N88-16366

# An Easy-to-use Diagnostic System Development Shell

L. C. Tsai, J. B. Ross*, C. Y. Han**, W. G. Wee
Department of Electrical And Computer Engineering
University of Cincinnati
Cincinnati, Ohio 45221

## ABSTRACT

This paper describes an expert system development shell for diagnostic systems, the Diagnostic System Development Shell (DSDS). The major objective of building the DSDS is to create a very easy-to-use and friendly environment for both types of users -- knowledge engineers and end-users. The DSDS is written in OPS5 and CommonLisp. It runs on a VAX/VMS system. A set of domain-independent, generalized rules is built in the DSDS, so the users need not be concerned about building the rules. The facts are explicitly represented in a unified format. These features make the DSDS very easy to use. A powerful check facility which helps the user to check the errors in the created knowledge bases is provided. A judgment facility and other useful facilities are also available. A diagnostic system (DS) developed based on the DSDS system is question driven and can call or be called by other knowledge-based systems written in OPS5 and CommonLisp. A prototype DS for diagnosing a Philips constant potential X-ray system has been built using the DSDS.

## 1. Introduction

Recently, expert systems have been applied to a variety of fields such as medicine, finance, engineering, and science [4,5]. The number of expert systems developed has doubled annually since the late seventies [2]. This rapid increment of the number of applications has led to the appearance of a wide range of commercially available expert system tools or so-called expert system development shells. Currently, these expert system tools are responsible for about 85% of fielded systems [3]. Most of the shells are powerful and fancy. But they are complex. Reference manuals with hundred of pages are not uncommon. It takes a relatively long time to learn them and to use them effectively. In addition, users are supposed to have some background in AI techniques and programming languages. In general, even with the shell available, developing an expert system is not a trivial task. For instance, in building of a medium size rule-based expert system hundreds of rules have to be created. Many issues such as the overall system structure and control schemes have to be considered by the user. Thus, it is desirable to create an environment that would take away the burden of learning a complicated development shell and creating rules that are related to the domain knowledge. In addition, this shell could be easily used by the system designer and end-users alike. With this objective in

* with the Aircraft Engine Business Group of General Electric Co., Evendale, Ohio.
** with Department of Computer Science, University of Cincinnati.

mind, the Diagnostic System Development Shell (DSDS) has been developed and is presented in the next sections.

Both the system architecture and its major components are presented in Section 2. A prototype diagnostic system (DS) developed by the DSDS is discussed in Section 3 and, finally, the concluding remarks are in Section 4.

## 2. The Diagnostic System Development Shell

The Diagnostic System Development Shell (DSDS) is a software utility that allows users, who know little about AI concepts, techniques, and any high level computer languages, to develop diagnostic systems and to use them.

Unlike the existing tools, there is a set of domain-independent, generalized rules in the DSDS. All the user needs to consider is the coding of domain knowledge (facts only) into knowledge bases. The facts, which are explicitly represented in lists with a unified format, are organized in nodes of decision tree structures. These features make both the DSDS easy to use and the knowledge base easy to build, modify, and expand. To monitor and to help users in building the DS a powerful check facility in the DSDS can pick up most of the possible mistakes the user might make. Since no AI techniques and programming skills are required, the domain experts can concentrate on coding the domain knowledge. A number of facilities, which include judgment, explanation, and help with graphics, will make the end-users feel the system is more friendly and convenient to run a DS developed by the DSDS.

### 2.1 Architecture of DSDS

The DSDS consists of eight major components. They include the generalized rule set (GRS), the fact knowledge base (FKB), and facilities such as the checker, the helper, the explainer, the judger, the DCL LISP interface and the graphic displayer. Among these components, the GRS, the FKB, and the checker are more important and unique. They will be discussed in the rest of this section.

The system architecture of the DSDS is shown in Figure 1. The built-in generalized rule set is in charge of reasoning. In the development mode of a DS, the user inputs domain knowledge into the FKB through an editor and invokes the checker to check the related LISP syntax and the node semantic rules.

During the execution of the DS, the system interacts with the user by generating questions to the user. The answers provided by the user will be checked by the judger which will identify and treat all the possible incorrect answers. The helper offers help with graphics when the user asks for. The DCL LISP interface makes it possible to use DCL under the OPS5 and COMMON-LISP environments. The graphic displayer are Fortran programs that displays graphics when help is provided. The explainer provides explanations of 'why' and 'how'.

