N90-27315

OBJECT ORIENTED FAULT DIAGNOSIS SYSTEM FOR SPACE SHUTTLE MAIN ENGINE REDLINES

John Rogers
Senior Research Associate

Saroj Kumar Mohapatra.
Graduate Research Assistant

Johnson Research Center
University Of Alabama In Huntsville

ABSTRACT

A great deal of attention has recently been given to Artificial Intelligence research in the area of computer aided diagnostics. Due to the dynamic and complex nature of space shuttle red-line parameters, a research effort is under way at the Johnson Research Center on the campus of UAH to develop a real time diagnostic tool that will employ historical and engineering rulebases as well as sensor validity checking. The capability of AI software development tools (KEE & G2) will be explored by applying object oriented programming techniques in accomplishing the diagnostic evaluation.

What is an Expert System?

In its simplest sense an "Expert System" is a system which can handle any domain limited situation as efficiently as a human expert would handle it. The keywords here are domain limited and human. Here is a simplified example. It might be easy to conceptually perceive a system that handles the breaking mechanism in an automobile. Now suppose these brakes are hydraulically activated as are some of the brakes in large trucks. If the system suddenly lost pump pressure due to some type of power system failure, the system should be expected to determine the element in its system (namely the pump) was malfunctioning, but probably would not be able to totally diagnosis the problem as a power failure. The power system is a separate entity in the automobile and therefore does not fall into the hydraulic systems limited domain. Theoretically the domain of an expert system may be as large as the designer desires; however, the larger the system domain, the larger the knowledge base requirements and the more complex the inference engine.

The second keyword, human, details the performance requirements of the system. The system must be able to handle a situation that exists within its limited domain as efficiently as a well trained expert in the field. Though it can not be has not yet begun to reach its size or pattern matching potential. Also, since the system is currently operating as a stand-alone system and no sensor input is available, simulation and data file supplied values are driving the system. Output is restricted to operator notification and suggested corrective procedures by the system. Scan Intervals are well within one second, yet far from the ultimate goal. These limitations are being slowly reduced and eventually it is hoped they will be removed entirely.

expected to determine the faulty power supply, a properly designed expert system should immediately identify the pump as the source of the localized problem. Of course, the system is not doing anything that a human can not do. But for large problems that involve many parameters the system, if properly designed, can reach the conclusion much faster and free the human expert to be accomplishing other tasks.

CREATING AN EXPERT SYSTEM

An expert system has three major modules. These are the systems interface with its environment, the knowledge base of rules and information governing the system, and the user interface.

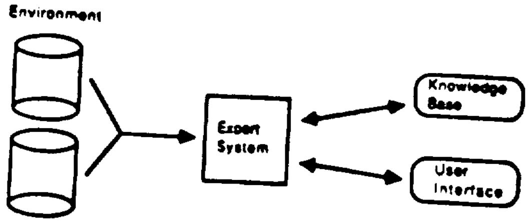


Figure 1

Each of these modules plays an important role in the operational efficiency of the overall system. If the interface between the environment and the system is inadequate the system will be unable to properly monitor the environment's condition. If the knowledge base is not complete or contains erroneous information the system will reach improper conclusions. And finally the system must be designed so it is easy for the individual to use. Each of these topics will now be presented.

Interface between the System and the Environment

Depending upon the type of expert system under development the interface between the environment and the system may take on different forms. For example, if the system is being designed to maintain certain operating parameters in a particular piece of computer software, the interface may include a communications protocal between the two machines (if they are in fact located on separate machines). Or possibly for a mechanical system, the interface between the mechanism and the expert system monitoring it may be a connection to an analog converter that takes in raw sensor voltages and converts it into meaningful data. Any combination of these forms is possible. In different systems throughout industry a variety of interfaces have been designed to fit the needs of the application.

Regardless of the type of interface, the primary objective is to have as complete a communication between the environment and the system as possible. Erroneous results would be produced by a system that has the potential of functioning properly due to simply bad input data What- ever needs to be included in the interface to insure this communication clarity must be included. Obviously this is a critical necessity for the system and must be satisfied in order for it to monitor the environment correctly.

It is the job of the knowledge engineer to acquire a complete set of facts and rules that govern the environment. This is accomplished in a variety of ways. First, many processes follow a given set of engineering and natural laws. The applicable formulations of these laws can be incorporated into the system. Previous or historical data can be interpreted and stored. And learned and experienced individuals can be interviewed to gain from their knowledge.

