International Journal of Artificial Intelligence in Education (1997), 8, 262-283

Approximate Reasoning Techniques for Intelligent Diagnostic Instruction[1]

Douglas M. Towne

E-mail: dtowne@usc.edu
University of Southern California,
Behavioral Technology Laboratories
1120 Pope St., Suite 201C
St. Helena, CA 94574

Abstract. Intelligent instruction of fault diagnosis, or troubleshooting, tasks requires the capability to automatically infer the significance of particular test outcomes observed by the learner in a practice environment. This single process is central to virtually every aspect of intelligent instruction in this domain, ranging from evaluating learner proficiency to recommending an effective testing strategy. In highly complex target systems this task becomes intractable, for the number of possible faults and system configurations is enormous. The issue is not that an artificial diagnostic expert must be highly precise, but that the internal expert must interpret symptoms observed by the learner in terms of the symptoms which *might* emerge from all possible sources, not just the faults that are defined in the pool of sample faults that are presented in the exercises.

The approach presented here operates upon qualitative expressions of the possible symptoms produced by faults in various units of the target system. This greatly reduces the burden placed upon the author of the application, while capturing the essence of his or her symptom knowledge. This approach has been implemented in an instructional system that automatically computes a provisional bank of symptom possibilities by simulating the faults in the exercise pool, then acquires from the author qualitative refinements to those computed possibilities, and finally delivers instruction based upon that body of information.

BACKGROUND

While there have been a number of intelligent tutoring systems (ITSs) that have demonstrated excellent instructional capabilities in the domain of fault diagnosis, the cost of authoring a tutor for a new target system

[1] This work was funded by Office of Naval Research under contract No.F33615-90-C-0001.

increases dramatically with the complexity of that system. This is not to say that existing methods in diagnostic training systems have failed to achieve the instructional objectives their developers established, but rather that the current methodologies are proving to impose very high development costs in the face of increasing complexity in the diagnostic environment. As with all ITSs, therefore, those in the realm of fault diagnosis must be judged both by their ultimate instructional power and the cost of authoring new applications. A brief examination of the existing methodologies will establish the problem.

Approaches

The approaches that have been taken to author and deliver intelligent instruction in fault diagnosis tasks may first be considered under two major categories:

1. Knowledge engineering (KE) approaches, such as Sherlock (Lesgold, Eggan, Katz, and Rao, 1992; Lajoie and Lesgold, 1992), which capture and operate upon one or more experts' strategic and tactical knowledge about diagnosing particular problems in a particular domain. In Sherlock the underlying domain expertise is a hierarchy of subgoals, each representing a problem solving method for some problem area. Upon this structure, and a graphical simulation of the target system, computer-coached practice is supported.
2. Model-based approaches, in which the target system is represented via a model that serves to inform a domain-independent model of diagnostic expertise about its normal and abnormal behaviors. Typically the learner operates upon a separate graphical representation of the target system, and the instructional functions are kept advised of the tests performed by the learner on that presentation.

The model-based approaches can be further differentiated by the manner in which the actual device is modeled. Existing approaches range from very superficial device models to deep models that represent the physical processes that govern the system's operation. These differences produce markedly different levels and types of diagnostic reasoning processes that can be conducted by the ITS. The following three categories typify model-based approaches.

1. 'Signal–tracing' approaches, such as MITT (Johnson, Duncan, and Hunt, 1988; Wiederholt, Norton, Johnson, and Browning, 1992), represent a particular device in terms of its normal connectivity. The basic elements of a MITT-type model are 1) blocks, which may be functionally defined or physically defined, and 2) inputs to and outputs

of those blocks. Test outcomes are then interpreted in terms of the system units which are, or are not, connected to the observed indication in some way, i.e., a normal test outcome absolves the units that are 'upstream' from the observation, while an abnormal outcome implicates the predecessors.

2. Simulation approaches, such as IMTS (Towne and Munro, 1988; Towne, Munro, Pizzini, Surmon, and Wogulis, 1990), represent the target system via a composition of objects, each of which carries rules stating how that object behaves normally and misbehaves when failed, in terms of its surrounding environment. This system model can then be automatically executed under a wide range of modes and failure conditions to produce a bank of symptom information suitable for generating expert interactions with the learner.
3. Physical process models, which represent a device or system via the underlying physical principles that govern its operation. For example, the processes might be chemical, biological, electronic, or thermodynamic. One such system, CyclePad (Forbus and Whalley, 1994), supports the learner in both analyzing a complex thermodynamic system and experimenting with design alternatives. Because the physical process being modeled is well-defined and stable, the instruction can deal with a learner in qualitative terms as well as quantitative ones.

These four approaches (KE, signal tracing, simulation, and process models) have some commonalities. All may employ an interactive graphical representation of the target domain upon which the learner works, and may provide instructional expertise which determines such issues as when to intervene, what support to provide, and what faults to present as exercises. The fundamental difference between them is that the KE approach requires the authors to explicitly express their troubleshooting strategies, while the model-based approaches require the authors to explicitly express knowledge about the operation of the target system. This difference results in markedly differing authoring costs and benefits.

Weaknesses of the Four Prototype Approaches

All four of the foregoing approaches have been implemented, and have demonstrated excellent instructional capabilities. Each method has performed admirably in its own intended domain. Strangely, if we wish to now produce a new diagnostic ITS for a different target system, we find that none of these approaches is likely to both produce excellent

instruction and permit development at a reasonable authoring cost. Let's consider the problems with each.

