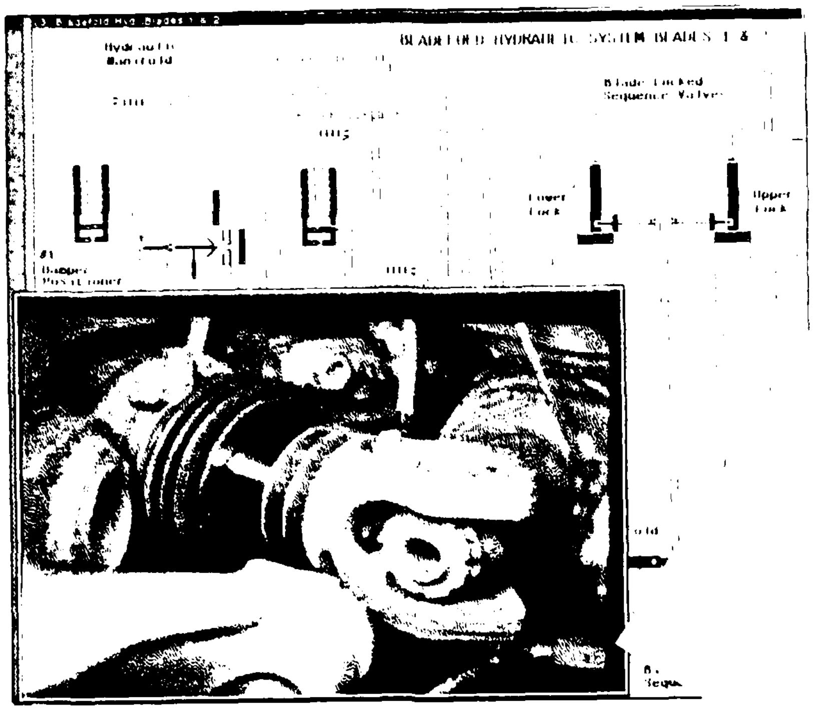

BLADEFOLD HYDRAULIC SYSTEM BLADES 1 & 2
Blade Locked Sequence Valve
Lower Lock
Upper Lock

Figure 10. A Bit-mapped Graphic Image

Figure 11 presents a view of the IMTS screen during a diagnostic exercise. There are four major sections of the display:

1. *The fixed view of the system organization*, shown at the top left. This window provides the student with a means for selecting close-up views of system subsections, for those simulations that are too large to display in one section. The Bladefold application, for example, requires twelve screens to display the entire system. Selecting one of the rectangles brings a detailed diagram into the largest window at the lower right of the screen area. The selected box remains highlighted in the upper window.

2. *The text area.* Verbal messages from the IMTS are presented in this window. These may be words generated by IMTS in response to student actions, or they may be messages created by a human expert as part of a guided simulation.

3. *The main simulation display area*, the largest window at the bottom right. In this window is shown a detailed diagram of one portion of the system. All of the objects which change appearance are displayed in their current state, according to the positions of switches and possible malfunction state. The student operates the simulated system by setting switches in this window and by observing indicators and test equipments displayed here.

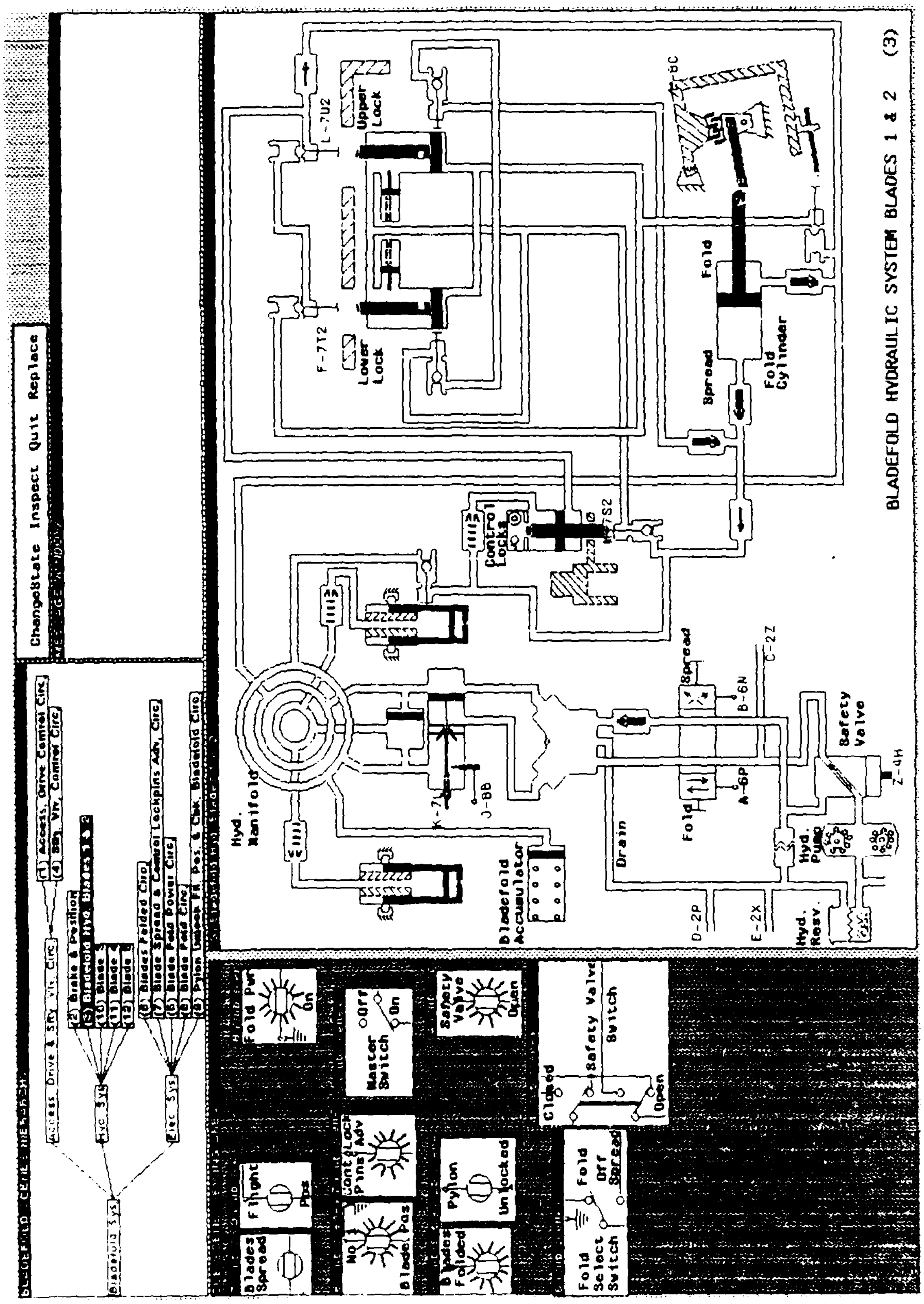


Figure 11. IMTS Screen

4. *The scratch-pad viewing area*, along the lower left side of the screen. This window is used to display copies of some of the system elements which appear in the detailed diagrams. The copies are identical to the originals in all respects, i.e., they change exactly as their originals do, in response to actions by the student, and they can be manipulated exactly as are the versions which are displayed in context. The only difference is that the copies in the scratch-pad area remain available for examination or manipulation after their host scenes are overlaid with other scenes.