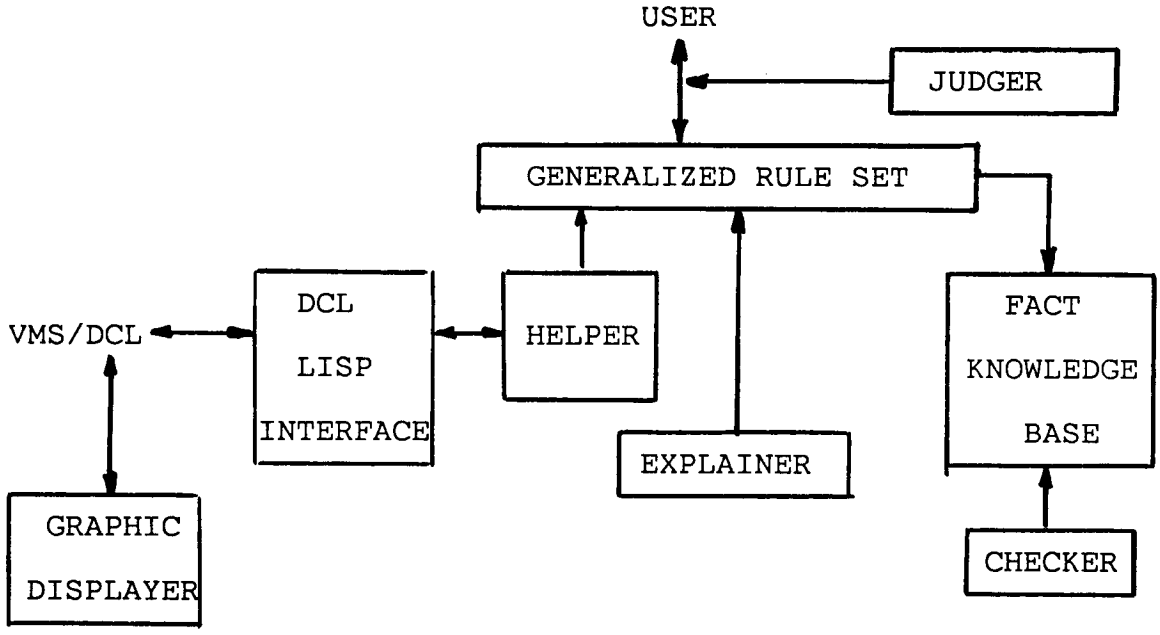


Figure 1. The Architecture of the DSDS system.

2.2 The Generalized Rule Set

The GRS contains the rules that will control the reasoning process while traversing down a decision tree during a diagnosis. The generalized rule set is based on two assumptions. One is that there is only a single source of error and the error is not transient. This assumption, as can be seen in most of the troubleshooting guides, is common in the field of diagnoses [1,7]. In fact, if there are more than one error source, the DS may be run several times to isolate all of them.

The other assumption is that the decision trees, on which the DS is based, are basically binary trees. The branches of every node of the decision trees represent two opposite propositions, yes and no, or positive and negative. This assumption fits diagnostic procedures well. The "car won't start" problem [6] is a typical example. Actually, the the binary tree structure may become a type of graph in which a node still has only two children, but different nodes could share a child, one node's children could be its parents or ancestors.

To illustrate the GRS, assume a simple, generic, binary decision tree of five nodes is given, in which the nodes N1 is the parent node of the nodes N2 and N3, and the node N3 is the parent node of the nodes N4 and N5. The nodes of the decision tree contain domain dependent facts, which will be described in the next section. Also, assume that in the node N1, a diagnostic test is being made and either answer yes or no is obtained. If yes, the node N2 will be pursued. In this case, N2 is a leaf that means a set of conclusions can be generated, problem related suggestions can be listed, and explanations be provided. If no, N3 is pursued and facts in node 3 will identify that this node is

not a leaf and a new series of actions such as generating intermediate status or results, continuing further testing, or other diagnostic actions are triggered. The process repeats for each node traversed until a leaf node of the decision tree is encountered.