KE Methodology

While very impressive instructional dialogues can be authored and delivered via the KE approach, that result comes at an extremely high price. KE methodology requires specialized skills and time–consuming efforts to acquire the rules that experts use in diagnosing specific systems. The developers face the daunting task of making explicit that which is often implicit and subconscious.

The workload upon the author increases dramatically with increased system complexity, leading to very high costs and unknown reliability of the captured expertise. Furthermore, the captured strategic expertise is necessarily couched in some assumed maintenance environment (such as depot level, flight line, etc.). Thus, changes in the maintenance setting or to the design of the target system can require that the knowledge base be reworked.

Signal tracing

At the other extreme is the signal tracing approach which is certainly the easiest to apply. In the face of extremely complex target systems, the ITS author may elect to limit the level of decomposition reflected in the connectivity model, thereby keeping development cost well within reason. The fatal flaw of the connectivity model approach is that its diagnostic inferences are based upon the assumption that fault effects propagate along the connections expressed in the static and fault-free input-output representation, and that the direction of these effects can be expressed via static connectivity specifications. In the real world, there are at least three ways these assumptions are violated:

1. different faults produce effects that propagate in different directions; no one signal flow serves all;

2. not all indicators are sensitive to all abnormalities that pass through them; and

3. many faults change the connectivity of the target system, invalidating the inferences made upon the fault-free representation.

This form of system representation can represent *normal* system operation in an instructionally useful manner, but lacking a behavioral model of the target system, it can reliably support diagnostic instruction only for those ideal target systems that do not violate the assumptions.

Simulation

The simulation approach dynamically determines the states of elements in the target system in normal and abnormal conditions, thereby generating the symptom information needed to support diagnostic instruction via an artificial expert. For target systems involving perhaps less than a thousand possible faults, this approach is highly effective, both in terms of instructional power and development cost. When applied to a target system that could fail in millions of ways, however, it becomes infeasible to simulate enough failures to support diagnostic reasoning about the faults in the exercises. It might appear that there are at least two workable ways to resolve this complexity problem:

- sample the possible faults, rather than attempting to produce a complete symptom data base; and
- reduce the level of decomposition of the target system to simplify the system model

Unfortunately, neither of these methods succeeds. Incomplete symptom knowledge is perfectly acceptable when computing good next tests, for there is no requirement that the proposed tests be optimal. However, incomplete symptom knowledge will result in incorrect statements from the ITS to the learner concerning the implications of certain readings, leading to ineffective instruction and well-deserved loss of credibility. The second possible solution, viewing the target system at a higher level of granularity, fails because this does nothing to reduce the true number of possible failures. A circuit board, for example, has as many, or more, fault possibilities as the sum of its parts, and those possibilities directly impact all discussions of test implications. This does not to suggest that an immense number of faults must be presented in exercises for instruction to be effective. Regardless of the instructional method, a relatively small set of well-chosen faults can provide very useful instruction.

Physical process models

The physical process models can deliver extremely powerful instruction at a very tolerable cost. While authoring a physical process model involves a substantial investment, the number of possible applications and the number of prospective learners is typically quite large, thus the development cost is easily justified. The problem with this approach is simply that it cannot take on applications outside its domain. It might seem that process models in such areas as electronics and electricity would support the modeling of most modern digital systems. Unfortunately the effort involved in

representing even a single CPU for example is immense, and simulating an entire system, in all its possible fault modes, is essentially infeasible.

APPROXIMATE REPRESENTATIONS OF FAULT EFFECTS

We require, therefore, some means for capturing and representing the body of symptom knowledge possessed by domain experts, while avoiding the necessity to enumerate the individual effects that result from each possible failure. The approach outlined here accomplishes this by having the domain expert express, without enumerating specific faults, the possible effects that could result from failures in each replaceable unit. The basic approach adapts methodology from a body of work formerly called 'fuzzy set theory' (Zadeh and Kacprzyk, 1992), but more recently changed to Zadehan theory in honor of its primary developer, Lotfi Zadeh.

Expressing Fault Effect Knowledge in Approximate Terms

We ask the domain expert to specify the possible symptoms that failures in a unit will produce, without asking for explicit fault-effect pairs. Under this scheme, the domain expert picks one of seven English terms — *always, usually, very_often, often-as-not, sometimes, rarely, never* — to indicate the extent to which a particular replaceable unit will produce particular symptoms at a specified indicator when that unit fails. Figure 1 indicates graphically the information this process yields.