Student Actions

The student uses the Xerox mouse to directly manipulate simulated objects and to select menu commands and other options that provide specific items of instruction or information. The simulation is always in an operate mode, i.e., the student may change switch settings and attach simulated test equipment at any time during a problem. In addition, the student may request 1) viewing a 'photograph' of an object (the bit-mapped graphic image), or 2) replacing the suspected component with a known good spare. Upon the completion of a practice problem, the student also has the option of inserting any malfunction of interest into the simulation, allowing an exploration of effects resulting from the known fault.

Switches are set by positioning the screen cursor on the desired switch setting and clicking the mouse. Test points are measured by selecting one of the displayed test equipments, 'attaching' it to the test point by clicking on the point of interest, and observing the test equipment object.

The current states of switches and controls are always displayed. In most cases the states of all indicators on the screen are displayed as the student operates the equipment. In this situation the IMTS cannot know exactly what the student is observing since the screen may display many indicators. In some instructional situations the IMTS may require the student to identify each indicator checked before displaying its reading, in order to track the student's actions precisely.

The IMTS also has the option of either displaying the states of all other objects (other than controls and indicators) or masking that information. When demonstrating procedures or debriefing the student about the previous problem, the IMTS will display the current internal states of all objects as the student actions are processed by the simulator. In most cases this display mode will reveal far more to the student than would the real equipment, a generally desirable result during explanations and debriefings.

During practice troubleshooting, however, displaying the internal states of objects which cannot normally be observed in the real world can destroy the diagnostic practice experience. Consequently the IMTS will mask such information unless and until the student specifically performs actions which would reveal the state of the object in the real world. As an example, a technician cannot be certain if a particular actuator is extended or retracted unless that part is directly observed. Thus observing the part represents a test. If the student requests seeing the part, the IMTS will display the graphic details and record the performance of that test.

Explanations of Expert Decisions

The Profile model is capable of generating a considerable amount of rationalization for its decisions. For example, Profile could generate a statement of the form

'The best test now is to check the BLADES SPREAD LIGHT. If it is ON then the fault cannot be one of <...>, if it is OFF then the fault must be in <...>.'

A human expert, however, might prefer to provide a rationalization which reflects and conveys a deeper understanding of the system. An example might be

'The best test now is to check the BLADES SPREAD LIGHT. If it is ON then we know that all the <...> functions are operating, whereas an OFF indication indicates that power from the blade interlock system is not getting to the override circuit.'

Automatically generating this type of explanation appears to be feasible, in light of the availability of the underlying system model, however more development effort will be required to do so. For the near future, human experts will add amplified rationalizations to the Profile-generated diagnostic strategies for delivery during the directed instructional mode.

The Tandem Configuration

In the first application of IMTS the trainer will be physically coupled to a relatively high fidelity two-dimensional simulator which presents videodisc images of Bladefold front panels and test points in response to student touch inputs. This simulator, called the Generalized Maintenance Training System (GMTS), is a descendent of a generalized simulation system developed at Behavioral Technology Laboratories in the late 1970's (Towne and Munro, 1981). GMTS accesses a data base representing Bladefold 'surface behavior' to determine and display the response of the Bladefold system to student actions.

The tandem configuration will provide highly realistic color images of the Bladefold system on the GMTS screen, and very flexible and individualized graphics and text on the IMTS screen. Because the GMTS input medium is touch panel, a virtually undetectable membrane placed over the display screen, the IMTS screen will also be configured to respond to touch inputs. In this two-screen configuration, the GMTS display will represent the real equipment upon which the student will perform testing actions, and the IMTS screen will represent the surrogate instructor supplying assistance and deeper explanation as required.

The GMTS system includes an Intel 8086 microcomputer development system equipped with a special-purpose video graphics overlay interface, a videodisc player, and a large-screen RGB color monitor. The Intel development system is driven by GMTS software programmed in Pascal. This software includes a generic simulation driver that accesses a database of Bladefold-specific surface behavior information to determine which videodisc still image should be shown in response to the student's actions and which (if any) graphics should be overlaid.

The two systems' computers are connected through their RS232C interfaces. During training the GMTS system will provide IMTS with information about student interactions with the GMTS surface simulation. IMTS will select training problems for GMTS to present, and will coordinate the flow of coaching, tutorial, and instructional activities in the two systems. In addition to presenting the graphic simulation, the IMTS screen will present all administrative and instructional text.

The IMTS and GMTS trainers have different training foci, but can be used together to provide features that would not be available from either in isolation. GMTS is a system that emphasizes surface simulation fidelity. Its color video displays of actual equipment panels and other system components help to ensure that students will be able to transfer the material they learn to the actual equipment. Further, GMTS offers a conventional CAI introductory course which acquaints the student with the organization and function of Bladefold. The IMTS emphasizes recognizing and remediating student misconceptions about how the target equipment works and how to diagnose it. This provides a way for students to operate and observe a functional model of the system rather than the physical model of GMTS. Furthermore, IMTS provides an online tutoring capability which will promote progress and minimize the occasions in which the student encounters difficulties which require human intervention.

Collaborative Simulation Training

In two special cases the instruction is delivered by either the IMTS or GMTS relatively independently. When the student is working through the GMTS tutorial on Bladefold organization and operation, the IMTS is essentially inactive. When the student is interacting directly with the IMTS functional simulation, as when exploring the effects of some fault not simulated in GMTS, the GMTS is inactive.

When the student is actually working to solve a simulated malfunction, however, the two systems collaborate in a close manner. The student performs testing actions on the GMTS display and observes videodisc displayed responses of the simulated system on the GMTS screen. The role of the IMTS is to monitor the work of the student and to provide individualized guidance and tutoring when required.

In the *undirected* mode, the IMTS allows the student to work without interrupting with constant criticism of detected imperfections in the diagnostic strategy. By comparing each student-selected test with the test which an expert would select *under the same conditions*, the IMTS maintains an ongoing measure of student ability. To do this, the IMTS invokes Profile (Section VI), a model of expert diagnostic decision-making, which makes its choice of test as if it too had performed exactly those tests already completed by the student.

In this mode of instruction, IMTS interrupts the interactions between student and GMTS only if 1) the student is about to perform an action with very serious implications (such as establishing an undesirable system state which is difficult to correct), or 2) the student's overall performance on the problem has reached a point where tutoring and guidance is indicated. Thus the objective in this mode is to permit some freedom of choice and exploration by the student, as long as the time consequences are not extreme, and to avoid incessant criticism of each and every action. This approach will not deprive the student of detailed performance assessment, as a detailed review is provided in the debriefing phase which follows each problem.

In the *directed* mode of instruction, the IMTS *explicitly* reviews the student's objectives and plans, and it ensures that the resulting diagnostic performance matches that of an expert. Thus, in this mode, IMTS provides both general guidance in carrying out the diagnostic exercise as well as specific assistance in performing and interpreting each test.

As in the undirected mode, the student's tests are compared to an expert's choices at each step, however this mode of instruction always directs the student to perform optimally. Since this can be precomputed, via Profile, a troubleshooting tree is used as the reference to minimize compute delays.

A more sensitive monitoring and assistance mode will be implemented in the near future. In the present system, once the student has been determined to require direction, the directed mode is active until the end of the current troubleshooting problem. The student cannot deviate from an ideal troubleshooting sequence from the time that guidance begins. In the new mode, students who do not know how to proceed at some point in a troubleshooting session will be able to request a test recommendation. This recommendation will take into account the information that can be determined from the troubleshooting tests that the student has thus far accomplished, even when those tests did not follow strictly a pre-defined troubleshooting tree. Recommendations will be determined, not by consulting a tree, but rather by a Profile analysis of its fault-effects data in light of the tests already completed.

