AI IN MAINTENANCE TRAINING -- SOME TANGIBLE RESULTS

David J. Sykes
L. C. Keskey P. E.
Honeywell Inc.
Training and Control Systems Division
1200 East San Bernardino Road
West Covina, CA 91790

ABSTRACT

This paper describes the implementation of an AI system that can perform the dual role of Job Performance Aid (JPA) and Intelligent Tutor (IT) for use in On-the-Job Training (OJT). It is well known that the best human experts possess a mental model of internal equipment operation and a good trainer will teach this conceptual knowledge as well as the usual diagnostic skills. The Intelligent Tutor portion is aimed at building this mental model through interaction with a simulation of the equipment. The student interface employs high resolution graphics and a mouse. The simulation is a qualitative causal model which is much simpler than a full mathematical model yet retains all the important distinctions between system states. The Job Performance Aid is an Expert System (ES) which is automatically derived from the qualitative simulation model. This is accomplished by using the model to predict the behavior of the equipment and the propagation of effects under all conceivable conditions. The ES rules are then induced from the fault symptom pattern produced by exercising the model. By taking this approach, the ES provides "deep reasoning" as opposed to the "shallow reasoning" often found in an ES based solely on externally observable features.

INTRODUCTION

In recent years Expert Systems have been successfully applied to a wide variety of maintenance problems. There are, however, several limitations of the current generation Expert Systems. First, they do little to improve the level of understanding of underlying principles by the user. Second, the domain of application is very limited and specific such that unusual situations are not easily handled. Third, most current Expert Systems are based on "shallow reasoning" in which observable symptoms are empirically associated with various end results. Consequently, in addition to providing an Expert System as a Job Performance Aid (JPA) it is necessary to improve the level of understanding on the part of the maintenance technician. It has been demonstrated that the real human experts have developed a mental model of the inner workings of the system being diagnosed. This enables them to diagnose problems they have never seen before (de Kleer and Brown, 1983). The mental model does not need to be at the same level of detail as a design engineer would have. All that is necessary is for the troubleshooter to possess a qualitative causal model which enables him to understand how the system elements interact with each other and how effects are propagated.

While diagnostic Expert Systems are very useful to the maintenance process, they do not reduce the need to develop true human experts. Ideally a cooperative problem solving environment should be provided in which both man and computer (ES) contribute according to their respective strengths (Thomas 1985). This goal can be accomplished by creating an OJT environment in which the function of the maintenance support system could vary as shown in Figure 1. Under crisis conditions it would act as an expert JPA, minimizing time to completion. Under conditions of light workload it would function as a off-line Intelligent Tutoring System (ITS) using advanced explanation facilities and tutoring techniques. Under normal conditions it would perform a mix of JPA and ITS which would help to improve the efficiency of OJT. For the maintenance domain, an expert OJT system could 1) improve the ability of the low end performers to the average level, 2) support the high end performers when they are under time pressure by forcing a rational approach, and 3) preserve eroding expertise in a usable form. The intention is not to replace man but to support him by integration of expert system knowledge with his own experience. For advanced JPA, this calls for a mixed initiative mode of operation in which either man or machine can take the lead depending on the particular circumstances. The basic architecture of a system intended for this dual role is illustrated in Figure 2.

