NASA Contractor Report 187113

TROUBLE III: A Fault Diagnostic Expert System for Space Station Freedom's Power System

David B. Manner
Sverdrup Technology Inc.
Lewis Research Center Group
Brook Park, Ohio

Prepared for
Lewis Research Center
Under Contract NAS3-25266

NASA

TROUBLE III:
A FAULT DIAGNOSTIC EXPERT SYSTEM
FOR
SPACE STATION FREEDOM'S POWER SYSTEM

David B. Manner
Sverdrup Technology, Inc.
NASA Lewis Research Center Group
Cleveland, OH 44135

ABSTRACT

Designing Space Station Freedom has given NASA many opportunities to develop expert systems that automate onboard operations of space based systems. This thesis describes one such development, TROUBLE III, an expert system that has been designed to automate the fault diagnostics of Space Station Freedom's electric power system. TROUBLE III's design is complicated by the fact that Space Station Freedom's power system is still evolving and changing. TROUBLE III had to be made flexible enough to handle system changes with minimal changes to the program. Three types of expert systems were studied: rule-based, set-covering and model-based. A set-covering approach was selected for TROUBLE III because it offered the needed flexibility that was missing from the other approaches. With this flexibility, TROUBLE III is not limited to Space Station Freedom applications, it can easily be adapted to handle any diagnostic system.

INTRODUCTION

As space based systems become larger and more technically advanced, the need for automating the monitoring of these systems becomes more apparent. Space Station Freedom has many areas where automation can be applied that will reduce the number of personnel needed, increase reliability and speed up response time to faults. One method of automating system monitoring is to use expert systems.

This thesis describes TROUBLE III which is a failure cause diagnostic expert system designed to detect and diagnose failures that have occurred in Space Station Freedom's power system. TROUBLE III is currently under

development at NASA Lewis Research Center. It was written using the commercially available AI shell ART (Automated Reasoning Tool) from Inference Corporation and resides on a Texas Instruments Explorer II workstation.

Since September 1984, the design of the electric power system of Space Station Freedom has changed several times. Design issues include:

- Will generation be solely photovoltaic (DC) or will it be a combination of photovoltaic (DC) and solar dynamic (AC)?
- Will distribution be 20 KHZ 440V AC, 400 HZ 440V AC or 160V DC?

Since the design of the power system is still evolving and changing, TROUBLE III must be flexible enough to handle drastic design changes with minimal changes to the program.

To accomplish this flexibility, TROUBLE III uses a set-covering technique instead of the traditional rule-based design. In set-covering, failure knowledge about the system is stored in a database instead of hard coded within the rules. TROUBLE III's rules are used to match the detected symptoms to this stored failure knowledge generating a list of failure hypotheses and the possible causes for each. This list of failure hypotheses is then ranked according to which set of mode-cause pairs are most likely responsible for the observed condition of the system.

Failure knowledge is obtained by doing a Failure Mode and Effects Analysis (FMEA) on each component of the power system. Each unique failure mode - cause pair identified during the FMEA is stored as an object in the failure knowledge database. As Space Station Freedom's power system evolves and changes, the rule logic of TROUBLE III is unaffected with only the failure knowledge needing alteration.

TROUBLE III has been designed to handle fault diagnostics of an evolving power system. When Space Station Freedom's power system design is finalized, TROUBLE III will be quickly adapted to it. This will allow TROUBLE III to provide the autonomous monitoring of a power system needed by a space-based system like Space Station Freedom.

DIAGNOSTIC EXPERT SYSTEMS

Many systems, especially electrical power systems, have human operators that continuously monitor the system and diagnose the system's failures.

This task becomes increasingly difficult as the system becomes more complicated making the idea of automating diagnostics with expert systems highly appealing. Expert systems are computer programs which emulate the knowledge and reasoning capabilities of a human expert. One requirement for implementing a successful expert system is to have human experts available who can solve the problems in their expertise domain. Since humans already have workable solutions for diagnostic tasks in areas such as power systems, chemical processes and manufacturing systems, failure diagnosing large systems is an ideal application for expert systems.