SECTION VI. REPRESENTING EXPERT DIAGNOSTIC BEHAVIORS

A key outcome of earlier research has been the development of a generic (device-independent) model of expert troubleshooting behavior which can be applied to a wide range of specific equipments (Towne, 1984, 1986). The model, termed Profile, generates a detailed sequence of testing actions required to isolate any fault of interest.

Profile is embedded in the IMTS to provide diagnostic expertise when required. Prior to using the IMTS for training, the IMTS simulator is run in a batch mode to determine what the effects of each possible failure are at each indicator and test point. This information is then employed by Profile during training.

When provided complete data about the internal design of a system, Profile's troubleshooting sequences are near-optimal, and appear very much like those of expert maintenance technicians. Studies (Towne, Johnson, & Corwin, 1982) comparing Profile performance to that of actual technicians have yielded insights into the ways in which poorer maintainers differ from experts. The studies showed that varying the precision of fault effect knowledge in the model produced variations in diagnostic performance very much like those observed in human technician samples, whereas varying the effectiveness of the troubleshooting *strategy* did not.

Operation

Profile is a form of expert system whose rules have been generalized and built into the model, rather than expressed as domain-specific data. The primary advantages of following the generic approach are 1) the cost and effort of capturing the necessary system-specific data is kept modest, 2) the quality of diagnostic prescriptions generated by Profile are not dependent upon an individual expert's skill, attention to detail, and recall abilities, and 3) the process can be used for training and for the generation of diagnostic approaches.

Currently, Profile requires that a subject matter expert enumerate a modest-sized set of potentially useful modes, or combinations of switch settings. Profile employs this set as its repertoire of testing modes as it computes testing sequences. The number of possible tests considered by Profile is normally a large multiple of the number of modes provided, for all indicators and test points offer potential information in each mode. In the first IMTS application, for example, each specified mode offers approximately 125 testing points Thus, the human content expert is not burdened with providing an immense selection of testing modes. If Profile is unable to resolve certain groups of malfunctions with the modes provided, it lists the faults which are confounded, allowing the subject matter expert to add further modes which can discriminate them. Automated identification of these modes could be implemented some time in the future, although the process which would efficiently generate useful modes could be a formidable research venture.

The Model's World-View

The general structure of the Profile model is a hierarchy of rules, expressed in Interlisp-D. The model performs three primary functions for each step of a corrective maintenance problem: 1) action selection, 2) symptom evaluation, and 3) replacement consideration. This cycle is repeated until the true failure is identified and resolved. The organization of the data and processes is shown in Figure 12. The Test Selector considers the time to perform alternative actions and the potential (expected) information available at each test to determine the best course of action. The Test Performer simply looks up the symptom information which the selected test yields (and adds on the test time to the expert's solution time). The Test Interpreter judges the normality of the test result and determines what failures could have produced (or allowed) such a symptom. Further, this function adjusts the suspicion levels of all the possible failures to reflect the new information gained.

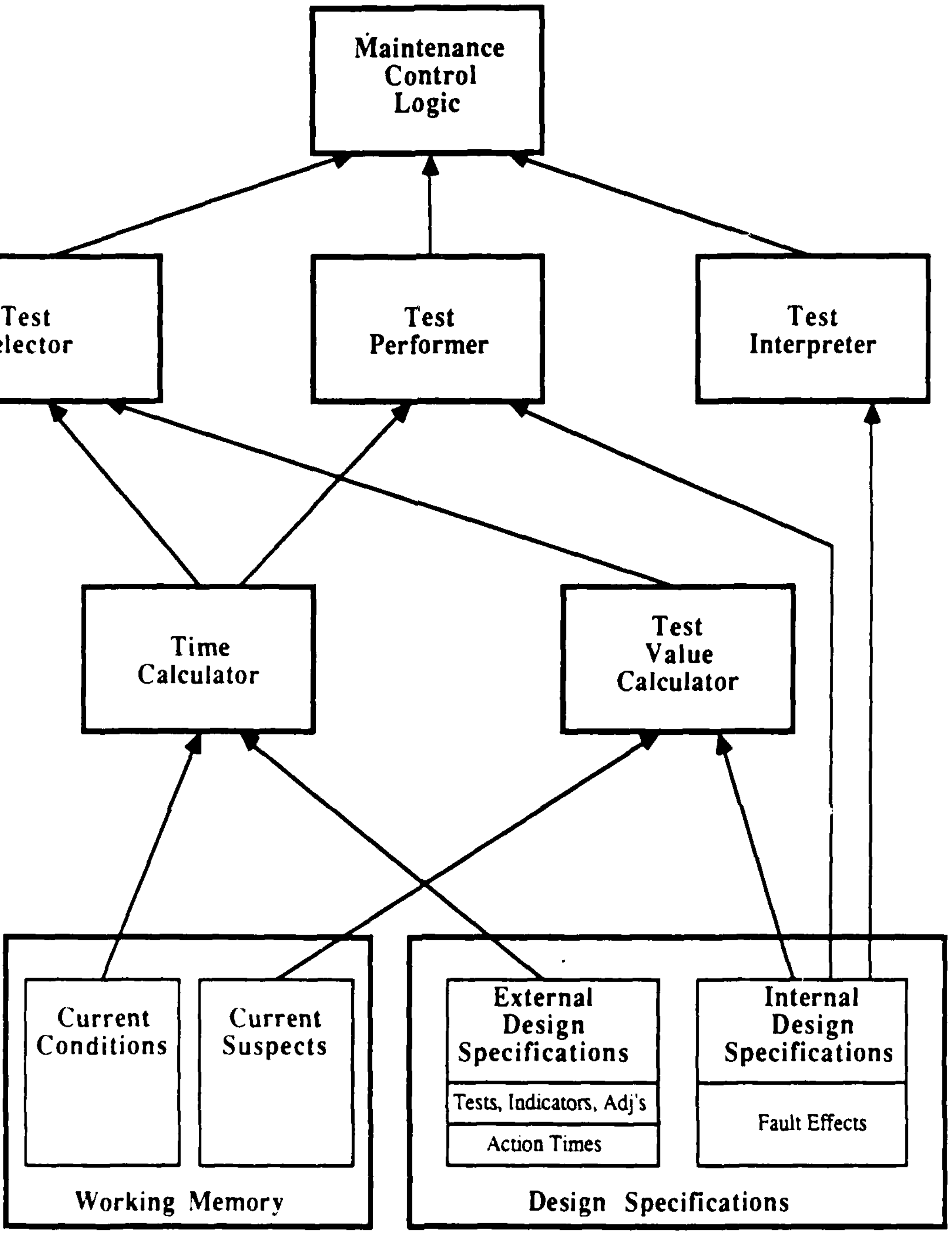


Figure 12. Profile System Organization.

When Profile starts a fault-isolation process, the initial suspicion level of each failure possibility is set according to the relative reliability of the generic object involved. If necessary, the failure rates can be manually adjusted to correspond more precisely to the particular target system and failure mode. The inherent failure likelihoods of components tend to influence Profile's early test selections, as Profile prefers tests which discriminate among the more highly suspected areas of the system.

