N89-19824

DIAGNOSTICS IN THE EXTENDABLE INTEGRATED SUPPORT ENVIRONMENT (EISE)

James R. Brink, Ph.D.
Battelle Columbus Division
505 King Ave.
Columbus, Ohio 43201

Paul Storey
Sacramento Air Logistics Center
SM-ALC MMESD
McClellan AFB, CA 95652

ABSTRACT

EISE is an Air Force developed real-time computer network consisting of commercially available hardware and software components to support systems level integration, modifications and enhancements to weapons systems. The EISE approach offers substantial potential savings by eliminating unique support environments in favor of sharing common modules for the support of operational weapon systems.

An expert system is being developed that will help support diagnosing faults in this network. This is a multi-level, multi-expert diagnostic system which uses experiential knowledge relating symptoms to faults and also reasons from structural and functional models of the underlying physical model when experiential reasoning is inadequate. The individual expert systems are orchestrated by a supervisory reasoning controller, a meta-level reasoner which plans the sequence of reasoning steps to solve the given specific problem. The overall system, termed the Diagnostic Executive, accesses systems level performance checks and error reports, and issues remote test procedures to formulate and confirm fault hypotheses.

BACKGROUND

In general, once a weapon system has been operationally accepted and placed into the Air Force inventory, management responsibility for its support is transferred from the acquiring agency to one of the Air Logistics Centers (ALC) within the Air Force Logistics Command (AFLC). Aside from the classical logistics management functions, these centers also provide the engineering capability to do systems analysis, modifications of the hardware and software, component level testing and evaluation, and system level integration and test. The primary engineering tool for AFLC weapon system support is the Integration Support Facility (ISF).

A typical ISF, being a subset of the tools the contractor originally used to develop the system,is useful for supporting just the original system. Because modification is difficult, individual ISFs are developed to support the various models of the same weapon system. Like the weapon systems they support, ISF support is costly, requiring thousands of personnel and hundreds of millions of dollars annually for the Air Force.

To alleviate these problems, the Air Force is developing the Extendable Integration Support Environment, or EISE. The EISE concept consists of common hardware and software modules for common integration support functions. These modules, or building blocks, are logically reconfigurable to provide support for multiple weapon systems within the same environment. The EISE building blocks are off-the-shelf, commercially available items to the greatest extend possible. The building blocks are connected by Ethernet for non-real time requirements, and by a high-speed token passing network during real time simulations. As a result, custom ISFs for each weapon system will no longer be needed, thus reducing support costs through resource sharing, economies of scale for sparing of the individual building blocks, and for facility maintenance contracts.

OVERVIEW OF DIAGNOSTICS APPROACHES

Diagnosing problems associated with accommodating a great variability in the building blocks for EISE on a real time high speed network is expected to be quite difficult. Thus, a complex expert system, named the "Diagnostic Executive", is being developed to ensure the functioning and availability of EISEs. It is an "executive" because it performs high level diagnostics and calls upon diagnostics in the various processors as required to identify and isolate faults in EISE. In order to provide background on the development strategy for the Diagnostic Executive, we first summarize existing approaches to diagnostic expert systems.

The **conventional** approach to diagnostic expert systems development involves collecting and organizing the knowledge gained through experience by repair technicians, essentially associating a set of symptoms to the set of faults causing those symptoms. This approach is termed surface, shallow, experiential or empirical reasoning and it works well where human maintenance experts have accumulated enough experience to provide rules of thumb for most of the probable faults. Early experiential diagnostic systems developed as a flat knowledge base with every rule being scanned at every step of inference. In spite of some weaknesses, this

approach has a proven track record and is largely responsible for the success the AI technology in the mid-1980's.