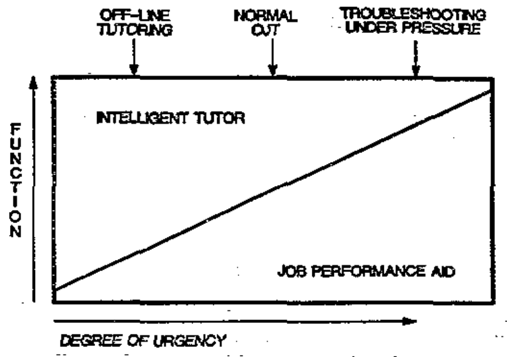

Figure 1. Varying Mix of Tutor and JPA

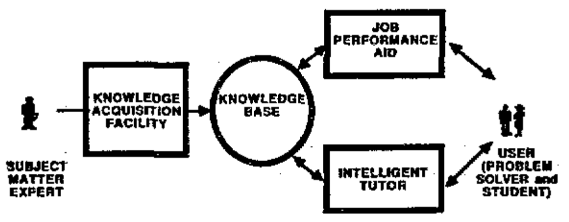

Figure 2. Combined JPA and Tutor

The central thrust of this architecture is the common knowledge base which at a high enough level of abstraction is a simple concept. However, the structure of this knowledge base from an implementation point of view creates a problem. The difficulty is that a different form of knowledge is necessary for instructional purposes than for diagnostic support (JPA) purposes. The first application requires theoretical knowledge of system operation under normal and malfunction conditions.

The second requires knowledge of diagnostic procedures and strategies. The solution to this dual need is a qualitative simulation model of the system in question. The qualitative model (with added tutorial material) can serve as an instructional tool and can also form the source of rules for the diagnostic Expert System.

In this paper the principles of qualitative simulation are described which are then applied to the construction of a specific model for a jet engine oil system. Next, the use of the qualitative model for instructional purposes is described. The procedure for deriving an Expert System from the qualitative model is discussed. Finally, a typical user scenario is illustrated.

QUALITATIVE SIMULATION

Qualitative Simulation is a relatively new technique (Kuipers, 1986) and is based on Qualitative Physics (QP). The goals of QP are to identify the distinctions and laws that qualitatively describe the behavior of physical devices without using quantitative methods. Specifically, these goals are:

1. To be far simpler than classical physics and yet retain all the important distinctions without invoking the mathematics of continuously varying quantities and differential equations.

2. To produce causal accounts of physical mechanism that are easy to understand.

3. To provide the foundations for deep reasoning models for the next generation of Expert Systems

4. To be executed efficiently thus providing a real-time capability.

The principles of QP can be applied to build qualitative models that are suitable abstractions of physical systems for maintenance purposes. The objective is to build a model that represents the causal behavior of a system based on the behavior of its component parts when interconnected in a particular manner. The propagation of effects both normal and abnormal can be studied and used for maintainability oriented problems in design, training, and troubleshooting.

Each state variable of Qualitative Simulation is represented by a small finite number of values known as landmark values. Landmark values can be numeric or symbolic such as low, medium, high, or present, absent or on, off, etc. In maintenance applications many landmark values are binary. The choice of landmark values depends on the criticality of the operational values of the actual variable. Rates of change of landmark variables are represented simply as +, 0, -, to mean increasing, steady and decreasing, respectively. A significant event as in the case of a change in a landmark value is represented by a distinguished time point t. A qualitative simulation has a finite number of distinguished time points. Any qualitative function f can therefore be described in terms of landmark values, derivations, and the finite set of timepoints.

The system can then be described by a series of qualitative functions and a number of constraints. A constraint is used to determine what happens when a landmark value is reached or a certain combination of landmark values of different functions occurs. In other words, the rules of behavior are expressed in terms of constraints. Using this approach, the possible behaviors of a system can be predicted from the initial conditions and the constraints. The behavioral description may then be used to explain a set of observations or the way a system operates.

Once a qualitative model has been developed, it can be used effectively for training maintenance technicians in the following topic areas, or levels of understanding and expertise.

- o Physical structure and connection
- o System functionality
- o Fault detection and diagnosis
- o Optimal troubleshooting strategies

In addition to forming the basis of an *instructional device*, the *qualitative model* can be used to derive a diagnostic ES by causal reasoning. The model thus serves as the common knowledge base in Figure 2.

Problem Definition

The size and complexity of this prototype was a *major concern in the early project planning phase*. It was decided that for a prototype system, a bounded, well-defined, problem of about 50 rules would be the optimum size. It was also decided that a more generic type of system such as an engine subsystem would be a more representative example. Therefore, a simplified version of an aircraft engine oil lubrication system was selected as the subject for this ES-Simulation prototype development project. It was named JEDI for Jet Engine Diagnostic.

The Simulation Model consists of nine components (i.e., objects). Included in the model are the oil tank, strainer, pump, filter, oil air cooler, cold start bypass valve, gear box, oil pressure transmitter, and the oil pressure gauge. This is a simplified version of the actual F-100 Jet Engine Oil Lubrication System. These objects are connected as shown in Figure 3.