Test Selection

The Test Selector evaluates each test in terms of the information value it offers (in relation to the current suspicion set), the time required to perform it, and the estimated time to complete fault isolation following performance of the test. This requires that the time required to complete fault isolation following each test under consideration be estimated (although a rigorous dynamic-programming solution was developed, the compute time requirements were enormous and the resulting testing sequences were only slightly more powerful than the heuristically determined strategies). Profile estimates the time to complete fault isolation following each test by determining the fault sets resulting from each possible symptom of the test and estimating the remaining fault isolation workload in terms of the replacement times of those components. This causes Profile to favor tests which discriminate effectively among components with high replacement times, and to favor shorter tests.

Undoubtedly, other concerns can affect a technician's approach to a maintenance problem in the field. Avoidance of danger, discomfort, excessive cognitive effort, or catastrophic error are almost certain to play major roles in maintenance of military systems. Such issues as these are often explicitly addressed in conventional expert system approaches to capturing diagnostic strategies. While such factors have not yet been included in Profile's rules for diagnostic decisions, the consideration of danger or discomfort would not be a difficult enhancement to incorporate. It would require a subjective evaluation of the negative characteristics of each of the major testing operations.

Profile includes a parameter which reflects the environment in which a particular maintenance task is to be analyzed. The parameter expresses the relationship between restoration time and cost of spares. A high setting of the parameter reflects an environment in which restoration time is paramount; consumption of spares is secondary to restoring the system as quickly as possible. A low setting reflects a depot environment in which additional testing is usually preferred to replacing expensive units which are not certain to be faulty. By varying this parameter one can explore the maintenance workloads under varying conditions.

Replacements and adjustments are regarded as *tests* in Profile, since these maintenance actions also have the property of providing new information. These actions, followed by a confirming test which previously produced an abnormal symptom, offer a small amount of new information (about one possible fault). As a result, Profile rarely selects these types of actions until it has exhausted more informative choices. Replacements are further penalized with the cost of the spare part being replaced, so that replacements are not often performed until there is high certainty that the failure has been identified. This rule is weakened, however, when time pressure is extreme. In this case expensive components and subassemblies may be replaced by Profile in its effort to minimize restoration time. In all cases, more expensive spares are less likely to be replaced than cheaper ones, all other factors being equal.

The simple expression for minimizing expected fault isolation time yields surprisingly diverse diagnostic behaviors under differing situations. In addition to driving the diagnostic model toward efficient performance, with which an expert would agree, and avoiding costly replacements, Profile exhibits these characteristics as well:

a. it generally performs front-panel checks prior to calling for test equipment usage, since the first use of test equipment involves a considerable set-up time cost. Once a particular test equipment has been used, Profile prefers its use to other equipments, since further testing is economical.

b. if 'known-good' spares are available for short-term substitution, it will use these if the time to swap them in and out is low, since the cost of using these spares is considered to be negligible.

c. it can 'profit' from past field experience, if component reliabilities are maintained to reflect their true values. All other factors being equal, Profile will pursue the testing of more failure-prone areas of a system.

Because there is uncertainty about what symptom will actually be obtained when a test is performed, the model will at times select tests which turn out to provide almost no new information (even though they had the *potential* of providing much new information), and it may at times replace units which are not the actual faulty unit. When this is done, however, it can be shown that the test or replacement selected was the most productive course of action to take, considering the time cost of alternative actions.

Test Interpretation

While the Test Selector is concerned with considering all possible symptoms of each possible test under consideration, the Test Interpreter function in Profile is concerned with drawing inferences from the symptom information actually obtained from the selected test (as provided by the Test Performer).

The Test Interpreter maintains a cumulative score for each failure possibility, reflecting the extent to which the pattern of received symptoms matches the possible symptoms the failure might produce. A high distance score for a failure indicates high mismatch between the symptoms received and those which would be produced by the failure, i.e., there is little likelihood that the failure in question has produced the symptoms received. These scores are initially set according to the component failure rates. This initialization affects the likelihood of replacing each component, by the Profile model, and it causes early test selection to focus on tests which relate to more failure-prone areas of the system.

The current version of Profile does not recognize that some tests are more error-prone than others, and are therefore less attractive than tests performed with higher certainty. Future versions of Profile may weight the symptom information according to the probability that the test can be performed and interpreted correctly. In this fashion the results of error-prone tests would be less significant than those for more easily performed tests.

Replacement Consideration

A replacement is selected by the Test Selector as the next action under two possible conditions:

a . a replacement, followed by a confirming check, offers the greatest information value per unit time, compared to all the other possible actions.

b. the received symptoms strongly implicate a particular fault.

Under all but the most urgent of conditions, a replacement decision by Profile will first trigger the performance of a special test, called the *most direct* test. The direct test, for each component, is that test which most clearly monitors the correct operation of the suspected component, and no others. While this is a very poor test to perform early in a diagnostic sequence, its performance prior to a replacement will minimize the chance of replacing a component which is not actually faulty.

SECTION VII. CONCLUSIONS

The IMTS project is an attempt to build a useful tool for simulation-based maintenance training, undertaken despite the many obvious and serious gaps in our knowledge about the nature of human learning and human instruction expertise. It could be argued that it would be better to wait until a more complete understanding of these difficult issues is at hand. We believe that while more research is needed to explore basic issues about human understanding, the IMTS project demonstrates that important strides can also be made by attempting to produce a functioning system now, relying largely on techniques we and our colleagues have developed in recent research efforts.

The IMTS is not a conventional experimental test of a single, well-controlled instructional variable, and it is difficult to attribute each IMTS feature to a particular psychological theory or finding. A significant portion of the IMTS is necessarily concerned with matters which have very little to do with instructional strategy, and therefore do not have psychological principles at their root. For example, the functions which support the simulation of the target system (in response to student actions and to malfunction conditions) and the functions which compute expert diagnostic actions have virtually nothing to do with instructional issues, yet they constitute a major portion of the IMTS software. Certainly the *form* of the simulation is a critical instructional issue, but the IMTS imposes almost no constraints on the form of the graphic simulation. The IMTS does not rely upon research in visual imagery, and few constraints on the form of visual imagery are imposed by the system. We see the IMTS as an environment for studying the learning effectiveness of alternate graphic forms and other issues in simulation training research.

Lessons Learned

Many of the lessons emerging from this work are specific to the peculiarities of simulation authoring and of programming in Lisp. Nonetheless, there are some lessons that may prove of benefit to others working in the area of computer based simulation training.

Powerful simulation software requires powerful computers. This research project has been guided by two not necessarily compatible goals — to advance the state of the art in simulation training systems and to provide practical computer based training, first for a helicopter bladefold system. The research goals demanded some fundamental departures from our earlier approaches to simulation composition systems (Towne & Munro, 1981, 1984; Towne, Munro, Johnson, & Lahey, 1983; Towne, in press). With those earlier, surface-oriented systems, simulation authors had to manually pre-compute all the possible states of a system. In the new deep-simulation approach, this intellectual work is automated. This not only makes life easier for authors, but also increases the power and range of the simulations delivered to students. In addition, it makes it possible to automatically generate some instruction that would have had to be authored by hand in a surface simulation system.

The down side of these advances is that the computation load for on-the-fly generated simulations is very much greater than for look-up-the-stored-effects simulations. For the Xerox 1108/1186 class of machines, and the Bladefold application, the response time to update the simulation is now averaging approximately fifteen seconds. For simpler applications, involving less than five or six screens of graphics, the response time is under five seconds.