Life-cycle maintenance and evolution of flat rule based systems has proven to be quite difficult for systems of even modest size. For **classification** problem solving, this problem is substantially reduced by adding control to the inference process (called "establish-refine") and modularzing the rule base via a taxonomy of pre-enumerated fault conditions, organized in a top down hierarchy. The most likely probable fault class is "established" and taken as a "hypothesis" to be "refined" into more finely detailed probable fault classes in the next lower level of the hierarchy, and this process is repeated recursively. Separate sets of rules are used for each class to relate probable fault classes to supporting evidence and to summarize results.

Both experiential and fault classification systems are shallow approaches, and represent condensed and streamlined knowledge which can yield accurate results. However, if a condition arises which is beyond its expertise (i.e., the pre-encoded human diagnostic rules), the system will perform poorly. CSRL developed at the Ohio State University (1983, 1986) and commercialized by Battelle (1986) represents an excellent example of this approach to classificatory problem solving. Rule Kit by General Dynamics (1984) is another example of this approach.

Structural based troubleshooting approaches incorporate information about the topology of the Unit Under Test (UUT). This includes connectivity between each of the components, similar to the schematic diagram of the circuit used by human technicians. Given a known good input and a known bad output, the system can use the connectivity to trace signals through the diagram. This is more flexible and robust than the shallow approach because failures can be diagnosed without pre-entering symptom-fault relationships. On schematics with many layers of logic, many stages of processing between input and output, large fan-in or fan-out, the ambiguity groups can be large, and combinatorial explosion problems often exist. By including with each component its failure probability, cost of access, test setup costs, required testing times and information yield, techniques can be employed to optimize for least test cost, least time to locate, least technical skill required, or least test equipment required. Cantone (1984) and Simpson (1982) are examples of this approach.

The next advancement in diagnostic reasoning approaches is to model the behavior and functionality of the components in addition to the structure. Behavior of a component can often be described with a set of rules, however, these rules come from the design engineer. Once a device has been modeled, it is generic and can be used for multiple purposes. This **"model based reasoning"** approach is sometimes termed "deep reasoning" when reasoning from physical first principles.

The model based reasoning strategy for diagnosis involves noting the differences between the expected outputs as predicted by the model and reality as measured at test points. Connectivity is used to find the components along the topological path of the signal. The dimension of the discrepancies (in analog circuitry, frequency, amplitude, phase, etc.) is used to narrow the search to the components that affect that dimension of the signal. This is done by pattern matching amongst the behavior rules of each component. The further screening of possible failed components by functionality greatly reduces the ambiguity groups, and therefore speeds up the search and reduces the number of tests required to isolate the fault. The use of behavior/functionality to determine which components affect a given measurement is the most important contributor to the diagnostic power of the model based reasoning approaches. Conversion of quantitative test measurements to terms suitable for qualitative reasoning (lo, OK, hi, always, sometimes, never), propagation of constraints, conflict resolution via tracking of assumptions and dependency chains in a truth maintenance system are some of the techniques employed in the model-based reasoning systems.

Model based systems, unlike heuristic or classification systems, can be robust without requiring exhaustive a priori enumeration of symptom-fault relationships. Using the structure and function of the UUT, symptoms and effects of failures can be dynamically computed. The disadvantages are that development times are longer, and execution time can become computationally intensive. Genesereth (1982), Sembugamoorthy (1984), Davis (1984), Pipitone (1986), de Kleer (1986), and Kaplan, et al (1988) are good examples of the model based reasoning approach.

THE DIAGNOSTIC EXECUTIVE

The approach used in the EISE Diagnostic Executive employs a mixture of the approaches described above. Experiential knowledge consisting of a priori failure probabilities and heuristic rules of thumb are tried first to locate the most common failures. This gives a quick response for problems within its range of expertise. When the diagnostic system encounters a failure mode beyond its surface heuristic rules, the deeper or model based reasoning system takes over, using its knowledge of structure (connectivity) and functionality (behavioral models) to search for components whose failure can explain the given symptoms.

Others who have used this approach to building diagnostic systems include Fink and Lusth (1986), Richardson and Barthelinghi (1986), Chu (1988), Pau (1986), Havlicsek (1986), McCown and Conway (1988), Warn (1988).