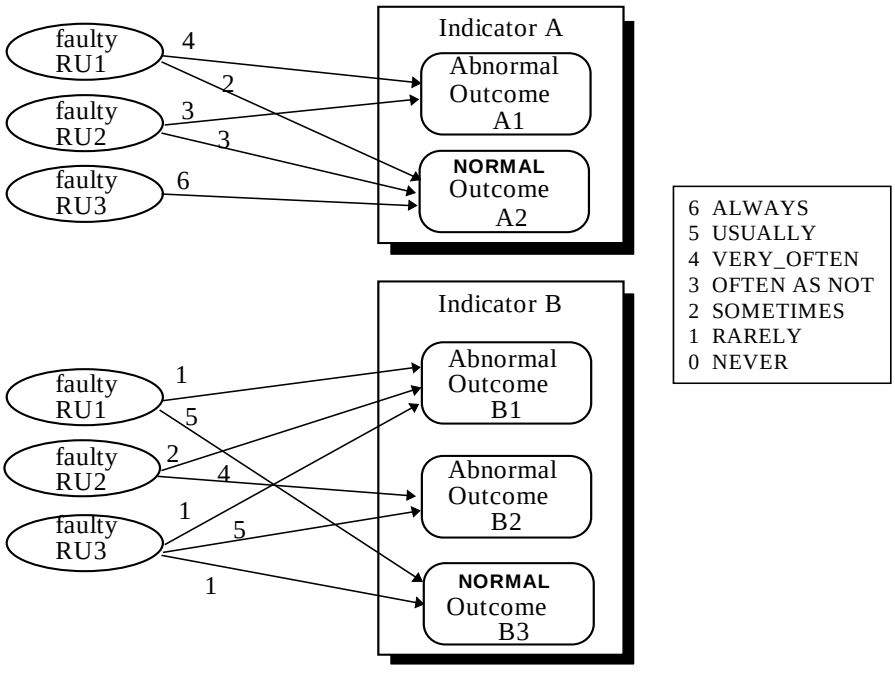

Figure 1. Fault Effect Specifications in Graphical Form.

The figure reflects the effects of faults in three replaceable units (RU1, RU2, and RU3) on two indicators (A and B). The top portion of the figure indicates the manner in which failures impact Indicator A, the lower part repeats this for Indicator B. The numbers 1 through 6 on links between RUs and indicators give the 'causal strength' of functional relationships between the RUs and the indicators. Strengths of 0 (the outcome never occurs) are not represented explicitly in the figure.

According to the causal links shown, faults in RU1 will *very often* impact indicator A by producing abnormal symptom A1, and *sometimes* faults in RU1 do not adversely impact indicator A, and thus normal indication A2 would be maintained. By contrast, faults in RU3 *never* adversely affect indicator A, and we will observe a normal outcome regardless of the manner in which RU3 fails. We can say that there is no functional relationship between RU3 and indicator A. An example would be a fuel injector in an automobile having no functional relationship to the brake light.

The set of fault effect specifications linking one RU to one indicator is termed a *fault effect statement*. In English, the fault effect statement for RU3 and indicator B is:

Failures in RU3:
- rarely produce outcome 1, and
- usually produce outcome 2, and
- rarely produce outcome 3

at indicator B

Precision, Accuracy, and Consistency Considerations

Each fault effect statement implicitly reflects the domain expert's knowledge about the ways a particular unit can fail and the manner in which those failures can affect an indicator in the target system. Obviously, natural language is imprecise, which is also one of its great strengths. Even when considering single-word qualifiers, there can be differences among people concerning their meaning. We supply the following table to the domain expert, as a guide to the use of the seven likelihood qualifiers.

Qualifier	Portion of Faults that produce the Outcome
ALWAYS	virtually all
USUALLY	nearly all, with some exceptions
VERY_OFTEN	more than half
OFTEN_AS_NOT	about half
SOMETIMES	some, but less than half
RARELY	very few
NEVER	none

Without doubt, the domain expert cannot think though all the ramifications when there are thousands of possible faults in a unit. Nevertheless, domain experts can say things like “Most faults in this unit will affect this indicator, but very few affect that indicator.” They can also make more extreme statements, such as “There is no relationship between this unit and that indicator, thus failures in this unit cannot affect that indicator.” This level of precision has been found to be quite attainable from domain experts, and is fully adequate to author a diagnostic ITS. Importantly, this qualitative basis also provides a natural source with which to generate English language dialogs to rationalize the internal expert’s inferences and recommendations. Examples of this are provided in a later section.

There is certainly no doubt that different domain experts would supply different likelihood judgments. The impact upon instruction of errors of judgment by the domain expert are generally proportional to the magnitude of the error. If a domain expert indicates that some outcome *rarely* occurs, when in fact it *often* occurs, then there will be some distortion in the level of suspicion which the internal expert attaches to some faults during instruction, and this distortion could somewhat affect the recommended diagnostic strategy. The distortion would only be critical, however, if the fault effect statement specified an *always* relationship when it should be *never*, or vice versa. It should also be noted that the fault effect statements of two or more experts could be compared automatically, since the repertoire of fault effect statements is fixed for a particular model of the target system. So, once that model is developed, two or more domain experts could supply the required fault effect judgments, and those could be checked for consistency. This is not currently implemented in the authoring system described below, but is a very feasible approach to determining the validity of the knowledge base.

A fault effect statement should also be internally consistent – the sum of the likelihood qualifiers should reflect totality. To illustrate this, suppose we assign the following percentages to the seven qualifiers:

Qualifier	% of Faults
ALWAYS	100
USUALLY	95
VERY_OFTEN	75
OFTEN_AS_NOT	50
SOMETIMES	25
RARELY	5
NEVER	0

Using these values, we can sum the qualifiers in any fault effect statement. The sum for the fault effect statement for RU3 and indicator B

listed previously is 105% (one *usually* and two *rarely's*), which is certainly not a serious inconsistency. Clearly, a fault effect statement that totals 190% would be inconsistent (e.g., two symptoms classified as *usually*), as would be one that totals 30%. The authoring system described below points out large inconsistencies as the domain expert supplies the information in the authoring system. Minor inconsistencies, however, are not flagged for correction. Instead, the authoring system factors the supplied qualifiers so that the total likelihood is 100%. Consequently, the methodology does not burden the domain expert with maintaining unnecessary precision.

This discussion of quantities points out that there is nothing magical about the seven qualifiers. We could alternatively ask the domain expert to estimate the percent of faults that would produce each outcome, rather than asking for a selection of English terms. Doing this, however, changes the nature of the domain expert's task significantly, and suggests a much higher precision standard than is really required.