Even using good tools, describing object behaviors is not simple. One of the goals of this project was to make it possible for non-programmer device experts to build more powerful simulations with less effort. We believe that this goal has been attained, but some of our expectations for ease of authoring have been tempered by experience. The object behavior editor lets the author describe the behavior of an object in terms of the relationships among its ports — the electrical, mechanical, and hydraulic connections that it can have with other objects. Experience indicates that objects sometimes cause effects based on their behavior descriptions that were not expected by the author. As a consequence, there is more cyclic describe/debug activity than was initially expected.

Although the task of object behavior analysis in isolation has proven to be more involved than expected, several new authoring tools make the describe/debug cycle easier. The system editor contains a "breadboard" mode that makes it possible to connect several objects experimentally and observe their interactive behaviors. This makes it easy to build a temporary local context for a new object in order to test its behavior in a more comprehensible environment than a full simulation like Bladefold. Further we are learning to construct objects which are useful only as artificial input devices. For example defining a variable-voltage power supply allows an author to supply the required inputs to all electrical ports in a scene, and to then observe the behavior of the subsystem.

In the long run, three factors should limit the problems of authoring object behaviors. First, authors can now experiment with behavior effects during the authoring process, making it easier to arrive at correct behavior descriptions, and to learn how to avoid possible problems. Second, as more systems are simulated, the library of reusable object definitions will grow. Eventually, many simulations could be created simply by putting together scenes composed entirely of existing object types. Third, it may be possible to extract object behavior definitions for new objects from existing sources, such as CAD/CAM libraries. This feature would integrate the equipment design and training process.

There is no perfect single level for specifying effects. In our earlier simulation authoring systems simulation effects were authored as though all were entirely global. If turning a switch to "standby" in one module could, under certain circumstances, cause a "ready" light to come on in a remote module, this effect would be authored directly, without referring to the propagation of effect from the switch through the relevant circuits to the light. If there were intervening objects that changed state when the switch was thrown, then their behavior would also have to be globally specified. This approach had the advantage that simple remote effects could be directly authored. At simulation time, such effects were quickly computed. It had the disadvantage that it was extremely difficult to build accurate simulations for complex devices, and that such simulations were even more difficult to maintain. Furthermore, there was no potential for reusing portions of such simulations, since effects were all intertwined at a global level.

In IMTS, simulation effects are authored as though they are entirely local. Remote effects are derived at run-time through a sequence of behavior computations and value propagations. This has the advantage that authors can take a modular approach to simulation construction, describing each element in isolation. Once the correct component behaviors have been authored, simulations can be easily maintained and modified. Even more importantly from an instructional viewpoint, the local effects approach offers the opportunity to generate intelligent instructional interactions using the model of the equipment embodied in the simulation data. The disadvantages of this approach are that 1) it places significant burdens on the generic simulation management software, 2) it is likely to result in slower simulations than a global approach, 3) it may enforce a less natural approach to authoring when the builder of a simulation has a surface understanding rather than a deep understanding of the system, and 4) it may preclude the simulation of some systems which could be accomplished with a global approach.

Future Directions

Extended Authoring Approaches

We intend to explore the possibility of adding a layer of non-local effects to an IMTS simulation and the consequences of such a hybrid approach on the authoring difficulty, range of possible simulation, and speed in computing the simulation. One direction for this work would be to incorporate into the IMTS model a provision for authoring composite objects. These would be modules made up of more primitive objects grouped in a particular topology. Certain of the value computations for the primitive objects in such a composite would be specified not at the component behavior level, but rather at the composite behavior level. Many substantive issues in design and implementation must be resolved before the feasibility of this approach can be evaluated.

One goal for future work is to reduce the dependence of simulation training on the skill of simulation/tutorial authors and to develop means for producing simulations for incompletely-specified systems. Ultimately, we would like to see authors simply draw a correct representation of a complex equipment system, and let an intelligent training system generate simulations and instructional materials from those drawings. This goal is not as remote as one might expect, if it is understood that the drawing primitives are previously defined objects with complex behaviors and other attributes that can be referred to in simulation and training. This goal can be approached in a modular fashion, and we have identified a number of enhancements to be explored.

Test Repertoire Generation

At present, an author must identify the tests that may be of interest for troubleshooting before running the Profile generic troubleshooting expert to generate symptom malfunction data using the simulation. If an intelligent process could deduce which tests are likely to be fruitful, this step could be automated. The process would have to use its understanding of the propagation of effects in a simulated system to determine where the effects of a malfunction could appear. If the process is extremely accurate, it could be used to reduce the volume of data that must be computed and stored, directing the simulation to compute test results only for those cases in which they are likely to depart from normals.

Additional Presentation Modes and Features

A number of technologically unchallenging enhancements would improve the instructional appeal of IMTS. Although they do not present significant research hurdles, these improvements could be expected to have significant training benefits. *Graphically simulated motion*, such as hydraulic flow in pipes, would improve understanding of simulated effects, especially for novice students. This is not a viable option using the present IMTS delivery system, due to the computational limitations of current AI machines. *Videodisc presentations* under IMTS control would enhance the viability of IMTS as a stand-alone training system in real-world training environments. Improved 'view' options could be delivered with videodisc, and motion sequences could be presented to demonstrate difficult maintenance procedures. *Voice output* technology could be used to supplement or eliminate the (already minimal) reading requirements that the system imposes on students.

Additional Applications

To test the generality of IMTS, a second equipment system will be simulated. The new system will have a complexity comparable to the Bladefold equipment that served as the first training application, but will be very different in nature. In order to test the range of applicability of IMTS, the new application will provide training for completely dissimilar equipment, such as a radar system. In addition, a number of enhancements to the IMTS are scheduled.

Providing Practice in Diagnosing Multiple Failures and Cascading Failures

In a deep simulation system such as IMTS, simulation builders should be able to simulate multiple failures and cascading failures with virtually no additional authoring cost. No additional simulation behavior data need to be entered in order to simulate multiple faults. A failed object is simulated when IMTS replaces the normal behavior rules of the object with special failure mode behavior rules. It can simulate the failure of multiple elements by loading the failure mode rules for all the failed objects in place of the normal behavior rules.

To implement cascading failures, the object author will specify the triggering conditions that cause the object to fail. For example, pressure of greater than a threshold amount may cause a reducing valve to blow out and to begin behaving like a pipe. In an actual simulation, the introduction of a certain failure in a hydraulic component connected to a reducing valve may cause the pressure to exceed that valve's threshold and thereby induce the blow out failure. Adding the failure-triggering conditions to an object's behavior description ensures that the object will exhibit appropriate cascading failure behavior.

Guided Simulation

A new instructional mode, called guided simulation, will allow experts to perform operations on the simulation and to enter explanations of their procedures and the system's responses. Students can then observe replays of the process at their own pace, and they can participate in the decision-making which produced the process. This mode will be useful for providing procedure drills and for giving interactive demonstrations of troubleshooting approaches. In guided simulation, a student reads a series of brief instructions in the text window on the screen, and carries out those instructions by manipulating simulated controls, by using simulated test equipment, and by replacing components in the IMTS scene window.

Specific guided simulations can be easily created by an instructor who is familiar with the proper procedures for the actual target equipment. The instructor creates the student prompt texts using a simple editing window. To specify the student actions required after each such prompt, the instructor merely performs the action in the scene window. Guided simulation authoring is a very straightforward way to generate non-free-play simulation lessons. Once a simulation has been constructed for general IMTS use, guided simulation lessons can be developed unusually quickly — perhaps with as little as two or three hours of development time for each hour of instruction.