The types of faults expected to create the most problems for deployed EISE systems fall into three categories. First, the hardware components of EISE can fail. Because EISE involves mostly off-the-shelf commercial hardware, isolating hardware failures need only be accomplished to the vendor responsible unit, which might be a workstation (e.g., Sun, MicroVax) or a card (e.g., Heurikon).

Second, timing problems can occur during the real-time simulation phase, causing problems ranging from bad data to total system shutdown. Isolating these problems is expected to be especially difficult because they can arise from a variety of sources including hardware failures (intermittent or otherwise), errors in the network software, errors in the applications software or configuration errors. Third, faults can result from an incorrect setup or operation of the simulation itself. These faults can occur when an operator does not correctly follow the setup protocols, when an object code file does not get properly downloaded to a processor, or when the incorrect, or old version of object code is downloaded to a processor.

Many of these problems are highly interrelated and may have unpredictable side effects. Error messages resulting when an anomaly finally surfaces and becomes detectable by system checks may be highly unrelated to the cause of the problem. Thus, the diagnostic executive incorporates reasoning approaches to resolve the resulting ambiguities.

IMPLEMENTATION

The Diagnostic Executive is currently being implemented on a Symbolics 3675 using Intellicorp's suite of knowledge engineering tools. Diagnostics information from each layer of the ESIE network (see Figure 1) is routed to the Diagnostic Executive, which calls the appropriate diagnostic routines and repair actions. The planned structure of the Diagnostic Executive (shown in FIGURE 2.) is a multi-level, multi-expert diagnostic system which uses experiential knowledge relating symptoms to faults and also reasons from structural and functional models of the underlying system. The individual expert systems, termed Reasoners, are orchestrated by a supervisor termed the Reasoning Controller, a meta-level reasoner which plans the sequence of reasoning steps to solve the given specific problem. The Reasoners are integrated via highly structured common working memory managed by the truth maintenance system which keeps track of all relevant facts, deductions, hypotheses and chains of reasoning. A conclusion reached by any reasoner serves to constrain the space of possible causes of difficulty known as the ambiguity group. The constraints from all reasoners are summed in the Ambiguity Group Truth Maintenance System as they are determined. This stepwise summation of constraints, known as propagation of constraints, has the tremendous advantage of limiting the size of search space. The principle is that two simple constraints by separate reasoners can synergistically add to tremendously reduce the size of the search space a third reasoner must look through.

Coordinating the multiple expert systems employed in the Diagnostic Executive is the responsibility of the "Reasoning Controller", a knowledge base that contains information concerning which reasoning strategy is best to employ for each type of diagnostic problem. Upon malfunction, the Reasoning Controller is activated to determine the state of the network, what parts are functioning correctly, and the nature and extent of the problem.

The Reasoning Controller consists of a planner, agenda, scheduler and progress monitor. Using the refinement of skeletal plans technique, the planner matches relevant features of the problem, symptoms, and states of the network or operator requests, and chooses a sequence of applying the Reasoners that best fits the problem at hand. The sequence is placed in the agenda and executed by the scheduler. The progress monitor is a regularly executed watchdog process which reports information on the current known state of all nodes, processors and functions as available from the current ambiguity group, its current strategy, goals and deductions. In the future, it is expected to compare current progress against established norms and time constraints so that replanning can be directed to the planner when required. Explanations are also available in the form of dependency records and tracings of rule firings.

The typical control strategy is for the Reasoning Controller to first invoke the Event Reasoner to look at current system status data (including reported status of each node and error conditions), and to trace the operator command input history, forward chaining from this information to deduce the estimated status of all processors on the network. Next, the Reasoning Controller invokes the Surface Reasoner, to identify probable fault classes that can explain the observed symptoms (using its experential knowledge). Assuming the fault is not isolated by the Surface Reasoner, the Structural Reasoner is called upon to locate optimal test points in the system, given the malfunctions currently under diagnosis. The test points are chosen both to reduce the number of tests required to isolate a fault and to minimize the cost of performing the test. At this time, the Functional Reasoner can be used to determine expected values of intermediate test points from known good points or known bad points. Differences between the model predictions and the actual responses of the system constitute the symptoms to be used by the model based Reasoners.