Diagnostic expert systems accept quantitative measurements about the current state of the system as inputs, change this data into qualitative knowledge, determine if any failures exist and hypothesize the cause(s) for those failures. The data acquisition task involves sampling raw measurements from appropriate equipment located throughout the system and converting these measurements to appropriate engineering units. Quantitative state data alone carries no notion of whether or not the system is performing within its tolerances. Knowing that a device has an output of 115 volts is not very useful to the expert system. This type of data needs to be compared to operating limits to obtain a qualitative statement like "device is operating above operating limits". This statement is more useful to the expert system when it begins reasoning about the current state of the system. In its most rudimentary form, qualitative knowledge is a single threshold alarm - something is on or off. At a higher level, it allows for classifying equipment operating performances as being abnormally low, within tolerance, abnormally high, etc. As abnormal operations are observed, an expert system must be able to diagnose what events caused these abnormalities. To do this, the expert system generates hypotheses about the cause of the malfunctions. There may be several hypotheses to explain a single malfunction. Ideally, one wants to selectively generate hypotheses in a way that limits the number of hypotheses considered but, at the same time, will generate enough hypotheses to identify the most likely hypothesis[15]. An expert system can help the human operator(s) determine the probable cause for the detected failures by identifying all the possible causes for each failure. The goal of the diagnostic process is to generate a diagnosis which can "explain" all of the observations, especially the deviant ones.

Design Techniques

Design techniques used in diagnostic expert systems usually fall into one of three classifications: Rule-based systems, Model-based systems or Set-covering systems. Rule-based systems are the most common and are based on experiential knowledge encoded in specific if-then rules. Model-based systems contain a simulation of expected behavior. System performance is

compared to expected performance to detect anomalies. Set-covering systems are based on the failure knowledge being pre-defined and stored in a database with general rules matching symptoms to failure data[11].

Rule-Based Systems

Rule-based systems are the best known and most widely used type of expert systems. The knowledge of an expert in the problem domain is used to describe possible malfunctions of the system and what conditions must exist for that malfunction to be present in the system. A series of if-then rules are then developed that link the current system's conditions to the conditions described by the expert. Each possible fault or malfunction of the system has its own rule. If the right conditions exist, the rule concludes an appropriate failure cause.

All the failure knowledge about the system and the reasoning ability of the program is hard-coded into the rules. This makes rule-based systems very domain specific. Seldom can they be generalized for use on a similar problem. If the system or failure knowledge changes, the programmer must change all rules affected by these changes as well as adding new rules to cover any new situations.

Rule-based systems are said to contain shallow or experiential knowledge about how the system responds to certain conditions rather than deeper knowledge about how the system actually functions. Since rule-based systems are based on the knowledge of experts, their reasoning ability is limited in that they can only detect malfunctions that the experts are aware of. They cannot make inferences.

Model-Based Systems

In a model-based system the experts develop a system model which predicts the expected behavior for the system and compare this to the actual behavior. A fault is considered to be any discrepancy from the modeled behavior. From the discrepancies between predicted and actual system behavior, model-based programs can reason about where and why anomalies occur.

To determine why something has stopped working, it's useful to know how it was supposed to work. Therefore, model-based systems contain knowledge about the structure and behavior of the system and each component in the system. This knowledge enables the program to predict the system's behavior under certain conditions and compare it to its actual behavior.

Since the overall task of a model-based system is to use knowledge about the component's structure and behavior to determine which components could have produced the anomalies, the following information is needed[5]:

1) measurements (observations) from the system

2) descriptions of the system's internal structure (including connections)

3) a description of the behavior of each component.

In model-based systems the large number of ways the system can fail is reduced to the number of ways in which each component type can fail - a technically equivalent and more manageable approach. This technique is strongly component INDEPENDENT. Given a design description of a component, work can begin on diagnosing the component before it is put into a system. Since each component has its own model of how it operates, creating models for new systems is done by simply merging the individual component models.