Views for Instructional Purposes

The simulation scenes constructed for IMTS must be complete since they are executed almost as if they were physical constructs. As such, the simulation scenes may become complex, and they may be poor representations for novice students, who would benefit from the initial presentation of simplified drawings. Other students might benefit from being able to simultaneously view several active objects from different scenes, so that they could see how the behavior of one affects the other. Thus there is a strong need for the ability to display portions of the system in ways designed to ease *understanding* as opposed to driving the simulation. We call such simplified presentations *pedagogical views*.

In order to support the pedagogical view approach, we have developed the capability to make "active snapshots" of portions of scenes. A rectangular portion of any scene can be copied from its scene into other views, such as the scratch-pad view area described above.

The objects in these snapshots retain all the characteristics of objects in the full scene window. They change in response to simulation events, and control objects can be directly manipulated with the mouse.

A composition tool must now be developed for the use of instructors and course designers to extend this concept. This tool will permit a course developer to build special scenes consisting primarily of snapshots of portions of existing scenes and to create images of subsystems which are physically realistic rather than schematically accurate. It will also be possible to add background graphical elements and text to these scenes. Different explanatory sequences can be authored with different pedagogical views, so that each instructional interaction can make use of the most appropriate visual presentation. Pedagogical views will improve the presentation of simulation and instruction to students with different levels of understanding.

Summary

Despite a number of problems related to computational limitations, the feasibility of direct manipulation authoring of simulations and the object-oriented approach to specifying behavior has been demonstrated in the IMTS. Moreover the IMTS demonstrates that intelligent maintenance training interactions can be generated automatically by executing functions which operate upon a specific system representation in a generic manner. The IMTS is intended to address immediate training requirements and to offer an attractive environment for continuing research in learning and instruction. A number of challenges remain, and there is a great potential for further exploiting the intelligence embodied in this training system.

REFERENCES

Davis, R. Diagnostic reasoning based on structure and behavior, *Artificial Intelligence*, 1984, 24, 347-410.

De Kleer, J. How circuits work. , *Artificial Intelligence*, 1984, **24**, 205-280.

De Kleer, J. & Brown, J. S. A qualitative physics based on confluences, *Artificial Intelligence*, 1984, **24**, 7-84.

Forbus, K. D. Qualitative process Theory, *Artificial Intelligence*, 1984, **24**, 85-168.

Govindaraj, T. Intelligent computer aids for fault diagnosis training of operators of large dynamic systems. In J. Psotka, L. D. Massey & S. A. Mutter (Eds.) *Intelligent tutoring systems: Lessons learned.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc., in press.

Hollan, J. D. 1983. STEAMER: An Overview with Implications for AI Applications in Other Domains. Presented at the Joint Services Workshop on Artificial Intelligence in Maintenance, Institute of Cognitive Science, Boulder, CO: October 4-6, 1983.

Hollan, J. D., Hutchins, E. L., and Weitzman, L. 1984. STEAMER: An Interactive Inspectable Simulation-based Training System, *The AI Magazine*, 1984, 2.

Johnson, M .C., Munro, A., and Towne, D. M. Evaluation of the BTL Trainer in a Navy school environment. (Technical Report No. 94) Los Angeles: University of Southern California, Behavioral Technology Laboratories, 1981.

Norman, D. A. & Draper S. W. (Eds.) *User-centered system design: New perspectives on human-computer interaction..* Hillsdale, NJ: Lawrence Erlbaum Associates, 1986.

Towne, D. M. A generalized model of fault-isolation performance. *Proceedings, Artificial Intelligence in Maintenance: Joint Services Workshop*, 1984.

Towne, D. M. A generic expert diagnostician. In *The Proceedings of the Air Force Workshop on Artificial Intelligence Applications for Integrated Diagnostics*, 1986.

Towne, D. M. The generalized maintenance trainer: Evolution and revolution. In W. B. Rouse (Ed.), *Advances in man-machine systems research, Vol 3,* JAI Press, in press.

Towne, D. M., Fehling, M. R., & Bond, N. A. 1981. Design for the Maintainer: Projecting maintenance performance from design characteristics. Los Angeles, CA: Behavioral Technology Laboratories, University of Southern California, Report No. TR-95.

Towne, D. M., Johnson, M. C. and Corwin, W. H. 1982. A Technique for Projecting Maintenance Performance from Design Characteristics. Los Angeles, CA: Behavioral Technology Laboratories, University of Southern California, Report No. TR-100.

Towne, D. M., Johnson, M. C. and Corwin, W. H. 1983. A Performance-based Technique for Assessing Equipment Maintainability. Los Angeles, CA: Behavioral Technology Laboratories, University of Southern California, Report No. TR-102.

Towne, D. M. & Munro, A. Generalized maintenance trainer simulator: Development of hardware and software. (Technical Report No. 81-9) San Diego: Navy Personnel Research and Development Center, 1981.

Towne, D. M. & Munro, A. Preliminary design of the advanced ESAS System. Technical Report No. 105, Los Angeles: Behavioral Technology Laboratories, University of Southern California, December 1984.

Towne, D. M., Munro, A., Johnson, M. C., and Lahey, G. F. Generalized Maintenance Trainer Simulator: Test and Evaluation in the Laboratory Environment. (NPRDC TR 83-28) San Diego: Navy Personnel Research and Development Center, August 1983.

Towne, D. M., Munro, A., Pizzini, Q. A., & Surmon, D. S. Development of intelligent maintentance training technology: Design study. Technical Report No. 106, Los Angeles: Behavioral Technology Laboratories, University of Southern California, May 1985.

Towne, D. M., Munro, A., Pizzini, Q. A., & Surmon, D. S. Intelligent tutoring technology. Technical Report No. 110, Los Angeles: Behavioral Technology Laboratories, University of Southern California, in preparation.

Towne, D. M., Surmon, D. S., Pizzini, Q. A., Penrose, L. & Munro, A. *IMTS: Intelligent Maintenance Training System User's Guide*. Los Angeles: Behavioral Technology Laboratories, University of Southern California, 1987.

Dr. Beth Adelson
Department of Computer Science
Tufts University
Medford, MA 02155

Personnel Analysis Division,
AF/MPXA
5C360, The Pentagon
Washington, DC 20330

Air Force Human Resources Lab
AFHRL/MPD
Brooks AFB, TX 78235

Human Factors Laboratory
Naval Training Systems Center
Orlando, FL 32813

Dr. Ed Aiken
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. John Allen
Department of Psychology
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. John R. Anderson
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Steve Andriole
George Mason University
School of Information
Technology & Engineering
4400 University Drive
Fairfax, VA 22030

Technical Director, ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Patricia Baggett
University of Colorado
Department of Psychology
Box 345
Boulder, CO 80309

Dr. Eva L. Baker
UCLA Center for the Study
of Evaluation
145 Moore Hall
University of California

Dr. Gautam Biswas
Department of Computer
Sciences
University of South Carolina
Columbia, SC 29208

Dr. John Black
Teachers College
Columbia University
525 West 121st Street
New York, NY 10027

Dr. Deborah A. Boehm-Davis
Department of Psychology
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Jeff Bonar
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Robert Breaux
Code N-095R
Naval Training Systems Center
Orlando, FL 32813

Mr. Raymond E. Christal
AFHRL/MOE
Brooks AFB, TX 78235

Distribution List (BTL/Towne) NR-535-001

Dr. John S. Brown
XEROX Palo Alto Research
Center
3333 Coyote Road
Palo Alto, CA 94304

Dr. Bruce Buchanan
Computer Science Department
Stanford University
Stanford, CA 94305