The interview method is the most difficult to perform, but often the most beneficial. It is easy to see how valuable it is to have experience with a certain type of problem. Many times a process can be completed rather routinely by an individual, but it is time consuming and takes the individual away from other much needed tasks. This is a perfect example of where the interview method would readily apply. However, the familiarity with the problem can become a mixed blessing. Crucial steps in the performance of a task become routine to the expert and might be omitted in the performance description. Therefore leaving the system designer with an inadequate instruction set.

Knowledge Base

Perhaps the most difficult task facing the developer of an expert system is constructing the knowledge base. The knowledge base is a collection of facts and associated rules (sometimes called the rule base) that are connected with the environment of the expert system. These rules and facts are based on information gathered by a process called Knowledge Engineering.

Most tasks that humans perform are complex in nature, yet they seem quite simple to the individual. For example, consider the everyday activities that an ordinary individual might find themself doing such as brushing their teeth, getting dressed, or preparing a meal. Certainly most of us know how to perform each of these simple tasks quickly and easily; however, it is not quite as simple a problem to describe or explain all of the steps that are involved. If one of the steps is omitted the total process is likely to fail. This knowledge engineer must familiarize himself with the envirnment that is being monitored. Only then can a successful dialog between himself and the experts occur. It is his responsibility to question these experts in a manner such as to gain a complete set of facts and rules governing the specific environment.

User Interface

The user interface is the individuals gateway to the expert system. One of the most often heard complaint about any software package is the inadequacy of the user interface. A user interface should be as simple to use as possible, yet have the flexibility to allow the operator the ability to handle virtually any situation.

There are many different formats for the user interface. It can be menu driven or allow for command input lines. It might include mouse sensitive items, help documentation, and meaningful error messages for the user. Any or all of these items can help to make a quality user interface. However, remember the most important aspect is usability.

Current Application - S.S.M.E. Redline Diagnostics

The current application involves the monitoring and diagnostic analysis of a group of flight parameters for the Space Shuttle's main engine known as the redlines. These parameters include the High Pressure Oxidizer Preburner discharge intermediate seal purge pressure, the High Pressure Oxidizer Turbopump turbine discharge temperature and secondary seal cavity pressure, the High Pressure Fuel Turbopump turbine discharge pressure, High Pressure Fuel Preburner coolant liner pressure, and the Preburner shutdown purge pressure.

This system is divided into three subsystems: a monitoring system, a rule based diagnostic system, and a historical data base of past performance. Each plays an important role in emulating an expert in the identification and evaluation of possible operating problems.

The monitoring system is similar to those in place in many applications around the globe. It receives sensor data that it compares to a given set of parameter limits. If the data falls within the acceptable range for each of the redline parameters, then opeations continue unaffected. However, if the data does not remain confined between the operational limits, flags are set which indicate a problem has occurred. At the completion of a given cycle of data entry, these flags are examined. If they indicate possible problems the diagnosis procedure is initiated.

The diagostic system is a rule based system that forward chains or background chains through a set of rules and hopefully arrives at a possible conclusion for the situation. For the SSME redlines these rules consist of engineering and interparameter relationships which must be maintained. As a conclusion the system may either initiate an attempt to correct the situation and/or report the probable causes and siggested ways to proceed to the operator.

The diagnostic system should also have a built-in mechanism for validity checking on data. This is accomplished in this system in two ways. The rule base has certain rules which define physical laws. If according to these rules all but one dependant parameters abide by these laws or if data trends for parameters like pressure and temperature exhibit sporadic almost non-continuous behaviour then there existsd a high probability that the sensors are not functioning properly.

The third sub-system involves pattern matching current data to past experiences. This gives the system a type of historical prospective on the current events and may indicate the cause of the current situation based on the causes of the similar historical event. This sub-system is initiated only if the diagnostic subsystem was unable to arrive at a plausable conclusion. If initiated, however, the current events are compared to a data base of past events and a ranked list of possible situations along with a weight factor that describes the degree of parameter matching is returned.

Prototype Development

The original system was prototyped using a software development tool named KEE (Knowledge Engineering Environment) which is a product by Intellicorp. It was menu driven and used graphical displays to relate the information to the user. KEE proved to be a good starting place for the early development. However, time is of primary importance in this system. Each scan interval for the system is desired to be less than 60 microseconds. Therefore, an alternate method had to be explored. G2, a real-time expert system, software development tool developed by Gensym, was selected for the second development.