Set-Covering

A third type of diagnostic expert system is a set-covering system. A set-covering system is very similar to a rule-based system except that knowledge about the system's failure modes is stored in a database instead of being encoded as rules. Set-covering is more dynamic and powerful than simple rule-based systems but, like rule-based systems, set-covering systems can only recognize and react to system conditions that have been predefined by an expert.

In set-covering, once an abnormal operation has been detected, the symptoms of the anomaly are matched against the predefined failure modes to come up with an ordered set of possible explanations. Therefore, it is imperative that knowledge of all failure modes of a system is available. This diagnostic knowledge is a collection of known component failures, their symptoms, and possible causes for the failure.

A set of rules will look at the detected symptoms of the current state of the system and seek an explanation for them by linking all symptoms to the failure and failure cause data that can best account for all the observations. Every time a detected symptom matches the predefined symptom of one of the failures, a failure hypothesis is created which postulates a failure and its cause as a possible explanation for the observed abnormalities. Once all

hypotheses have been generated, the system determines which hypotheses are the most likely cause(s) for the abnormalities.

The Need for Expert Systems in Space:

As mentioned before, as systems become larger and more complex they become more and more difficult to monitor and diagnose. On Earth expert systems are an appealing monitoring alternative or aid to human operators. Space is no exception. Technology is advancing and the size and complexity of the systems being put into space is growing. The need for automated diagnostics is becoming apparent. For example, Skylab had an 8KW power system that required 15 - 20 electrical power specialists for ground support and a lot of valuable crew time to deal with faults. Space Station Freedom's power system, with its initial 75 KW power system, has become so much larger and more technically advanced than Skylab that it will require hundreds of ground support personal if not automated[12].

Expert systems offer a number of advantages for space bound projects. As mentioned above, expert systems can reduce the number of airborne and ground support personnel necessary for safe operation[6]. This in turn reduces the cost of operating such projects as Space Station Freedom. Reducing the number of personnel aboard is a big advantage since the small volume of such space craft limits crew size. Therefore, crew time is very valuable and should not be used to monitor a system that an expert system could monitor.

Reliability is another advantage expert systems offer space bound projects. In space it is imperative that the power system, life support systems, etc. remain operable. Therefore, reliable monitoring and diagnosing of these systems is a necessity. Unlike human operators, expert systems are always 100% focused on their monitoring tasks. Expert systems do not get distracted, bored or tired[3]. They are always alert, operate 24 hours a day, and react faster than humans.

With the complexity and size of new space based systems and the limited number of available personnel it is apparent that there is a need for automating the diagnostic and monitoring processes in space. Expert systems offer many advantages for space based system applications. With their reliability, responsiveness, and the ability to reduce the number of airborne and ground support personnel needed, expert systems can effectively satisfy diagnostic automation needs in space.

Review of Existing Expert Systems

Expert systems have been developed for a broad range of applications. In the discussion that follows, only expert systems whose application falls into one of the three domains related to TROUBLE III will be examined. Since TROUBLE III is a diagnostic expert system for a space based power system these domains are: 1) power systems, 2) diagnostics systems and 3) space based systems.

Expert Systems for Power Systems

In the power domain, expert systems have been developed to handle fault diagnostics, resource scheduling and contingency screening. Since power systems' performances are fairly easy for experts to predict, they lend themselves very nicely to the application of diagnostic expert systems. A detailed discussion about diagnostic systems for space based power systems is deferred to section 2.3.3. What follows is a brief description of two terrestrial power diagnostic systems: D2 and PERF-EXS.

D2, developed at Carnegie Mellon, is a distributed expert system for fault diagnosis of a power system[2]. D2 uses a set-covering model written in OPS5 to generate all the credible hypotheses that explain a given set of detected alarms. It runs on three microVAXes in an environment developed to allow it to run in parallel.