Model Implementation

An object oriented programming approach was taken with each of the nine components of the oil system implemented as an object. The objects communicate with each other by exchanging attribute values. Each object is only aware of its immediate neighbors but the effects of a change in one object can be propagated around the network. In Figure 4 a digraph representation shows the exchange of attribute values between adjacent nodes. Note that two extra components have been added (the branch and the join) to cope with modification of attribute values at these junctions. The attributes used between the components are as follows.

s,	oil supply
p,	oil pressure
f,	oil flow
e,	oil temperature
a,	oil foam
r,	resistance to flow

The first five attributes are passed in the direction of oil flow whereas the resistance is transferred in the reverse direction. Thus a change in resistance in the gear box (perhaps due to a change in temperature) will be passed back to the *pump where it will affect delivery pressure*.

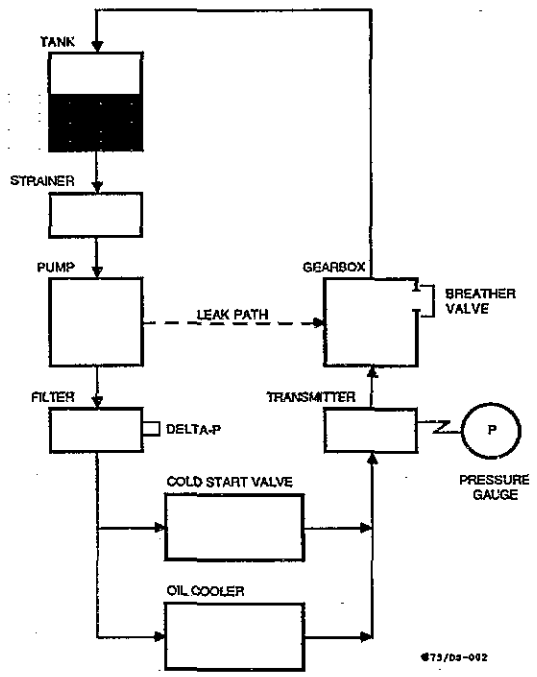

Figure 3. Basic Oil System Diagram

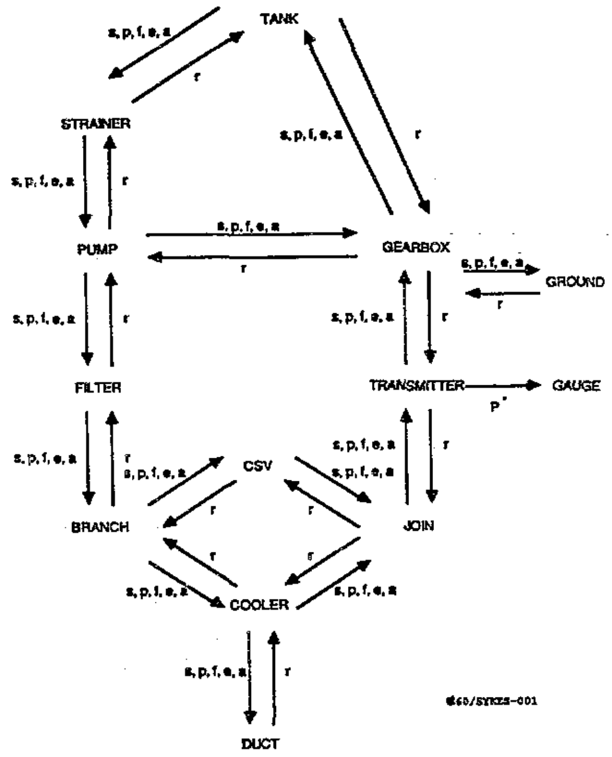

Figure 4. Directed Graph

The behavior of each component in the system is represented by a set of rules. The values of the *output attributes are a function of the input* attribute values and the current state of the object. For example, consider the pump shown in Figure 5. The rpm is set externally by the model user as is the state which is normal or malfunctioning. One of several rules that define the pump's behavior is as follows.

```
If the pump state is normal
    and oil supply is present
    and the rpm is idle
    and the resistance is low
Then the output pressure is low
```