Maj. Hugh Burns
AFHRL/IDE
Lowry AFB, CO 80230-5000

Dr. Patricia A. Butler
OERI
555 New Jersey Ave., NW
Washington, DC 20208

Joanne Capper
Center for Research into Practice
1718 Connecticut Ave., N.W.
Washington, DC 20009

Dr. Pat Carpenter
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. John M. Carroll
IBM Watson Research Center
User Interface Institute
P.O. Box 218
Yorktown Heights, NY 10598

Dr. Michelene Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. L. J. Chmura
Computer Science and Systems
Code: 7590
Information Technology Division
Naval Research Laboratory
Washington, DC 20375

Dr. Stephanie Doan
Code 6021
Naval Air Development Center
Warminster, PA 18974-5000

Professor Chu Tien-Chen
Mathematics Department
National Taiwan Universi
Taipei, TAIWAN

Dr. Yee-Yeen Chu
Perceptronics, Inc.
21111 Erwin Street
Woodland Hills, CA 9136

Dr. William Clancey
Stanford University
Knowledge Systems Laborat
701 Welch Road, Bldg. C
Palo Alto, CA 94304

Dr. Allan M. Collins
Bolt Beranek & Newman, In
50 Moulton Street
Cambridge, MA 02138

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 N. Quincy Street
Arlington, VA 22217-5000

Dr. Albert T. Corbett
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Robert B. Davis
Curriculum Laboratory
(Education)
University of Illinois
Urbana, IL 61801

Dr. Denise Dellarosa
Dept. of Psychology
University of Colorado
Boulder, CO 80309

Dr. Andrea di Sessa
University of California
School of Education
Tolman Hall
Berkeley, CA 94720

Dr. Dedre Gentner
University of Illinois
Department of Psychology
603 E. Daniel St.
Champaign, IL 61820

Distribution List (BTL/Towne) NR-535-001

Defense Technical
Information Center
Cameron Station, Bldg 5
Alexandria, VA 22314
Attn: TC
(12 Copies)

ERIC Facility-Acquisitions
4833 Rugby Avenue
Bethesda, MD 20014

Dr. Marshall J. Farr
2520 North Vernon Street
Arlington, VA 22207

Dr. Gerhard Fischer
University of Colorado
Department of Computer Science
Boulder, CO 80309

Dr. Kenneth D. Forbus
University of Illinois
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL 61801

Dr. Barbara A. Fox
University of Colorado
Department of Linguistics
Boulder, CO 80309

Dr. John R. Frederiksen
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

Dr. Michael Friendly
Psychology Department
York University
Toronto ONT
CANADA M3J 1P3

Julie A. Gadsden
Information Technology
Applications Division
Admiralty Research Establishment
Portsdown, Portsmouth PO6 4AA
UNITED KINGDOM

Mr. William Hartung
PEAM Product Manager
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Robert Glaser
Learning Research
& Development Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Sam Glucksberg
Department of Psychology
Princeton University
Princeton, NJ 08540

Dr. Sherrie Gott
AFHRL/MODJ
Brooks AFB, TX 78235

Dr. T. Govindaraj
Georgia Institute of Techny
School of Industrial & Syss
Engineering
Atlanta, GA 30332

Jordan Grafman, Ph.D.
2021 Lyttonsville Road
Silver Spring, MD 20910

Dr. James G. Greeno
University of California
Berkeley, CA 94720

Dr. Dick Gregory
Behavioral Sciences Division
Admiralty Research
Establishment
Teddington
Middlesex, ENGLAND

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Dr. Bruce Hamill
The Johns Hopkins Universiy
Applied Physics Laboratory
Laurel, MD 20707

Dr. R. J. K. Jacob
Computer Science and Systems
Code: 7590
Information Technology Divn
Naval Research Laboratory
Washington, DC 20375

Dr. Barbara Hayes-Roth
Department of Computer Science
Stanford University
Stanford, CA 95305

Dr. Joan I. Heller
505 Haddon Road
Oakland, CA 94606

Dr. Jim Hollan
Intelligent Systems Group
Institute for
Cognitive Science (C-015)
UCSD
La Jolla, CA 92093

Dr. Keith Holyoak
University of Michigan
Human Performance Center
330 Packard Road
Ann Arbor, MI 48109

Ms. Julia S. Hough
Lawrence Erlbaum Associates
6012 Greene Street
Philadelphia, PA 19144

Dr. James Howard
Dept. of Psychology
Human Performance Laboratory
Catholic University of
America
Washington, DC 20064

Dr. Ed Hutchins
Intelligent Systems Group
Institute for
Cognitive Science (C-015)
UCSD
La Jolla, CA 92093

Dr. Janet Jackson
Rijksuniversiteit Groningen
Biologisch Centrum, Vleugel D
Kerklaan 30, 9751 NN Haren (Gn.)
NETHERLANDS

Dr. Benjamin Kuipers
University of Texas at Austin
Department of Computer Sciences
T.S. Painter Hall 3.28
Austin, Texas 78712

Dr. Claude Janvier
Directeur, CIRADE
Universite' du Quebec a Mol
P.O. Box 8888, St. "A"
Montreal, Quebec H3C 3P8
CANADA

Dr. Robin Jeffries
Hewlett-Packard Laboratories
P.O. Box 10490
Palo Alto, CA 94303-091

Dr. Marcel Just
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Daniel Kahneman
Department of Psychology
University of California
Berkeley, CA 94720

Dr. Milton S. Katz
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Wendy Kellogg
IBM T. J. Watson Research .
P.O. Box 218
Yorktown Heights, NY 1058

Dr. David Kieras
University of Michigan
Technical Communication
College of Engineering
1223 E. Engineering Buildig
Ann Arbor, MI 48109

Dr. Kenneth Kotovsky
Department of Psychology
Community College of
Allegheny County
800 Allegheny Avenue
Pittsburgh, PA 15233

Dr. Jane Malin
Mail Code SR 111
NASA Johnson Space Center
Houston, TX 77058

Distribution List (BTL/Towne) NR-535-001

Dr. Jill Larkin
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Robert Lawler
Information Sciences, FRL
GTE Laboratories, Inc.
40 Sylvan Road
Waltham, MA 02254

Dr. Alan M. Lesgold
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Jim Levin
Department of
Educational Psychology
210 Education Building
1310 South Sixth Street
Champaign, IL 61820-6990

Dr. John Levine
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Clayton Lewis
University of Colorado
Department of Computer Sciences
Campus Box 430
Boulder, CO 80309

Library
Naval War College
Newport, RI 02940

Library
Naval Training Systems Center
Orlando, FL 32813

Vern Malec
NPRDC, Code P-306
San Diego, CA 92152-6800

Dr. Randy Mumaw
Program Manager
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. William L. Maloy
Chief of Naval Education
and Training
Naval Air Station
Pensacola, FL 32508

Dr. Elaine Marsh
Naval Research Laboratory
Code 7510
4555 Overlook Avenue, Soutt
Washington, DC 20375-5000

Dr. Manton M. Matthews
Dept of Computer Sciences
University of South Carolia
Columbia, SC 29208

Dr. James McMichael
Assistant for MPT Research,
Development, and Studies
OP 01B7
Washington, DC 20370

Dr. Barbara Means
Human Resources
Research Organization
1100 South Washington
Alexandria, VA 22314

Dr. James R. Miller
MCC
9430 Research Blvd.
Echelon Building #1, Suite 1
Austin, TX 78759