The Event Reasoner is the primary diagnostic aid in beginning failure analysis of network startup from power-off condition to full-up, real-time condition. Major event sequences modeled include initialization of the individual computers, communication over a non-real-time network, the downloading of configuration files from servers to the diskless systems, communication over the real-time network, initialization of processors with real-time application data, handshaking and communication in real-time and post simulation activities.

The Event Reasoner uses models of temporal sequences to constrain the search to those portions of the system active during each action. Correct sequences of operations indicate processors and functions which have operated correctly. Thus the recognition of temporal sequences can serve as a landmark to indicate state of the system. The results of time sequences trigger rules to insert facts into the structural and functional dimensions of our ambiguity group truth maintenance system.

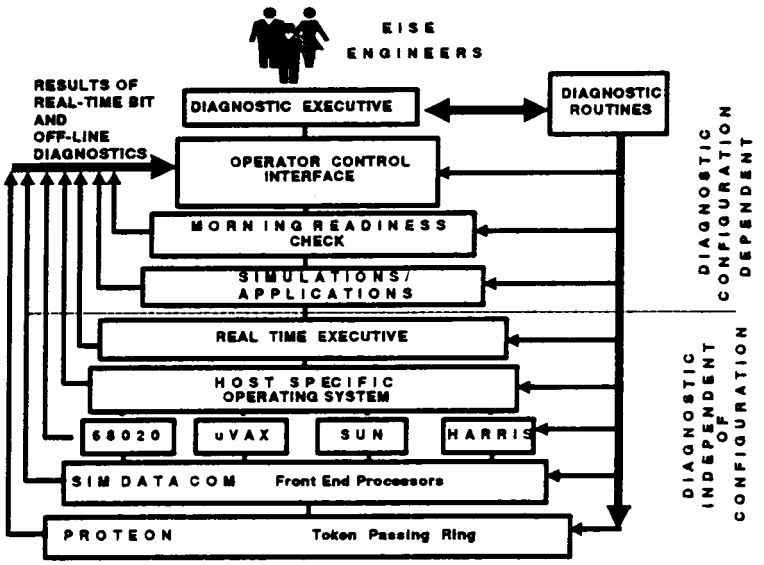

Figure 1. Overview of the Diagnostic Executive

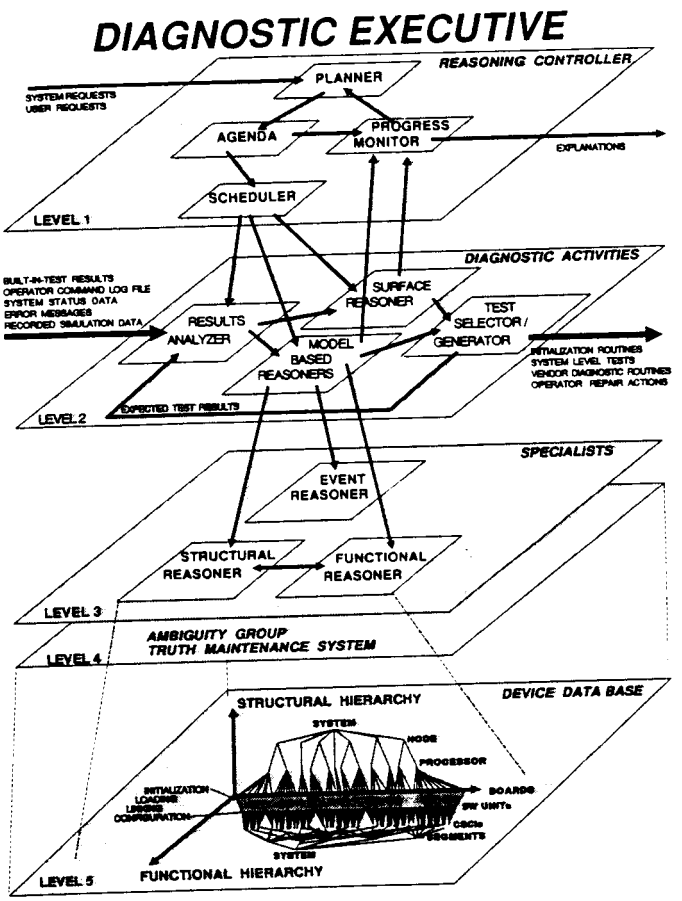

Figure 2. Architecture of the Diagnostic Executive

The Structural Reasoner is a topological based expert system as described in the Diagnostics Overview section. Each system component is represented as a unit, containing slots for source and destination of data paths. A connectivity tracer uses this information to find the paths from known good test points to known bad test points. After the paths have been found, feedback loops are marked to be opened, and calculation is done to find the point in the path that will yield the most information for the least cost. Included with each component is its failure probability, cost to access, test setup costs, required testing times and information yield. A computation is performed to optimize for least test cost, least time to locate, least technical skill required, or least test equipment required. The criteria to be optimized is passed down from the Reasoning Controller each time the Structural Reasoner is invoked. The result of this process is the optimal test point.

The Functional Reasoner is a model based reasoner as described in the Diagnostics Overview section. It models the transformations which occur as a signal is passed through the component. We have adapted the methodology of Pipitone (1986). A separate rule is used for every dimension of the signal. Qualitative reasoning (always, sometimes, never, low, OK, high) is used, not quantitative, numeric reasoning. All functionality rules are bi-directional. Computation of expected signal downstream from a known signal is done by forward chaining through any rules (always, sometimes, or never). Backward chaining can also be done from a measured failed testpoint to find components responsible for the erroneous behavior.