## 2.3 The Fact Knowledge Base

Since the GRS is built in, the knowledge base includes facts only. So the knowledge bases in the DSDS are regarded as the fact knowledge bases. Facts, which are represented in lists, are organized in nodes of decision trees. Information related to a node is included only in the corresponding list, which consists of attribute-value pairs. All the lists are defined by a unified LISP format. There are three types of attributes: the necessary, the link, and the optional attributes. Each node list has all the necessary and link attributes and part of optional attributes. There are some syntax rules, which specify that the value types of certain attributes should be symbols or strings. There are also some semantic rules, which stipulate relations between attributes.

The FKBs are accessed by the generalized rules and transferred into OPS5 working memory. Unlike the generalized rules, the FKBs can be developed, modified and expanded by the user.

## 2.4 The Checker

Since the FKBs are defined by LISP functions and the users are supposed to know nothing about LISP. Some facilities should be available to avoid making mistakes. Two kinds of facilities could be helpful. The first is a special interactive interface which asks users questions, interpreters the answers and codes them into FKBs. But with this approach the users may easily get bored, especially all the facts are represented in the same way, the user will be asked the same questions repeatedly. The other way is to provide a check facility. Users can code the facts into FKBs directly through a text editor, then use the checker facility to check the developed FKBs. Since the facts are represented in a unified format, the second approach is more effective in the DSDS.

The checker is able to pick out related LISP syntax errors. It withdraws the decision trees from the built FKBs and displays them on the CRT. In addition, the checker can check the semantic rules. Once the checker picked out some error, it would print out error message, analyze the error and give the possible causes of the error. If a fatal error happens, the checker will stop working. Otherwise, it will continue to check until all the FKBs are checked.

## 3. An X-ray System Diagnostic System

A prototype -- an X-ray system DS was built by using the DSDS on VAX/VMS system. It is a DS for diagnosing a PHILIPS Constant Potential X-ray system. The X-ray system consists of two major units, the high voltage power supply and the computerized

controller. The controller contains nine printed circuit boards. The domain expertise came from the troubleshooting part of the X-ray system service manual and a domain expert. Based on this expertise twenty three decision trees corresponding to the same number of symptoms were built. The FKB includes 19 files with more than 5000 lines of code. The Checker was used to pick out all the syntax and semantic errors. To reduce the search space, this DS isolates the faults at four levels. The diagnostic procedure first finds out the fault is in the controller unit or in the high voltage power supply unit. Secondly, it isolates the faults in a certain board, then in a certain circuit and finally in a replaceable component.

## 4. Conclusion

An easy-to-use and friendly environment for designing diagnostic systems has been presented. The major features of the system are that the knowledge representation and input are facilitated by a unified list format and the reasoning of the expert system is done by the built-in generalized rule set. No previous knowledge of AI techniques and programming languages is required. The user needs only to concentrate on coding the domain knowledge into the so-called FKBs.

Overall the system is very straightforward to use. The prototype, explained in Section 3, has shown that the main objective of the DSDS system has been achieved. As a minor flaw of the implementation, the display of graphics is rather slow, since the process of generating graphics in VAX/VMS is complex. The OPS5 program has to call the display program written in Fortran via Lisp, DCL, and the display program has to get the data from disk.

## 5. References


[1] Davis, R. et al., "Diagnosis Based on Description of Structure and Function," Proc. of AAAI-82, 1982, pp.137-142.

[2] Gilmore, J. F. and Pulaski, K., "A Survey of Expert System Tools," Proc. of 2nd Conf. on AI Applications, 1985, pp.498-502

[3] Hamon, P. (ed.), "Inventory and Analysis of Existing Expert Systems," Expert System Strategies, Vol.2, No.8, Aug.,1986, pp.1-17.

[4] Hayes-Roth, F., Waterman, D. A., and Lenat, D. B. (eds.), Building Expert Systems, Addison-Wesley, Reading, MA, 1983.

[5] Waterman, D. A., A Guide to Expert Systems, Addison-Wesley, Reading, MA, 1985.

[6] Weiss, S. M. and Kulikowski, C. A., A Practical Guide of Designing Expert Systems, Rowman & Littlefield Pubs, 1984.

[7] A Constant Potential X-Ray System Model 161/321 Service Manual, Philips.