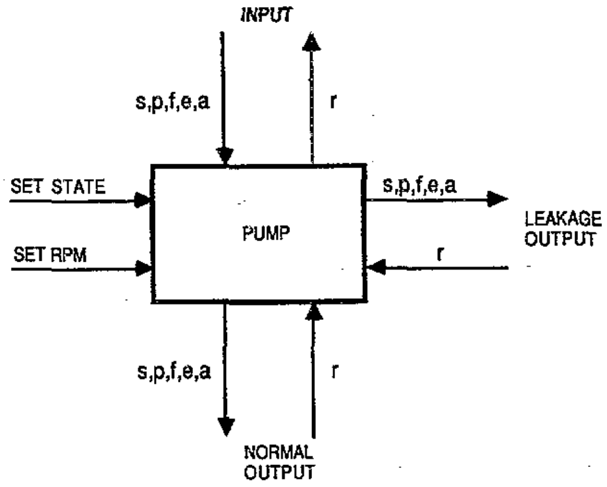

Figure 5. Pump Interface

Each component has its own independent rule set and when connected as shown in Figure 4 the model behaves in a similar manner to the real system.

Results of the Simulation

By running the simulation model several times with a different fault inserted each time, a pattern of attribute values is obtained for each case. For diagnostic purposes this can be used in reverse such that the given pattern of attribute values infers a specific malfunction. Table 1 shows the *symptom/malfunction matrix resulting from the* simulation. Tables 2 and 3 define the attribute values X and the malfunctions Y inserted at each run. Note that not all attributes are included; e.g., resistance and flow. This is a good example of underlying mechanisms not externally visible which separates deep reasoning from shallow reasoning. The results in Table 1 were then used to derive a diagnostic ES for the oil system.

Generation of the ES

The diagnostic procedure starts with the recognition of an initial symptom,usually zero oil pressure or abnormally low oil pressure. Following this a *series of tests is performed to determine* further symptoms; i.e., other attribute values. The sequence of tests is based on the value of each test defined as follows.

$$\text{Value of Test} = \frac{\text{Information Gained}}{\text{Cost of Test}}$$

Table 1. Symptom Malfunction Relation

X1	X2	X3	X4	X5	X6	X7	X8	X9	Y
L	2	1	0	0	1	0	0	0	0
H	4	1	0	0	1	0	0	0	0
*	0	1	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	1	2
L	1	1	0	1	1	0	0	0	3
H	3	1	0	1	1	0	0	0	3
+	0	1	0	0	0	0	0	0	4
+	0	0	0	0	0	1	0	0	5
+	0	1	0	0	1	0	0	0	6
+	0	1	0	0	1	0	1	0	7
L	1	1	0	0	1	0	0	0	8
H	3	1	0	0	1	0	0	0	8
*	0	1	1	0	1	0	0	0	9
L	1	1	1	0	1	0	0	0	10
H	3	1	1	0	1	0	0	0	10

* = any value of X1
+ = L or H values of X1

Table 2. Attributes and Eligible Values

X	Attribute	Eligible Values	
X1	rpm	0	zero
		L	idle
		H	mil/max
X2	indicated oil pressure (p')	L	idle
		1	low at idle
		2	normal at idle
		3	low at mil
		4	normal at mil
X3	oil supply	0	no oil in tank
		1	some oil in tank
X4	foaming oil	1	no foam
		1	foam present
X5	oil temperature	0	warm (normal)
		1	hot
X6	breather pressure	0	no pressure
		1	normal pressure
X7	oil in the ducts	0	no oil in ducts
		1	oil in ducts
X8	delta-p	0	not extended
		1	extended
X9	oil on the ground	1	oil on ground
		0	no oil on ground

Table 3. Malfunctions

Y Value	Malfunction
0	no malfunction
1	sheared pump shaft
2	warm pump gears
3	cold start valve stuck open
4	breather valve stuck closed
5	ruptured oil air cooler
6	clogged strainer
7	clogged filter
8	loose shut-off value
9	pressure transducer total failure
10	pressure transducer partial failure