What is G2

G2 is an advanced tool for developing real time expert systems. Real time expert systems are programs that can respond intelligently to events as they occur. The need for such systems is evident in the process and manufacturing evident in the process and manufacturing industries, telecommunations, medical care, aerospace, robotics, and so on. At the heart of G2 is its ability to do the following:

.G2 can scan an application (like an human operator) and can focus on key areas when it detects potential problems or oppertunities.

.G2 can control events in a continuously changing application. This is possible because of a real time inference engine that can maintain validity intervals and history for variables.

.G2 can respond to events when they occur (without having to continually poll sensors, for example.)

.G2 allows the application experts to create prototypes rapidly using structured natural language in an intuitive, graphically oriented environment.

Idea of developing a expert system using G2

To use G2, the developer first describes each class of object in the application, what it looks like, and what its attributes are. After describing the classes of objects that are found in the application, the developer can create a model of the application by placing objects on a workspace and connecting them to show their relationships. Associated with each object is a table that describes the object. G2 automatically creates this description from the definition of the object class. After drawing the schematic, the developer writes rules that indicate how to respond and what to conclude from changing conditions within the application. The developer enters these rules and all other statements within G2, in structured, natural language using a context-sensitive editor that guides him through each part of the grammar.

When the knowledgebase is running, G2's real time inference engine uses the rules, together with data that it receives from the data server (the sensor, G2 simulator, and other external sources) to infer how to respond to conditions.

The data server attribute indicates where G2 should go to get values for the variables. The inference engine scans key rules at rates associated with each rule. It focuses on key objects by trying rules associated with the objects. The inference engine invokes rules of a particular category for a particular class of objects. It backward chains to other rules to find values and forward chains to rules when values are found.

The knowledgebase uses the G2 simulator to simulate values for sensors and other variables while the knowledge base is running. It can solve algebraic difference and first order differential equations.

The developer also can create operator controls like check boxes, radio buttons, and action buttons that an operator will be able to use to enter values or to give instructions to G2. The developer can create displays like graphs, dials, meters or readout tables.

The development of a knowledge base usually proceeds incrementally. To develop a full-sized application, he proceeds in stages, at each state refining the prototype, adding a little more to the knowledge base and testing the result. After the knowledge base is built, the developer can interface to an application using "Gensym Standard Interface." The expert system can then receive values from sensors or other sources, set the values of setpoints, write to databases, and so on.

Example of - G2 Redline Implementation

The project required implementing eight parameters: Preburners S.D., HPOP (High pressure oxidizer pump), HPOT (High pressure oxidizer turbopump), HPFT (High pressure fuel turbopump), HPFP (High presusre fuel pump), LPOP (low pressure oxidizer turbopump), LPFP (Low pressure fuel turbopump) and POGO. After creating a workspace those redline parameters are implemented using different kinds of objects(Figure 1). A superior class object is defined so that redline parameters can inherit its attributes whenever they need.

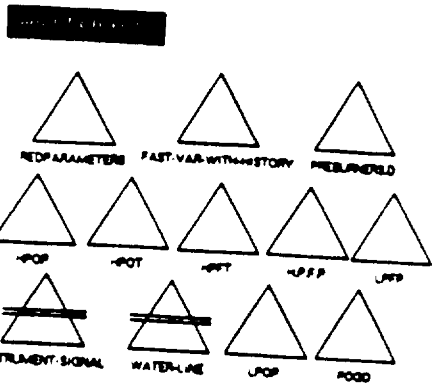
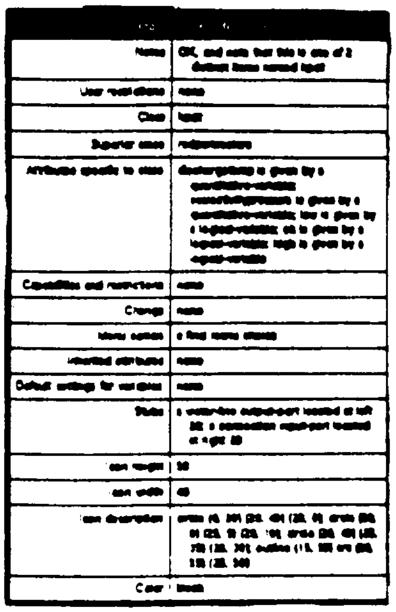