Dr. William Montague
NPRDC Code 13
San Diego, CA 92152-6800

Dr. Nancy Morris
Search Technology, Inc.
5550-A Peachtree Parkway
Technology Park/Summit
Norcross, GA 30092

Dr. Harold F. O'Neil, Jr.
School of Education - WPH 1
Department of Educational
Psychology & Technology
University of Southern Cal
Los Angeles, CA 90089-001

Dr. Allen Munro
Behavioral Technology
Laboratories - USC
1845 S. Elena Ave., 4th Floor
Redondo Beach, CA 90277

Spec. Asst. for Research, Experimental & Academic Programs,
NTTC (Code 016)
NAS Memphis (75)
Millington, TN 38054

Dr. Donald A. Norman
Institute for Cognitive Science
University of California
La Jolla, CA 92093

Deputy Technical Director
NPRDC Code 01A
San Diego, CA 92152-6800

Director, Training Laboratory,
NPRDC (Code 05)
San Diego, CA 92152-6800

Director, Manpower and Personnel
Laboratory,
NPRDC (Code 06)
San Diego, CA 92152-6800

Director, Human Factors
& Organizational Systems Lab,
NPRDC (Code 07)
San Diego, CA 92152-6800

Fleet Support Office,
NPRDC (Code 301)
San Diego, CA 92152-6800

Library, NPRDC
Code P201L
San Diego, CA 92152-6800

Dr. Michael Oberlin
Naval Training Systems Cntr
Code 711
Orlando, FL 32813-7100

Psychologist
Office of Naval Research
Branch Office, London
Box 39
FPO New York, NY 09510

Special Assistant for Marine
Corps Matters,
ONR Code 00MC
800 N. Quincy St.
Arlington, VA 22217-5000

Psychologist
Office of Naval Research
Liaison Office, Far East
APO San Francisco, CA 96503

Dr. Judith Orasanu
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Douglas Pearse
DCIEM
Box 2000
Downsview, Ontario
CANADA

Dr. James W. Pellegrino
University of California,
Santa Barbara
Department of Psychology
Santa Barbara, CA 93106

Dr. Tjeerd Plomp
Twente University of Techny
Department of Education
P.O. Box 217
7500 AE ENSCHEDE
THE NETHERLANDS

Dr. Martha Polson
Department of Psychology
Campus Box 346
University of Colorado
Boulder, CO 80309

Dr. Steven E. Poltrock
MCC,
Human Interface Program
3500 West Balcones Center Dr.
Austin, TX 78759

Dr. Harry E. Pople
University of Pittsburgh
Decision Systems Laboratory
1360 Scaife Hall
Pittsburgh, PA 15261

Dr. Joseph Psotka
ATTN: PERI-1C
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Paul S. Rau
Code U-32
Naval Surface Weapons Center
White Oak Laboratory
Silver Spring, MD 20903

Dr. James A. Reggia
University of Maryland
School of Medicine
Department of Neurology
22 South Greene Street
Baltimore, MD 21201

Dr Fred Reif
Physics Department
University of California
Berkeley, CA 94720

Dr. Brian Reiser
Department of Psychology
Green Hall
Princeton University
Princeton, NJ 08540

Dr. Gil Ricard
Mail Stop C04-14
Grumman Aerospace Corp.
Bethpage, NY 11714

Mark Richer
1041 Lake Street
San Francisco, CA 94118

Dr. Mary S. Riley
Program in Cognitive Science
Center for Human Information
Processing
University of California
La Jolla, CA 92093

Dr. Linda G. Roberts
Science, Education, and
Transportation Program
Office of Technology Assest.
Congress of the United States
Washington, DC 20510

Dr. William B. Rouse
Search Technology, Inc.
5550-A Peachtree Parkway
Technology Park/Summit
Norcross, GA 30092

Dr. Walter Schneider
Learning R&D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Miriam Schustack
Code 51
Navy Personnel R & D Center
San Diego, CA 92152-6800

Dr. Marc Sebrechts
Department of Psychology
Wesleyan University
Middletown, CT 06475

Dr. Judith Segal
OERI
555 New Jersey Ave., NW
Washington, DC 20208

Dr. Colleen M. Seifert
Intelligent Systems Group
Institute for
Cognitive Science (C-01)
UCSD
La Jolla, CA 92093

Distribution List (BTL/Towne) NR-535-001

Dr. Sylvia A. S. Shafto
Department of
Computer Science
Towson State University
Towson, MD 21204

Dr. Ben Shneiderman
Dept. of Computer Science
University of Maryland
College Park, MD 20742

Dr. Randall Shumaker
Naval Research Laboratory
Code 7510
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Dr. Edward E. Smith
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Richard Sorensen
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Kathryn T. Spoehr
Brown University
Department of Psychology
Providence, RI 02912

Dr. Kurt Steuck
AFHRL/MOD
Brooks AFB
San Antonio TX 78235

Dr. Albert Stevens
Bolt Beranek & Newman, Inc.
10 Moulton St
Cambridge, MA 02238

Dr. John Tangney
AFOSR/NL
Bolling AFB, DC 20332

Dr. Kikumi Tatsuoka
CERL
252 Engineering Research
Laboratory
Urbana, IL 61801

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1185 Coleman Avenue, Box 50
Santa Clara, CA 95052

Dr. Martin A. Tolcott
3001 Veazey Terr., N.W.
Apt. 1617
Washington, DC 20008

Dr. Douglas Towne
Behavioral Technology Labs
1845 S. Elena Ave.
Redondo Beach, CA 90277

Headquarters, U. S. Marines
Code MPI-20
Washington, DC 20380

Dr. Jerry Vogt
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Ralph Wachter
JHU-APL
Johns Hopkins Road
Laurel, MD 20707

Dr. Donald Weitzman
MITRE
1820 Dolley Madison Blvd.
MacLean, VA 22102

Dr. Keith T. Wescourt
FMC Corporation
Central Engineering Labs
1185 Coleman Ave., Box 58
Santa Clara, CA 95052

Dr. Barbara White
Bolt Beranek & Newman, In
10 Moulton Street
Cambridge, MA 02238

Dr. Christopher Wickens
Department of Psychology
University of Illinois
Champaign, IL 61820

Distribution List (BTL/Towne) NR-535-001

Dr. Heather Wild
Naval Air Development
Center
Code 6021
Warminster, PA 18974-5000

Dr. Michael Williams
IntelliCorp
1975 El Camino Real West
Mountain View, CA 94040-2216

Dr. Robert A. Wisher
U.S. Army Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Martin F. Wiskoff
Navy Personnel R & D Center
San Diego, CA 92152-6800

Dr. Dan Wolz
AFHRL/MOE
Brooks AFB, TX 78235

Dr. Wallace Wulfeck, III
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Joe Yasatuke
AFHRL/LRT
Lowry AFB, CO 80230

Dr. Masoud Yazdani
Dept. of Computer Science
University of Exeter
Exeter EX4 4QL
Devon, ENGLAND

Dr. Joseph L. Young
Memory & Cognitive
Processes
National Science Foundation
Washington, DC 20550

Lt. James Hooper
Naval Air Systems
Command
Code APC 205-ON
Department of the Navy
Washington, DC 20361-120

Ms. Vicky Medley
Naval Air Maintenance
Training Group
Naval Air Station (71)
Millington, TN 38054-502

Mr. Howard Quisenberry
Chief of Naval Technical
Training (Code N31)
Naval Air Station (75)
Millington, MD 38054-502

Dr. Nancy Perry
Chief of Naval Education
& Training
Naval Air Station
Pensacola, FL 32508