After the expected values have been computed, the test is run. Comparison of actual test results with expected test results yields a symptom. The dimensions of the symptom which differ from the predicted value is used by the Functional Reasoner to search for rules of each component along the test path that influence this dimension of the signal. This is done by matching on the functionality rules. Components along the signal path that contain rules that influence the dimension of the signal differing from prediction are the ambiguity set, the set of possible causes of the abnormality. Most of the diagnostic power of the Functional Reasoner comes from chaining on rules describing behavior. This narrows down the topological search to only those components that affect the behavior of the specific parameter under measurement.

The Ambiguity Group Truth Maintenance System constitutes Level 4 of Figure 2 and overlays the device data base. The deductions and corresponding rule firings are recorded in a set of worlds, or state/dependency graphs, managed by the truth maintenance system. Most of the intermediary conclusions made by the system are stored in these situation graphs or worlds. The truth maintenance systems adds, deletes, merges or invalidates worlds as information is confirmed, contradictions are found and new hypotheses are generated. Rule firings explicitly control this process. Facts not fitting into the worlds structure are stored in an unstructured facts list.

The Device Data Base constitutes Level 5 of Figure 2. This data base contains a structural hierarchy of EISE, from system to node, to individual processors on the node, to cards inside each processor. Boards are the replaceable unit in the EISE system, therefore the structural hierarchy only goes down to the board level. The device data base also contains a functional hierarchy of EISE.

As indicated above, the architecture described here represents our implementation plan. Implementation began early in 1988, and a working version of the Diagnostic Executive for the A-10 EISE will be ready for validation in late 1988. Currently, the Event Reasoner has received the most implementation attention and can handle many of the problems encountered. Portions of the other reasoners have been implemented, but integration of all reasoners as described through the reasoning controller is as yet incomplete.

CONCLUSIONS

The diagnostic executive described here is being built on the A-10 EISE, i.e., the EISE which serves as the Integration Support Facility for the A-10. Other Air Logistics Centers are expected to use EISEs to support their weapon systems. Extending EISE to other weapon systems will not only introduce other configurations and hardware components to diagnose, but operator setup protocols are expected to be more complex and timing problems will be more severe because of greater traffic on the network.