Figure 2

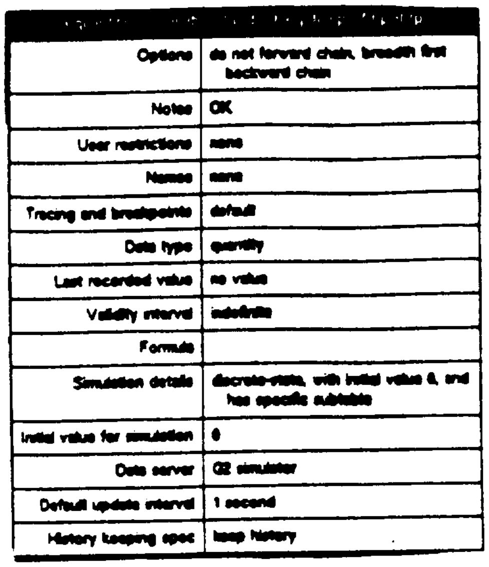
G2 has the capabilities to create tables (Figure 3) for each object For example in Figure 3, HPOT is the object and the table has user restrictions, where the user can describe the restrictions needed for HPOT;.HPOT is the class for which also the superior class is redline parameters. Attributes specific to.the class HPOT are discharge-temp (a quantitative variable), sealactivity pressure (a quantitative variable), OK (a logical variable), and high (a logical variable). The capabilities and restrictions of HPOT are "none", change is "none", and menu option is "final menu choice." In this case no attributes are inherited, so inherited attributes get the value "none." Default settings for variables are also "none." Notice the attribute stubs in Figure 3. There is a water line output-port located at left 20; a connection input-port located at right 20. To use the connection between the instances of objects the user has to define an object called water-line or instrument line. So as shown on Figure 3 the instrument signal and water-line are defined. The user can define an instance out of the objects. The last four attributes describe the icon which is used to display the instance of the object HPOT and can modified to the designer's specifications.



Notes	[illegible]
User restrictions	none
Class	hpot
Superior class	redline-parameters
Attributes specific to class	discharge-temp is given by a quantitative-variable; sealactivity-pressure is given by a quantitative-variable; low is given by a logical-variable; ok is given by a logical-variable; high is given by a logical-variable
Capabilities and restrictions	none
Change	none
Menu option	a final menu choice
Inherited attributes	none
Default settings for variables	none
Stubs	a water-line output-port located at left 20; a connection input-port located at right 20
Icon height	[illegible]
Icon width	[illegible]
Icon description	[illegible]
Color	[illegible]

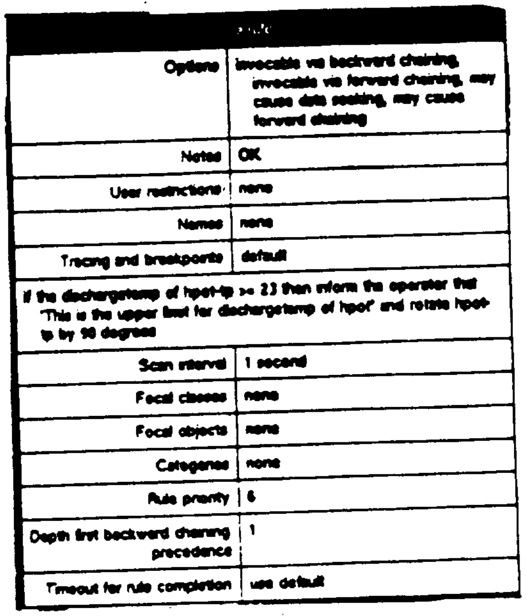
Figure 3

After creating the instances of objects, the user can see the table for that instance by selecting it with the mouse, that is automatically created by G2 like Figure 4. For each attribute in the table, simulation details can be written on a specific subtable. The use can get a value for any attribute using the G2 simulator, the inference engine, or from outside files and sensors. After considering where to get values for all the attributes of the instances, the user can write rules on the workspaces attached to object definitions, the instances, and G2 itself like the example given below for HPOT. Each rule has its own table and G2 scans keyrules at rates associated with each rule. For example in the figure 4.1, the scan interval is one second. The scan interval provides a way to regularly invoke a rule to monitor an event. The value of the scan interval tells G2 how often to invoke the rule. The user can follow rule firing path since everytime G2 fires a rule using forward chaining or backward chaining it highlights that rule.