Additional Domain Expertise Required to Support Intelligent Instruction

In addition to the fault effect information, the domain expert provides to the authoring system 1) qualitative judgments of the reliabilities of the RUs, 2) approximate RU replacement times, 3) a pool of specific faults for presentation within exercises, 4) a set of verbal *problem reports*, one of which is issued to the learner at the start of each exercise, and 5) a technical explanation of each fault.

RU Reliability

The reliability of each unit is specified relative to the other units in the target system, under one of the following five categories:

1. *least* reliable of the units in the system;
2. *less* reliable than most other units in the system;
3. *average* reliability of the units in the system;
4. *more* reliable than most other units in the system; or
5. among the *most* reliable of the units in the system.

Based upon these judgments, the instructional system rank orders its suspicions of each unit at the start of each problem. As symptom information is obtained by the learner, within a simulation environment described later, the instructional system updates its suspicions of the units. Consequently, the units are initially sorted into at most five rank levels,

and then they disperse into many more levels of suspicion as symptom information is received and processed.

Replacement Times

The RU replacement times impact the decision to conduct further testing, versus replacing the most suspected RU. Sometimes two equally suspected RUs cannot be discriminated from the test opportunities that the design of the target system offers, thus the learner must replace the one with the shorter replacement time (RU cost is also a vital factor in this decision, but this is not currently used by the instructional authoring system).

Pool of Faults

The pool of specific faults are particular failures in particular replaceable units. These faults are chosen to be instructionally appropriate and representative of faults found in the real world. While the learner may never be required to determine exactly what in the RU failed, or how that component failed, the specific symptoms produced by a particular sample fault are fully determined only when the fault is fully defined.

Problem Reports

For each exercise, the domain expert authors a verbal problem report which is issued at the start of the exercise. This statement can be purposely vague or specific, helpful or not helpful. The following problem statement is typical.

> The Overload light keeps coming on when we switch from Standby to Radiate mode.

Technical Explanation

Also, for each fault in the exercise pool, the domain expert prepares a technical explanation of its impact upon the target system. This technical content is presented to the learner at the conclusion of each exercise. An example of such an explanation is shown here.

> There was a failure in the Optical Fan Sensor A14 which caused it to cut off the fan and raise the alarm condition. Any loss of signal in the first stage amplifier circuit will also produce this abnormality.

GENERATION OF DIAGNOSTIC EXPERTISE

This section will outline the processes that are executed within a diagnostic ITS which employs the approximate fault effect information to represent the real world. This discussion assumes that the learner is operating upon a graphical simulation of the target system.

Problem Selection and Fault Introduction

The first task of the ITS is to select a problem that offers the appropriate level of difficulty for the individual learner. Problem difficulty is computed by automatically examining the symptoms produced by each candidate fault. Let's consider two candidate faults, fault x and fault y, both occurring in RU1 of Figure 1. Suppose these two faults produce the following symptoms:

Fault	Indication at A	Indication at B
Fault x in RU 1	A1	B3
Fault y in RU 1	A2	B1

From Figure 1 we see that each fault produces one abnormal symptom and one normal symptom. Therefore there is no difference in the difficulty of detecting the presence of the two faults. All other factors being equal, faults that produce very few abnormalities are more difficult to resolve than those that provide more evidence of their existence.

Secondly, note that fault x corresponds well with the approximate symptoms of RU1; both outcomes A1 and B3 are representative of faults in RU1 (A1 occurs very often; B3 occurs usually). Fault y, however, will be much more difficult to identify, since its symptoms do not correspond well with the population of faults that typically occur in RU1 (A2 occurs sometimes; B1 occurs rarely). Thus, fault y will be much more difficult to diagnose than will be fault x.

It is also important to realize that a particular fault might exhibit symptoms that are more representative of a different RU than its true host RU. For example, a fault in RU1 that gives symptoms A2 and B2 looks more like a faulty RU3 than a faulty RU1. An expert troubleshooter would rationally suspect RU3 over RU1 after observing these symptoms.

After analyzing and rank ordering the faults that are available for exercises, the ITS attempts to select a fault that presents an appropriate level of difficulty for the individual. If the learner's previous problem was performed successfully and with few calls for automated guidance, then difficulty is maintained or increased. Otherwise problem difficulty is reduced below that of the previous problem.

Upon selecting the fault to be presented, the ITS introduces it into the simulation of the target system. In our simulation environment this is done by setting a *failure_state* variable which impacts the behavior rules of the RUs. This produces the normal and abnormal symptoms that would be seen on the real system, under that fault condition.

Symptom Interpretation and Maintenance of Suspicions

Recall that at the initiation of each problem the ITS rank orders its suspicion of the RUs to reflect their relative reliabilities. As the learner performs diagnostic work on the simulated target system, the internal diagnostic expert interprets, transparently to the learner, each symptom displayed according to the approximate fault effect information, and it revises its suspicion rankings. Suppose, for example, that indicator B in Figure 1 is a Power Meter with a normal reading of 12 VDC, and that RU3 is a Low Voltage Power Supply (LVPS). If a learner observes 12 VDC at the meter, the diagnostic expert would significantly reduce its suspicion of the LVPS, as this normal reading is rarely observed when the LVPS fails. Likewise, it would drastically reduce its suspicion of RU2, since failures in this unit always produce some abnormal reading at the meter. Thus, after this indication is observed, the ITS would greatly increase its suspicion ranking of RU1, compared to the other two RUs. Alternatively, suppose the learner observes outcome B2 at the meter. Then the internal diagnostician would drastically reduce its suspicion of RU1, since this outcome is never produced by faults in RU1.