The information gained is a function of the relative likelihood of the possible malfunctions that could cause the symptom. The cost of the test is simply the time taken to establish if a particular symptom is present or not. By using the data in Table 1 in conjunction with data on values of the various tests (not shown here), the decision tree in Figure 6 was obtained. This represents the optimum sequence of tests to diagnose problems in the oil system. This was then converted to an ES by translating into a query/response system using actual names as shown in Figure 7. This now forms the JPA portion of Figure 2. The goal of deriving an ES from a description of the system operation by deep reasoning has been achieved.

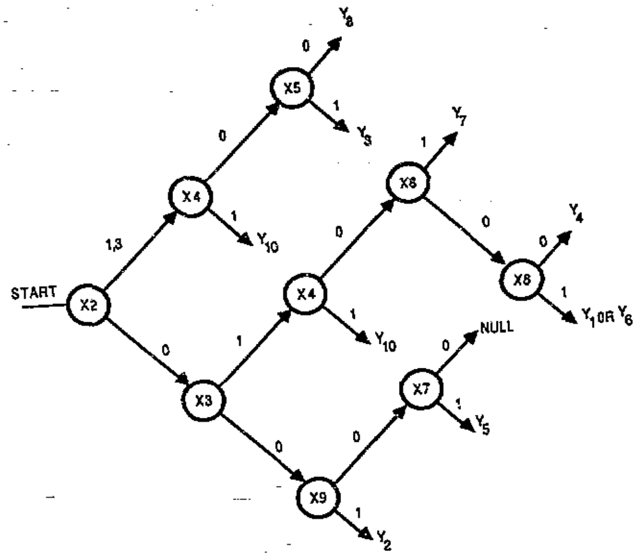

Figure 6. Decision Tree

System Operation

The interface with the user (student and/or problem solver) is shown in Figure 8. Animation using computer graphics is used to display the behavior of the oil system. The user graphics interface also contains a simulated throttle and engine RPM gauge, and an engine start switch (used to activate the simulation model).

A typical simulation model scenario would be to select one of ten possible malfunctions from the menu and "start" the engine. The model receives the malfunction state information from the interface handler, the results or effects of that condition are propagated around the model to all objects, until a new steady state condition is reached.

The user may inspect each object to determine the status of key parameters, such as oil pressure, oil flow, or oil temperature. The student may also inspect the condition or status of any component, such as the pump, or the filter, for the existence of a malfunction. A menu appears on the screen, from which the system user can request component status to be displayed in the component state window.