D2 is broken down into three members: the manager, proposal crew and construction crew. The manager uses set-covering for alarm processing. The proposal crew conjectures events to expand incomplete hypotheses found by the manager. The construction crew then takes these conjectures and translates them into actual events.

PERF-EXS is a diagnostic expert system based for performance evaluation of a power plant[4]. It was developed jointly by the companies APEIRON and ANSALDO and was aimed at capturing ANSALDO'S vast knowledge and expertise in power plants. Most diagnostic expert systems use a custom built expert system shell. PERF-EXS instead makes use of a general purpose, commercially available shell named KEE (Knowledge Engineering Environment).

PERF-EXS is broken down into two components: fault detection and fault models. These two components work together to detect malfunctions, start the diagnostics, and locate the cause(s) of the malfunction. The physical power plant is divided into the following subsystems: boiler, turbogenerator, condenser, electrical auxiliaries and water regeneration. The diagnostic

expert system monitors the performance of each subsystem and integrates information from all of them to correctly diagnose malfunctions.

In the detection phase, PERF-EXS receives information about measured variables and depends only on those readings to detect malfunctions. In the diagnostic phase, the general inference engine reasons on the failure knowledge base, searches for the cause(s) of the malfunctions detected, and suggest ways to correct them.

PERF-EXS is very similar to TROUBLE III. Both rely on measurement readings to detect the system's symptoms. Both divide the power system up into independent subsystems and both use a commercially available shell. The main difference is that PERF-EXS uses a mode-based approach while TROUBLE III uses set-covering.

Resource scheduling for a power plant on Earth is not a major concern. If every user on the system wants to operate at the same time, the power plant simply generates more electricity or buys it from other plants on the system grid to satisfy the peak electricity need. But in space, there is a maximum amount of electricity that can be produced and many more users than can be supplied at once. Therefore, the system has to schedule resources to ensure that every user gets the electricity it needs at some time while never overloading the system. Since this problem only occurs in space power systems, the discussion of these expert systems is deferred to section 2.3.3.

Contingency screening is the last power system area that will be discussed. It is very important for the power industry to be able to predict the consequences of disturbances and take measures to prevent deterioration in service quality.

Case Western Reserve University along with Cleveland Electric Illuminating have developed a rule-based expert system with knowledge of the power system that uses pattern recognition techniques to do contingency screening[17]. This program has been demonstrated on moderate size power systems.

Non-power Related Diagnostic Expert Systems

In this section, diagnostic expert systems will be discussed that are competent in domains other than electric power. These are typical of diagnostic systems that have been successfully developed for many diverse applications.

The most famous diagnostic expert system is MYCIN[19], a medical diagnostic

program which queries a patient like a doctor to diagnose the patient's illness. The program reasons on the symptoms the patient has entered. When it comes to a fork in the decision tree, it asks for more information to help decide which path to follow until it finally reaches a conclusion on the patient's health.

EDNACS (Expert Diagnostics for Nitric Acid Cooling System) is an example of a diagnostic expert system developed for a chemical processing plant[16]. EDNACS was developed at the University of Waterloo to regulate the temperature of nitric acid in a chemical cooling system. EDNAC is divided into 4 stages: sensory, fault detection, anticipation and prevention. The sensory stage observes the status of objects and reports abnormal conditions to the fault detection stage. The fault detection stage uses rules to determine what fault(s) corresponds to abnormal conditions. Anticipation state predicts the future consequences of the fault while the prevention stage suggests actions that may prevent future failures.

MYCIN and EDNACS are only two of the many diagnostic expert systems that have been developed. But they show how diverse the applications can be. There is no limit to where diagnostic expert systems can be applied and power system diagnostics, like TROUBLE III, are just a small part of that domain.

Space Based Expert Systems

Several areas are currently being explored to assess the benefits of applying expert systems to space based systems. Power system diagnostics, resource scheduling, potable water systems, and data reduction are just a few areas where expert systems are being investigated.