The magnitude of the reduction in suspicion level for a particular RU is proportional to the rarity of observing that symptom, according to the symptom information. If a symptom is observed that can *never* occur when a particular RU fails, then that RU suspicion ranking is reduced by the most possible. A symptom that *sometimes* occurs only slightly reduces an RU suspicion ranking.

Making Replacement Decisions

As evidence is accumulated by the learner, the RUs that could produce the observed symptoms move toward the top of the suspect list. If one RU emerges well above all the others, the instructional system records that the learner has acquired enough evidence to effect a replacement. Furthermore, the instructional system records whether or not a particular replacement is rational, i.e., did the learner replace a highly suspected unit? This is a crucial facility, since it allows the ITS to correctly credit a learner for making a rational replacement, even if that replacement turns out to be incorrect.

This does *not* promote wholesale swapping by the learner. The instructional system also detects when the learner makes a premature replacement, whether that replacement turns out to be correct or not. In this case, the learner replaces a unit which has not been brought under suspicion by the observed symptoms, and the ITS so notes in the data accumulated for each student.

Generating A Testing Sequence

We require a process that is highly proficient in making and justifying next test decisions. This capability then supports the instructional functions of 1) demonstrating expert solution strategies, 2) assessing the learner's test selections, and 3) suggesting good tests to perform next. Previous research (Towne and Johnson, 1987) has shown that excellent testing strategies can be generated by repeatedly performing that test which has the greatest potential for reducing *uncertainty*, per unit of time invested. System uncertainty, or entropy, is computed using Shannon's measure:

$$U = \Sigma\, p_j \log_2 p_j,\ j=1, \text{ number of hypothesis}$$

where p_j is the probability that hypothesis j is true.

To compute the expected uncertainty that would result from performing any test, the diagnostic expert notes the likelihoods of each outcome of the test (available from the fault effect statements), and projects the suspicion levels that would result from each outcome. From this, the expected system uncertainty can be determined. Those tests that offer the greatest reduction in system uncertainty, per unit time to perform, are the tests experts would perform at that stage in an ongoing exercise.

A simplified version of this has now been implemented which produces effective next test decisions with the qualitative symptom information discussed here. Under this process, the ITS determines, at each stage of a diagnostic problem, the extent to which each test is likely to change the RU suspicion rankings. The most productive tests are those promising the greatest impact upon suspicion rankings. An ideal test would be one that has many possible equally likely outcomes, each of which clearly eliminates some of the more highly suspected RUs from suspicion. Note that this is in harmony with the 'half–split' concept often taught, but is much more generalized than that simplified heuristic.

IMPLEMENTATION

The foregoing ITS authoring and instructional delivery concepts have been implemented in a system named DIAG (Diagnostic Instruction and

Guidance). DIAG provides the tools to author a diagnostic ITS and to deliver intelligent diagnostic instruction. The key feature of this system is that it generates all of the diagnostic instruction automatically, using the simulation model of the target system and the symptom knowledge base authored for it.

Authoring a New Diagnostic ITS

The steps in authoring a DIAG ITS are as follows:

1. produce an interactive model of the target system and establish modes of operation
2. provide RU reliabilities and RU replacement times;
3. specify the faults in the exercise pool;
4. specify each exercise (fault, problem report, initial mode, and maximum time);
5. produce the approximate RU symptom information.

Interactive Model

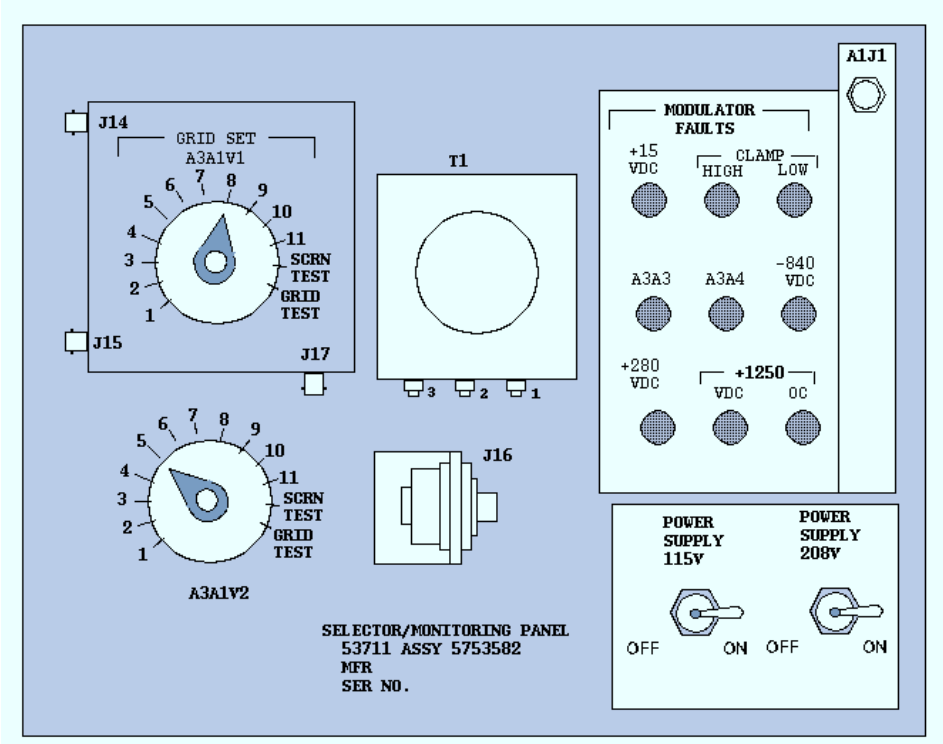

Figure 2. A Portion of a Device Model (1 part of 17)

The interactive model of the target system is produced using the RIDES[2] (Munro, Johnson, Surmon, and Wogulis, 1993; Towne, 1994) simulation authoring system. The resulting model responds to user actions, such as changes in control settings or attachment of test equipment, and to the failures that are introduced in the troubleshooting exercises. DIAG provides the authoring tools to structure the model in a hierarchical fashion, permitting the author to represent different subsystems to any desired level of detail. One section of a 17-section device model is shown in Figure 2.