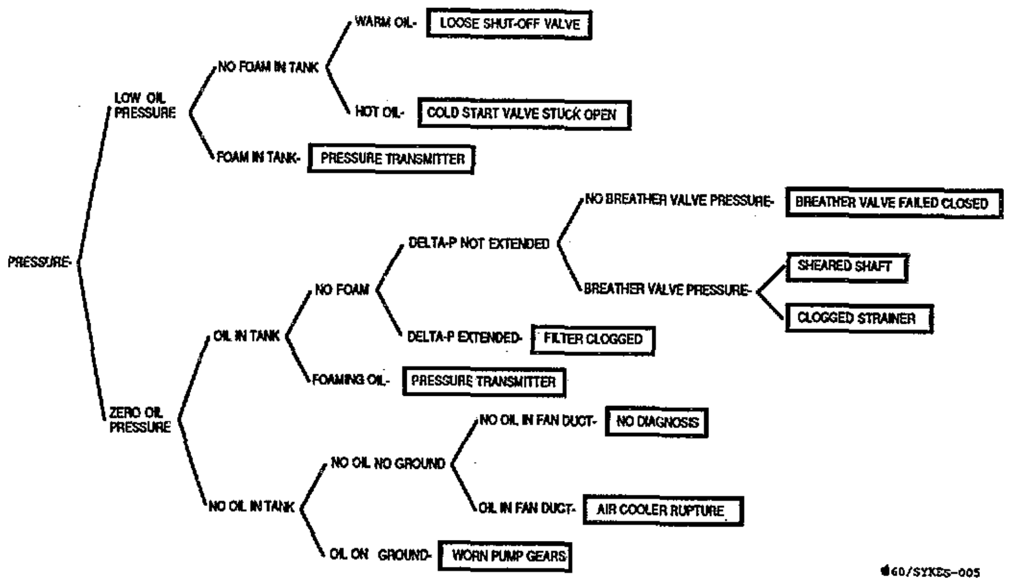

Figure 7. Diagnostic Expert System

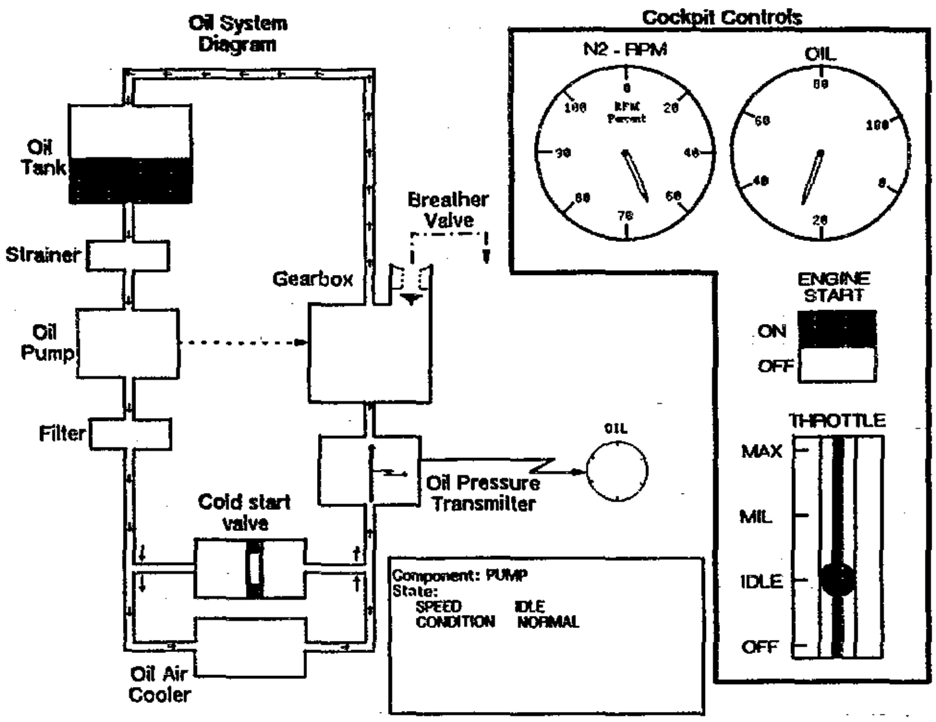

Figure 8. User Interface

Troubleshooting and Fault Isolation

The student may elect to perform his own troubleshooting and fault diagnosis, by examining the various components, until he isolates the fault. Or, the student may select the Expert System option on the Main Menu. In this case, the Expert System guides the student, with a query/response sequence, through an optimum fault isolation and troubleshooting procedure. The Expert System requests certain component status information from the student, based on the student's earlier responses. The student obtains this information by examining the appropriate component in the simulation model's schematic diagram.

In this way, the student may operate in either a practice or test mode. In a practice mode, the student selects the malfunction and performs his own troubleshooting or requests the guidance of the expert system. In a test model, the student can allow the system to select the malfunction, and then perform his own fault isolation troubleshooting procedure.

System Development Environment

The JEDI system was developed on a Xerox 1108 AI workstation. The system configuration included 3.5 Mb, RAM, 40 Mb hard disk, and a 17" graphics monitor (1024 x 808), and a keyboard and mouse. The system was written in Interlisp D, a programming environment based on the lisp programming language. It is widely used within the artificial intelligence community, where it has been used to develop a variety of applications, such as MYCIN (Teitelman and Masinter, 1981). The selection of this programming environment proved very helpful in the actual coding and development phase. The JEDI system including the simulation model, expert system, and interactive graphics interface was completed in a total of ten man days. This included coding, testing, review, changes, and edits.