NASA is committed to automating Space Station Freedom. Both the Lewis Research Center and the Marshall Space Flight Center have automation labs that are currently developing diagnostic expert systems for Space Station Freedom. Lewis' developments include TROUBLE III and APEX while Marshall is developing STARR, AMPERES, and ADEPT.

APEX (Autonomous Power Expert System) was developed using KEE and LISP on a Texas Instruments Explorer II workstation[9]. It is a rule-based system that performs fault diagnostics on the 20kHz switchgear. APEX is able to detect static faults in the switchgear as well as incipient failures (ie. insulation breakdown in transformers, contact depletion in mechanical switches, and thermal conductivity degradation in power semiconductors).

STARR was built on a Xerox 1109 workstation using KEE[20]. It is a rule-

based system that monitors a space power system, recognizes problem states, identifies the failure and recommends proper actions. Each component of the test bed has a corresponding object in STARR. The program monitors the test bed and looks for symptoms, then generates fault hypotheses. In normal operation, STARR gets updated information about each component approximately once per minute.

STARR was used as the prototype for AMPERES (Autonomously Managed Power system Extendable Real-time Expert System)[13]. AMPERES uses a "component centered" approach which encompasses the strength of a model-based approach. It, like STARR, is a rule-based system that is composed of the following functional models: main controller, status monitor, fault diagnosing, knowledge base and interface handler. Each component of the test bed has a corresponding object in AMPERES. Once symptoms are detected, fault rules are run to find the faults and give a cause and recommended action.

ADEPT (Automatic Detection of Electric Power Troubles) is yet another diagnostic expert system created by Marshall for fault isolation on a power system21[21]. ADEPT is implemented in LISP on a symbolic 3670. It uses Ohm's law as a basis for the rules that isolate the faults. ADEPT looks for significant changes at any sensor point then flags that fault condition and sends it off for fault rule analysis.

STARR, AMPERES and ADEPT encode their failure cause diagnostic knowledge in rules which makes them difficult to adapt to a new system. With Space Station Freedom's power system still evolving, frequent revisions need to be made to these expert systems to make their rules appropriate for Space Station Freedom. This is where TROUBLE III, with its flexible set-covering approach, has superior flexibility over other diagnostic expert systems.

Resource allocation is a major problem aboard Space Station Freedom. For the power system, allocating this limited resource so that station productivity is maximized without overloading the system would be a difficult task for a human operator to do quickly. Power allocation and scheduling need to be automated and expert systems are a prime candidate. NASA Marshal is investigating two schedulers on their test beds.

LES (Load Enable Scheduler) is coded in LISP and runs on a LISP workstation[22]. It schedules and reschedules the payloads for Space Station Freedom and is capable of handling hundreds of scheduling constraints.

LPLMS (Loads Priority List Management System) is also implemented in LISP

on a LISP workstation[22]. It continuously computes a priority list of the current users so in case of faults or loss or power, it can tell the system which loads to shed first to avoid shutting down the whole power system.

Two other applications of expert systems technology in space systems are for space station's potable water system and for Hubble telescope's data reduction. The Application Generator[1] is a fault diagnostic system analyzes failures which have occurred on the system that reclaims potable water from waste water. I-DARE[22] is an expert system that performs data reduction of the telemetry stream from the Hubble telescope.

SPACE STATION FREEDOM'S POWER SYSTEM

NASA has been engaged in the definition and design of Space Station Freedom since 1984. The design for the electrical power system has yielded two major requirements: 1) deliver 87.5 kW continuous electrical power and 2) deliver 113.5 kW peak power for up to 15 minutes during each orbit[18].

Deciding what type of generation and distribution to use are the two main design issues for Freedom's Electrical Power System (EPS). The three possible designs for generating the electrical power are an all photovoltaic (PV) system, all solar dynamic (SD) system or a hybrid (PV and SD) system. The options considered for distributing the electricity are a 20 kHz 440v AC system, a 400 Hz 440v AC system, and a 120v DC system. The following section describes Space Station Freedom power system's design evolution.