The Diagnostics Executive will substantially decrease the need to employ many high cost troubleshooting experts. In addition, faster and more thorough diagnostics will decrease downtime allowing greater utilization of existing valuable computer network resources. A third important benefit will be increased availability of the Integrated Support Facility allowing faster turnaround time to implement the needed, timely and highly responsive updates to our aircraft systems.

ACKNOWLEDGEMENTS

Mr. Steve Rinehart and Mr. Michael DeVaney of Battelle are preforming the detail design and implementation of the Diagnostic Executive. Most of the domain expertise is being supplied by the TRW Electronic Systems Group in Sacrament.

REFERENCES

Bylander, T. and Mittal, S., "CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling, AI MAGAZINE 7(3):66-77, 1986.

Cantone, Lander, & Gaynor. "IN-ATE/2: Interpreting High Level Fault Modes," in IEEE AUTOTESTCON, 1984.

Chandrasekaran, B., "Towards a Taxonomy of Problem Solving Types," AI MAGAZINE 4(1):9-17, 1983.

, S., "Approaches to Automatic Fault Diagnosis: A Critical Evaluation", AI IN ARMNAMENT WORKSHOP, American Institute of Aeronautics and Astronautics, 1988.

Davis, R., "Diagnostic Reasoning Based on Structure and Function", ARTIFICIAL INTELLIGENCE, Vol. 24, 1984.

de Kleer, J. & B. Williams, "Reasoning About Multiple Faults", AAAI 86 PROCEEDINGS, American Association of Artificial Intelligence, 1986.

Dudzinski, E., J. Brink, and D. Sharma, "CSRL-From Laboratory to Industry", EXPERT SYSTEMS IN GOVERNMENT SYMPOSIUM, 1986.

Fink, P. and J. Lusth, "A Second Generation Expert System for Diagnosis and Repair of Mechanical and Electrical Devices", in AI APPLICATIONS FOR INTEGRATED DIAGNOSTICS, University of Colorado, 1986.

General Dynamics Electronics Division, "Rule Kit: A Set of Tools for Building Rule Based Diagnostic Systems," 1984.

Genesereth, M, "Diagnosis Using Hierarchical Design Models", Proceedings of National Conference on AI, AAI, August, 1982.

Havlicsek, B, "A Knowledge Based Diagnostic System for Automatic Test Equipment", Artificial Intelligence in Maintenance, University of Colorado, 1985.

Kalpin, S., R. Schrag, and L. Volovik, "Performing Electronic Diagnostics With Distributed Expert Systems", AI in Armnament, American Institute of Aeronautics and Astronautics, 1988.

McCown, P. and T. Conway, "APU Maid: An Event-Based Model for Diagnosis", AI in Armnament Workshop, American Institute of Aeronautics and Astronautics, 1988.

Pau, L. F., "Survey of Expert Systems for Fault Detection", Test Generation and Maintenance, EXPERT SYSTEMS, Vol. 3, No. 2, April, 1986.

Pipitone, J., "The FIS Electronics Troubleshooting System", in Computer, 19 (7), pp. 68-76, Copyright 1986 by Institute of Electrical and Electronic Engineers, 1986.

Richardson, J. and G. Barthelenghi, "AI in Test Program Development, in AI Applications for Integrated Diagnostics", University of Colorado, 1986.

Sembugamoorthy, V. and Chandrasekaran, B., "Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems,", COGNITIVE SCIENCE, August, 1984.

Simpson, W. and H. Balaban, "The ARINC Research System Testability and Maintenance Program (STAMP)", IEEE AUTOTESTCON '82, Dayton, OH, 1982.

Warn, K., "Deep Reasoning Expert System for Armnament Diagnostics Applications, AI in Armnament Workshop", American Institute of Aeronautics and Astronautics,