[illegible]	
Options	do not forward chain, breadth first backward chain
Notes	OK
User restrictions	none
Names	none
Tracing and breakpoints	default
Data type	quantity
Last recorded value	no value
Validity interval	indefinite
Formula	
Simulation details	discrete-state, with initial value 6, and has specific subtable
Initial value for simulation	6
Data server	G2 simulator
Default update interval	1 second
History keeping spec	keep history

Figure 4



[illegible]	
Options	invocable via backward chaining, invocable via forward chaining, may cause data seeking, may cause forward chaining
Notes	OK
User restrictions	none
Names	none
Tracing and breakpoints	default
if the dischargetemp of hpot-tp >= 23 then inform the operator that "This is the upper limit for dischargetemp of hpot" and rotate hpot-tp by 90 degrees	
Scan interval	1 second
Focal classes	none
Focal objects	none
Categories	none
Rule priority	6
Depth first backward chaining precedence	1
Timeout for rule completion	use default

Figure 4.1

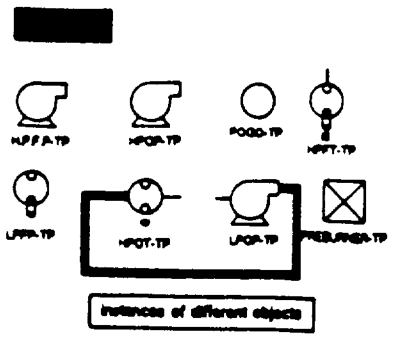


Figure 4.2

<u>**Implementation of Intelligent Fault Disgnosis Using G2**</u>

As described before to create any instance, an object should be defined. So for the S.S.M.E project in order to describe the project and to create icons which will have its own workspaces, the utility icon, the documentatio-details and KB -details are defined (Figure 5). An icon library is created as shown on the Figure 6, so that it will help the user to extend the system whenever he wants. As shown on the workspace (definitions) on Figure 7, the icons are defined from documentation-detail and KB-detail objects. On the subworkspace of "about-objects" we have successfully implemented eight redline parameters (Figure 2). Then as described before the instances for all these paramenters are created by defining the attributes on the subtables attached to them (Figure 3). To give a nice idea about the instances of objects, a new workspace is created and all the instances of redline parameters objects are defined on that workspace (Figure 4.2).Values for those instances of the objects are obtained using G2 simulator, inference engine and data files simulating sensors..As descirbed before, each instance has its own subtable, which has its own specific subtable where we defined our simulation formula as the abstract shown on Figure 8 for HPOT redline parameter.

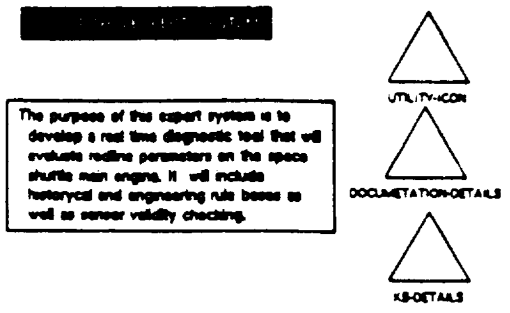

The purpose of the expert system is to develop a real time diagnostic tool that will evaluate redline parameters on the space shuttle main engine. It will include historycal and engineering rule bases as well as sensor validity checking.

Figure 5

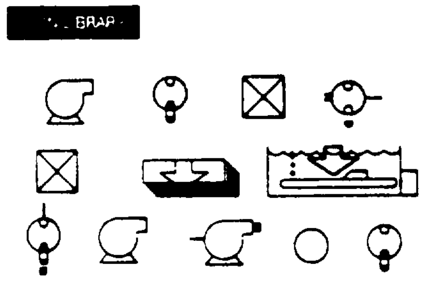


Figure 6

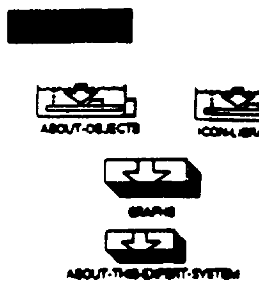
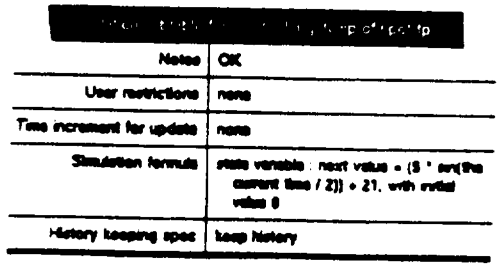