Chronology of Distribution System

In September 1984, the baseline configuration for Space Station Freedom's electrical power system consisted of photovoltaic generation and 20 kHz AC distribution. PV arrays generated 160v DC which was sent to the main invertor unit to be transformed to the primary distribution of 20 kHz, 440v AC. The primary distribution was then stepped down to 20 kHz, 208v AC for the secondary distribution to the loads[24]. This configuration made use of the mature PV technology but introduced 20 kHz - a new technology.

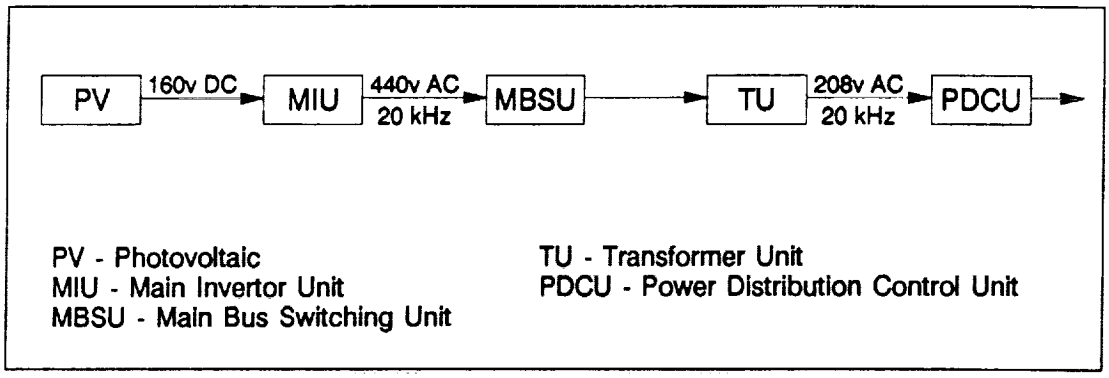

Figure 1: EPS Configuration September, 1984

After further study, 20 kHz technology was judged immature and risky because problems developing a resonant invertor to change DC to 20 kHz AC. So in December 1985, a new baseline configuration was adopted that featured 400 Hz distribution instead of 20 kHz. The generation remained the same, PV 160v DC, but the primary distribution was changed to 400 Hz, 440v AC and the secondary became 400 Hz, 208v AC[24]. 400 Hz was selected because that technology was used throughout the aircraft industry.

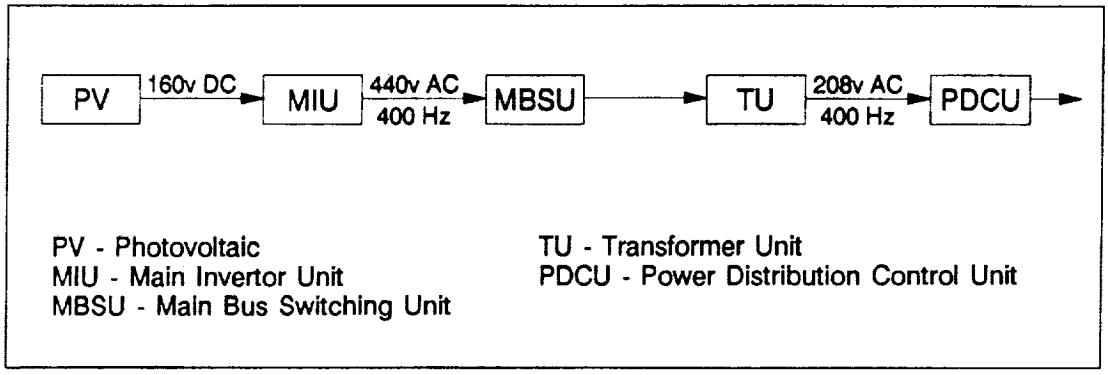

Figure 2: EPS Configuration December, 1985

In April 1986, the baseline configuration returned to the original September 1984, design: PV 106v DC generation and 20 kHz primary and secondary distribution[24]. The change was made due to the strong objections by experimenters to the electromagnetic interference (EMI) generated by 400 Hz.