Once the system model is constructed, the author defines the modes of operation that will be used in the exercises by setting the switches and keying in associated mode names.

RU Specifications

The author completes the DIAG dialog box shown in Figure 3 to specify the names, replacement times, and reliabilities of the replaceable units.

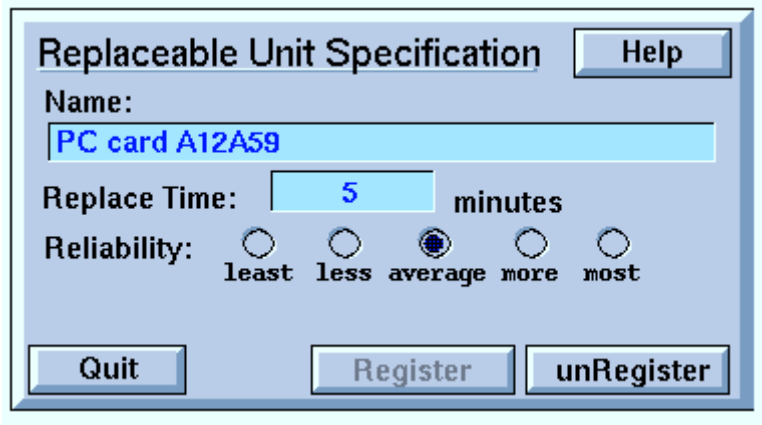

Figure 3. Authoring Interface for Specifying Replaceable Units.

Fault Specifications

The DIAG author enters a verbal account of the way each fault impacts the system. One of these is presented to the learner at the conclusion of each exercise. This account can be a simple statement or a much more technically detailed explanation.

Exercise Specifications

The DIAG author specifies an exercise by 1) selecting the fault that will be presented in the exercise; 2) authoring the problem statement to be issued

[2] RIDES was developed under Air Force funding, Contract No. F33615-90-C-0001.

at the start of the exercise, 3) selecting the mode in which the system will initially be presented, and 4) setting a maximum time limit allowed. This design permits the author to form multiple exercises for any fault, in which the problem is made easy or difficult by the extent to which the problem statement is informative.

Approximate Symptom Information

Finally, the DIAG author produces the symptom data that supports DIAG's diagnostic reasoning process. First the author commands DIAG to generate a provisional set of fault effect statements. DIAG does this by simulating each fault and noting the frequency of the various outcomes. If the fault set is large compared to the number of possible faults, then these automatically generated relationships will be nearly complete; if the fault set is small, the relationships will not reflect reality to a high degree. Next the author refines the automatically produced fault effect statements to reflect his or her much broader understanding of the target system, and its possible faults. The dialog box of Figure 4 is used for this purpose (the provisional fault effects are reflected in the dialog box as the author addresses each condition).

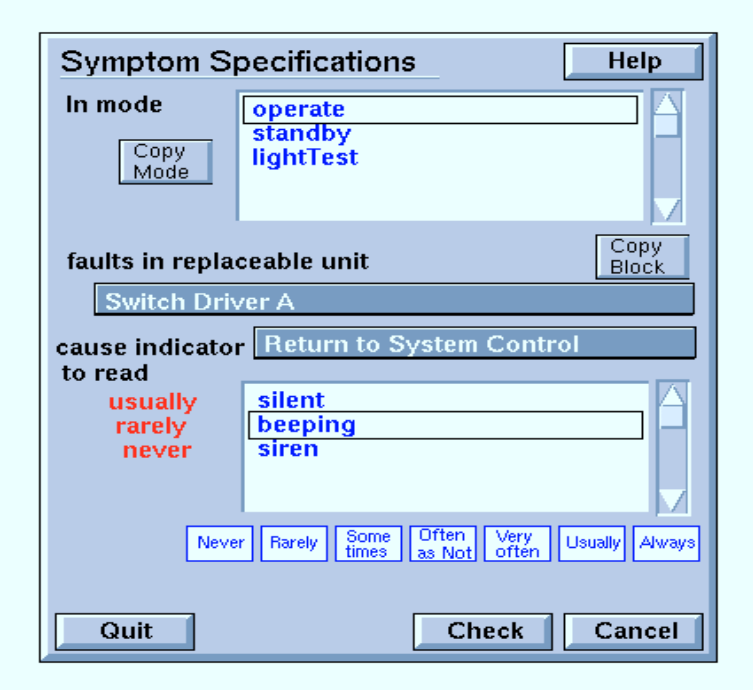

Figure 4. Authoring Interface for Specifying Fault Effects.