Figure. 7

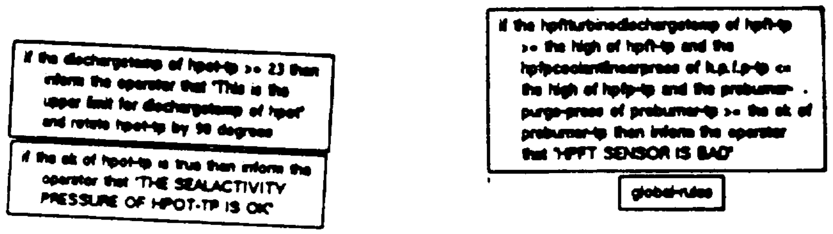


[illegible] temp of hpot-tp	
Notes	OK
User restrictions	none
Time increment for update	none
Simulation formula	state variable : next value = (5 * sin(the current time / 2)) + 21, with initial value 0
History keeping spec	keep history

Figure 8

After making sure that all the redline parameters are getting values, rules are implemented. The rule groups are devided into three parts: local rules, global rules and chained rules. Local rules are implemented on the workspace attached to object definitions. Global rules relate two or me things at a time and can be fired (Figure 9). Chained rules first try to get a conclusion from global or local rules and then try to find another solution for that conclusion (Figure 10).

The rule base is traversed after each intermediate conclusion is reached until all possible paths have been taken. Sometimes logical variables are used for some redline parameters like low, OK, and high. The low, OK, and high variables have values attached with them. These are the limitations for a particular instance. For example if the attribute dischargetemp of HPOT is greater than high or low or OK then after G2 fires the rule the operator can get the message. We have implemented graphs as shown on Figure 11 to display the result of realtime systems by keeping a history.



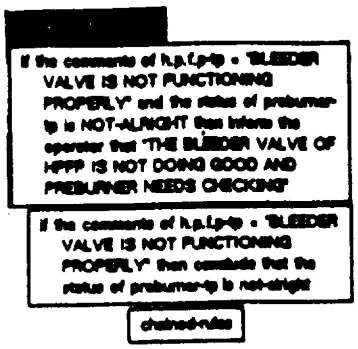
if the dischargetemp of hpot-tp >= 23 then inform the operator that "This is the upper limit for dischargetemp of hpot" and rotate hpot-tp by 90 degrees

if the ok of hpot-tp is true then inform the operator that "THE SEALACTIVITY PRESSURE OF HPOT-TP IS OK"

if the hpftturbinedischargetemp of hpft-tp >= the high of hpft-tp and the hpfpcoolantlinearpress of h.p.f.p-tp <= the high of hpfp-tp and the preburner-purge-press of preburner-tp >= the ok of preburner-tp then inform the operator that "HPFT SENSOR IS BAD"

global-rules

Figure 9



If the comments of h.p.f.p-tp = 'BLEEDER VALVE IS NOT FUNCTIONING PROPERLY' and the status of preburner-tp is NOT-ALRIGHT then inform the operator that 'THE BLEEDER VALVE OF HPFP IS NOT DOING GOOD AND PREBURNER NEEDS CHECKING'

If the comments of h.p.f.p-tp = 'BLEEDER VALVE IS NOT FUNCTIONING PROPERLY' then conclude that the status of preburner-tp is not-alright

chained-rules

Figure 10

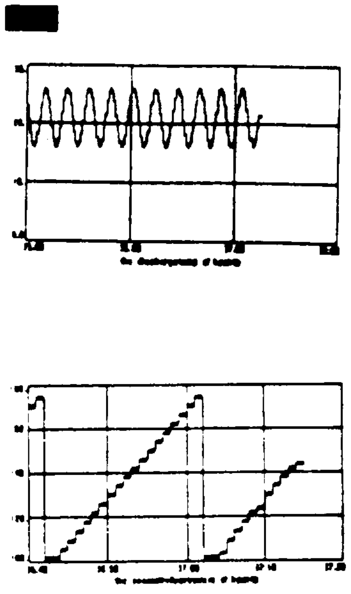
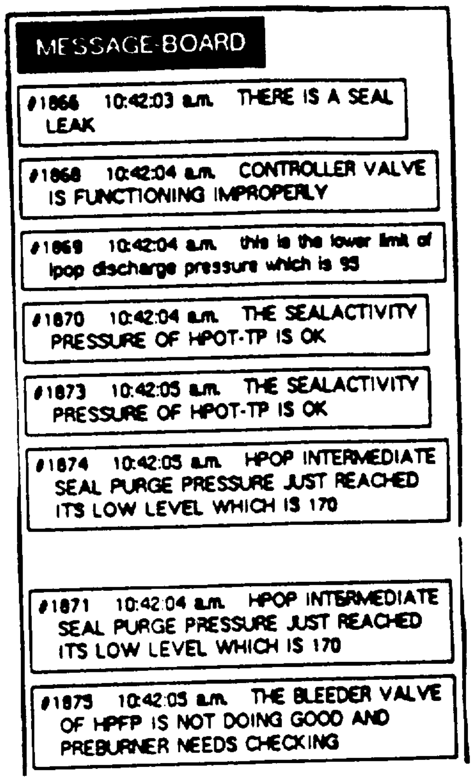