Recommendations and Future Plans

Following successful demonstration of the prototype JEDI system, the following enhancements and extensions were recommended. First, additional explanation should be added to the JPA Expert System. This feature should not only explain why a certain conclusion was reached, but also explain the significance of the requests for further information as the tree is traversed. When the fault diagnosis is reached, the prescribed corrective action should be requested from the student. This can be enhanced by the use of a video disk system that could also display video/audio segments depicting replace or repair activity. Another recommendation is to add an Intelligent Tutor (which is another form of Expert System) that embodies the skill of a teacher as opposed to the skill of a problems solver.

For the Intelligent Tutor to be effective, it is also necessary to include a Student Model which reflects the student's knowledge and learning abilities. Another area for future work is the true integration of the knowledge bases for JPA and OJT as was shown in Figure 2. This is a nontrivial task. In addition, the scope of the problem should be expanded somewhat so as to represent a real-world practical situation.

ACKNOWLEDGMENTS

The authors would like to thank Horace Barnes of Honeywell T&CSD, David Baum and Chad Thomas of Honeywell Systems and Research Center, Deborah Kelfer and Joshua Stern of Xerox AIS, and Jim Wogulis of UCI for their contributions to the JEDI project.

REFERENCES

de Kleer, J. and J. S. Brown, 1983, "Assumptions and Ambiguities in Mechanistic Mental Models." In D. Gentner and A. L. Stevens (Eds.), Mental Models.

Hollan, J. D., E. L. Hutchings, and L. Weitzman, 1984, "STEAMER: An Interactive Inspectable Simulation-Based Training System." The AI Magazine: Summer 1984, 15-27.

Kuipers, B., 1986, Qualitative Simulation," Artificial Intelligence 29:1, 289-338.

O'Keefe, R., 1986, "Simulation and Expert Systems," SIMULATION 46:1, 10-16.

Richardson, J. J., 1985, Artificial Intelligence in Maintenance, Noyes Publications, Park Ridge, N. J.

Stefik, M. and D. G. Bobrow, 1985, "Object Oriented Programming,; Themes and Variations." The AI Magazine, Winter 1985: 40-62.

Teitelman, W. and L. Masinter, 1981, "The Interlisp Programming Environment." IEEE Computer 14:4 (April): 25-34.

Thomas, C. E., D. J. Sykes, A. Scsigulinsky, 1985, "The Impact of Artificial Intelligence on Maintenance Training," in Proceedings of the 7th Interservice/Industry Training Equipment Conference (Orlando, FL, Nov 19-21): 24-40.

Towne, D. M., 1986, "The Generalized Maintenance Trainer: Evolution and Revolution." In W. B. Rouse (Ed.) Advances in Man-Machine Systems Research: Vol. 3, JAI Press, Greenwich, CT.

ABOUT THE AUTHORS

MR. DAVID J. SYKES is a Project Staff Engineer with the Training and Control Systems Division of Honeywell in West Covina, CA. His responsibilities include development of AI programs for training systems and advanced visual simulation systems. Most recently, Mr. Sykes was the Knowledge Engineer on the AI/OJT prototype described in this paper. Prior to this, he was with Honeywell Information Systems in Phoenix and Boston where he designed distributed processing systems and computer networks. Mr. Sykes has published 15 papers on AI, computer networks and simulation of distributed systems. He holds a BS in Electrical Engineering, a MS in Computer Science and is currently completing his Ph.D. in Industrial Engineering at Arizona State University.

MR. LEE KESKEY is a Project Staff Engineer at Honeywell Training and Control Systems Division, Maintenance Trainer Department. Mr. Keskey has extensive experience in design and management level positions in the development of real-time computer based systems. Mr. Keskey is presently involved in the introduction of new technology into the maintenance trainer product line, with emphasis on artificial intelligence and interactive video disc based applications. Most recently, Mr. Keskey directed the successful development of an AI-simulation based trainer prototype to demonstrate the application of Artificial Intelligence and Expert Systems to maintenance training. Mr. Keskey has a BSEE from California State Polytechnic University, an MSEE from the University of Southern California, and an MBA from Pepperdine University.