Here the author has produced one fault effect statement with five mouse clicks: 1) selected the mode from the list in the dialog box, 2) selected the RU from the graphical device model, 3) selected the indicator from the graphical device model, 4) selected the symptom "beeping", and

5) selected the qualifier “Rarely”. The resulting fault effect statement that DIAG retains internally is:

> In Operate mode:
> Failures in RU Switch Driver A:
> • rarely produce outcome “beeping”
> at indicator Return to System Control

A few more mouse clicks establish the possibilities of hearing “siren” and “silent” at this audible indicator under this fault condition. Then the DIAG author selects a different affected indicator and repeats the process. The authoring burden is significantly reduced by only requiring symptom specifications for abnormal outcomes.

GUIDED PRACTICE

When the authoring phase is complete, DIAG will deliver guided exercises in fault isolation, administering all aspects of the instruction. The first problem for a learner is selected, its problem report is presented, and the learner starts working the problem.

Within-exercise Support

As the learner conducts diagnostic actions, DIAG keeps records of the tests that were performed and the inferences which could be made from the symptoms shown. To consult with DIAG, the learner clicks a *Consult* button then chooses a consultation type from the presentation shown in Figure 5.

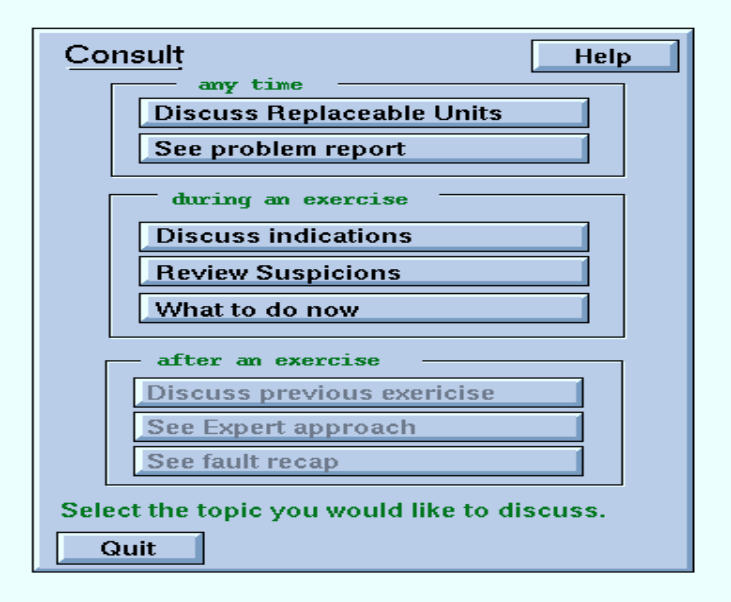

Figure 5. DIAG Consultation Selection Box. (The selections *See problem report* and *See fault recap* simply present that information again.)

Discuss Replaceable Units

The learner selects this consultation type when he or she wants to discuss how the symptoms seen in an ongoing exercise relate to a particular RU. DIAG then constructs a dialog that reviews and interprets the symptoms that were exhibited that bear upon the selected RU. The following is DIAG's response when a learner selected an RU named PC card A12A4:

```
PC card A12A4 is one of the stronger suspects,
   however some indications you have seen contradict that theory.
Here are some of them:
Vent Fan B2 sound was NOT_RUNNING in Radiate mode.
   This is an abnormal symptom (normal is RUNNING)
   which rarely results when this unit fails.
```

This reflects that DIAG was also suspecting this RU, based upon the symptoms the learner had seen, but that some symptoms do not support that theory.

Discuss Indications

The learner selects this consultation type when he or she wishes to discuss some currently exhibited symptom. DIAG first states the normality of the selected symptom, then it interprets the symptom in terms of the RUs that are currently most suspected. The following is DIAG's response when a learner selected the PC Card Interlock Indicator:

```
The PC Card Interlock Indicator is off which is
normal in this Radiate mode.
   Ventilating Fan B2 has no effect on this test.
   Optical Fan Sensors A3 and A14 has no effect on this test.
   PC card A12A36 sometimes
         allows this normal indication even when failed.
```

This indication does not rule out any of the currently most suspected RUs. If it had, DIAG would have stated that the symptom rarely or never results from failures in the selected RU.

Review Suspicions

The learner selects this consultation type to go over his or her current suspicions. After the learner selects the RUs he or she most strongly suspects, DIAG critiques those suspicions, while not revealing the true fault.

> The symptoms you have seen so far do not justify your suspicion of 3 of the units you indicated.
> These symptoms implicate 6 other units that you should also suspect.

Here the learner is suspecting three RUs which DIAG has ruled out, and is failing to suspect six other RUs that should still be under suspicion.

What to Do Now

When the learner makes this request, DIAG responds with the best next test to perform, considering the progress that has been made (symptoms seen) in the exercise so far.

> A good action now is to check Vent Fan B2 sound in Radiate mode.

After-Exercise Support

Upon completion of an exercise, the learner sees the fault recap that describes the mechansims by which it affected the target system. At this point, the learner may proceed to the next problem, or may request either or both of two kinds of debriefings about the just completed exercise:

1. a review of the learner's work on the previous exercise, and a discussion of the significant test outcomes that were seen, and
2. a demonstration of an expert diagnostic strategy for that problem.