Figure 11

Through the message board of G2 the diagnostic system communicates to the operator. When the diagnostic system fires the rules different kinds of messages appear on the message board as shown on the Figure below



```
MESSAGE-BOARD

#1866   10:42:03 a.m.   THERE IS A SEAL
  LEAK

#1868   10:42:04 a.m.   CONTROLLER VALVE
  IS FUNCTIONING IMPROPERLY

#1869   10:42:04 a.m.   this is the lower limit of
  lpop discharge pressure which is 95

#1870   10:42:04 a.m.   THE SEALACTIVITY
  PRESSURE OF HPOT-TP IS OK

#1873   10:42:05 a.m.   THE SEALACTIVITY
  PRESSURE OF HPOT-TP IS OK

#1874   10:42:05 a.m.   HPOP INTERMEDIATE
  SEAL PURGE PRESSURE JUST REACHED
  ITS LOW LEVEL WHICH IS 170

#1871   10:42:04 a.m.   HPOP INTERMEDIATE
  SEAL PURGE PRESSURE JUST REACHED
  ITS LOW LEVEL WHICH IS 170

#1875   10:42:05 a.m.   THE BLEEDER VALVE
  OF HPFP IS NOT DOING GOOD AND
  PREBURNER NEEDS CHECKING
```

Figure 12

Currently the system contains approximately twenty-five rules which attempt to bind the eight redline parameters. Historical data has been difficult to obtain and consequently that data base has not yet begun to reach its size or pattern matching potential. Also, since the system is currently operating as a stand-alone system and no sensor input is available, simulation and data file supplied values are driving the system. Output is restricted to operator notification and suggested corrective procedures by the system. Scan Intervals are well within one second, yet far from the ultimate goal. These limitations are being slowly reduced and eventually it is hoped they will be removed entirely.

Closing Note

The S. S. M. E. real-line diagnostic system is a good start toward a viable tool. Currently, even using G2, the system is much too slow and the rule and historical data bases are too limited for it to ever be considered for an online control system. However, the system does show potential as an analysis aid for engineers working with the S. S. M. E.

The next phase of the systems development will address these problems by increasing the number of historical database entries as well as continued exploration into methods of speeding up the programs execution. The primary objective of the project is to prove the feasibility of the concept of an extended diagnostic system to S. S. M. E. In this respect the program is succeeding.

REFERENCES

1. Ali, M.,Schrnhorst, D.A."Sensor-based Fault Diagnosis in a Flight Expert System." Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, FL, December 11-13, 1985.

2. Firebaugh, Morris W.; Artificial Intelligence, "A Knowledge Based Approach." Published by PWS-Kent, Boston.

3. Fox, Mark S., Lowenfeld, Simon and Kleinosky, Pamela "Techniques For Sensor-based Diagnosis," Carnegie-Mellon University, The Robotics Institute Technical Report, 1983.

4. "G2 user's manual" Version 1.1, 1988

5. Harmon, Paul and King,David "Expert Systems," Artificial Intelligence in Business; edited by Theron Shrene, 1985.

6. Palmer, Michael T., Abbott,Kathy H., Schutte,Paul C. and Ricks, Wendell R., "Implementation of a Research Prototype On-board Fault Monitoring and Diagnosis System," AIAA Conference of Computers in Aerospace, Massachusetts, 1987.

7. Scharnhorst, Dean A."On the role of Artificial Intelligence In SSS Computer- aided diagnosis ", The University Of Tennessee Space Institute, Knowledge Engineering Laboratory