Discuss Previous Exercise

For this debriefing type DIAG steps through each test performed by the learner that had significant impact upon the suspicion rankings, summarizing and displaying the symptom that was originally exhibited, noting whether that symptom was normal or abnormal, and listing the RUs that should be most suspected at that stage of the exercise. The textual presentation for one test appears as follows:

> Channel 2 Arc Indicator was on in Radiate mode, which was abnormal (normal is off).
> The most suspected units are now:
> T6
> Traveling Wave Tube V2
> Vac-ion Power Supply A15
> PC card A12A48
> PC card A12A4

See Expert Approach

For this consultation type DIAG diagnoses the fault in the previous exercise in an expert manner, explaining each test it selects, showing the symptom that results from the test, and listing the units that should be most suspected at that stage of the diagnostic sequence. The textual part of one step of the expert demonstration is shown here:

```
We check A1 Standby Button in Radiate mode.
    It is green which is abnormal.
(Normal is off)
The most suspected units are now:
PC card A12A48
T6
PC card A12A32
PC card A12A22
PC card A12A4
```

FINDINGS AND CONCLUSIONS

The effectiveness of DIAG has been partially demonstrated via a large prototype application. This application says a great deal about the cost-effectiveness of the authoring process, the robustness of the system in addressing complex systems, and the sufficiency of qualitative symptom information in generating meaningful instructional dialogs and demonstrations. The application alone says nothing about the effectiveness of that instruction that is produced, which for now we must judge simply by considering the kinds of instructional processes and dialogs that are produced.

Prototype Application

The application addressed a portion of a very large shipboard radar transmitter system spanning multiple equipment bays. The application focused on one of the major units, a Driver/PreDriver equipment. The system model was presented as 17 screens of operable simulation graphics, and consisted of 86 replaceable units, 49 active indicators, 31 operable controls, and 17 test points. The application was developed by the writer in approximately five weeks, while the domain expertise was provided by a Navy technician[3] in approximately four weeks, two of which were devoted to production of the fault effect information. The writer entered the fault effect information to DIAG in approximately one week. The technician

[3] Petty Officer James Armstrong, AEGIS Training Center, Dahlgren, VA.

who provided the fault effect information did so without difficulty, following less than one hour of direction. No inconsistencies were evident in the judgments he made, and he worked without assistance to completion of the effort.

Analysis

The very modest time investment to produce such a large application indicates that this methodology of representing expertise about the target system can be highly cost effective. Moreover, the application could be extended at any time with minimal impact on the existing knowledge base. Because the fault effect statements are specific to particular RU–indicator pairs, the addition of more test points, indicators, or possible fault areas would have no impact on the existing knowledge base.

There has not been a formal evaluation of instructional effectiveness. The instructional capacity of DIAG does seem promising, especially considering that it generates all interactions with the learner automatically. Nevertheless, it is clear that the depth of technical explanations that DIAG can generate automatically does not rival that which can be crafted via a KE-based approach. In essence, there is no real limit to the technical depth that a KE approach can capture and deliver, while DIAG cannot explain technical issues that are deeper than its qualitative knowledge base.

We intend to explore ways to further enrich the technical content that can be supplied by the domain expert while interacting with DIAG. For example, DIAG might offer to the domain expert the option to explain fault effect specifications as they are entered. While this would undoubtedly increase the cost and time to develop an application, it would allow DIAG instruction to approach the richness offered by KE techniques, while avoiding the necessity to engage in knowledge engineering processes.

References

Forbus, K. D., & Whalley, P. B. (1994) Using qualitative physics to build articulate software for thermodynamics education, In *Proceedings of AAAI-94*, pp. 1175-1182.

Johnson, W. B., Norton, J. E., Duncan, P. E., & Hunt, R. M. (1988) *Development and demonstration of an intelligent tutoring system for technical training* (MITT) (AFHRL-TP-88-8). Brooks AFB, TX: The Air Force Human Resources Laboratory.

Kosko B. (1993) *Fuzzy thinking: The new science of fuzzy logic*. New York: Hyperion.

Lesgold, A., Eggan, G., Katz, S., & Rao, G. (1992) Possibilities for assessment using computer-based apprenticeship environments. In J. Regian & V. Shute (Eds.*) Cognitive approaches to automated instruction* (pp 49-80). Hillsdale, NJ: Erlbaum.

Lajoie, S. P. & Lesgold, A. (1992) Apprenticeship training in the workplace: Computer-coached practice environment as a new form of apprenticeship. In M. Farr and J. Psotka (Eds.), *Intelligent instruction by computer: Theory and practice*. London: Taylor & Francis.

Munro, A., Johnson, M. C, Surmon, D. S., & Wogulis, J. L. (1993) Attribute-centered simulation authoring for instruction. In *Proceedings of AI-ED'93*, 1993.

Towne, D. M. & Johnson, M. C. (1987) *Research on computer-aided design for maintainability*. (Technical Report No. 109) Los Angeles: Behavioral Technology Laboratories, University of Southern California.

Towne, D. M., Munro, A., Pizzini, Q. A., Surmon, D. S., & Wogulis, J. L. (1990) *Intelligent maintenance training technology*. (Technical Report No. 110) Los Angeles: Behavioral Technology Laboratories, University of Southern California.

Towne, D. M. & Munro, A. (1988) The intelligent maintenance training system. In J. Psotka, L.D. Massey, and S. A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned* (479-530). Hillsdale, NJ: Erlbaum.

Towne, D. M. (1994) Model-based simulations for instruction and learning. In *Proceedings of Delta '94 Conference: Telematics for education and training*. Dusseldorf, Germany.

Wiederholt, B. J., Norton, J. E., Johnson, W. B., & Browning, E. J. (1992) *MITT Writer and MITT Writer Advanced Development: Developing Authoring and Training Systems for Complex Technical Domains*, Final Technical Report AL-TR-1991-0122, Armstrong Laboratory, Brooks Air Force Base, TX.

Zadeh, L. & Kacprzyk, J. (1992) *Fuzzy logic for the management of uncertainty* (Eds.) New York: John Wiley & Sons, Inc.