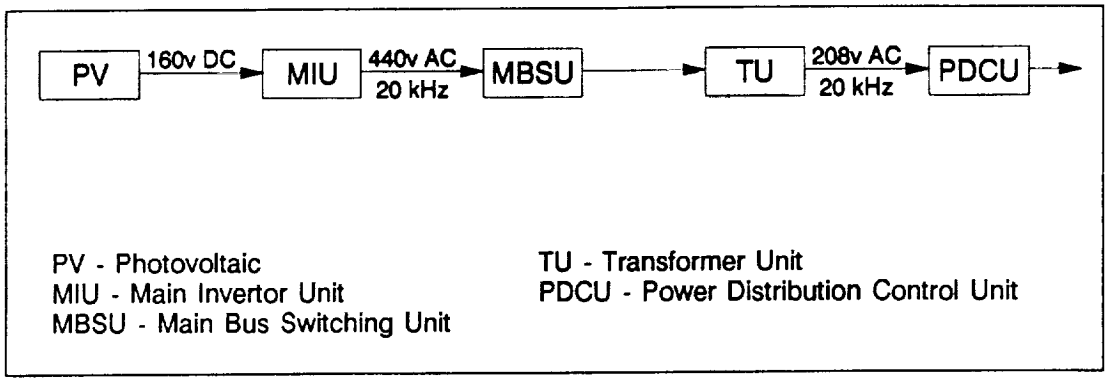

Figure 3: EPS Configuration April, 1986

May 1986, brought another change to the electrical power system baseline configuration. A hybrid power source was adopted. Generation was now achieved by both photovoltaic and solar dynamic. The original PV 160v DC source was still used but now a 1200 Hz AC solar dynamic source was added in parallel. In this configuration the 160v DC was inverted and the 1200 Hz AC converted to the 20 kHz 440v AC primary distribution then to the 20 kHz 208v AC secondary[21].

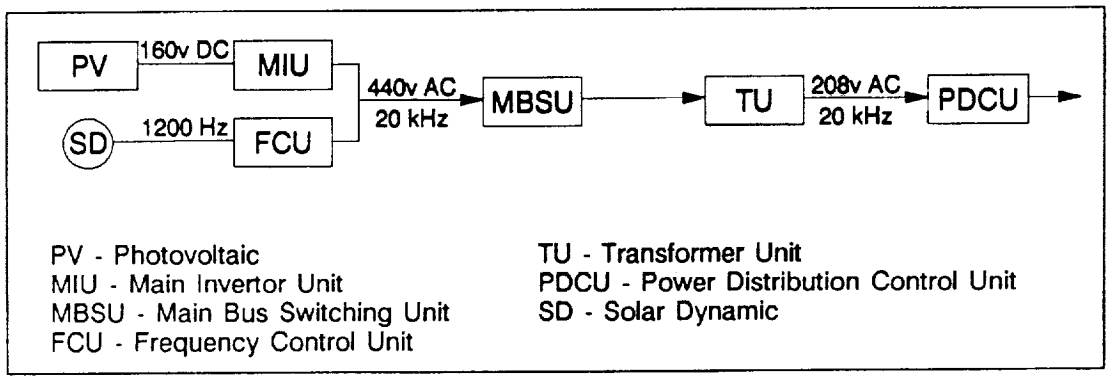

Figure 4: EPS Configuration May, 1986

Adding the immature solar dynamic technology increased the development time and cost. So in September 1987, the National Research Council recommended that solar dynamic generation be deferred until the growth stage of the space station. Once again the baseline configuration returned to the September 1984, version: PV 160v DC generation and 20 kHz AC distribution.

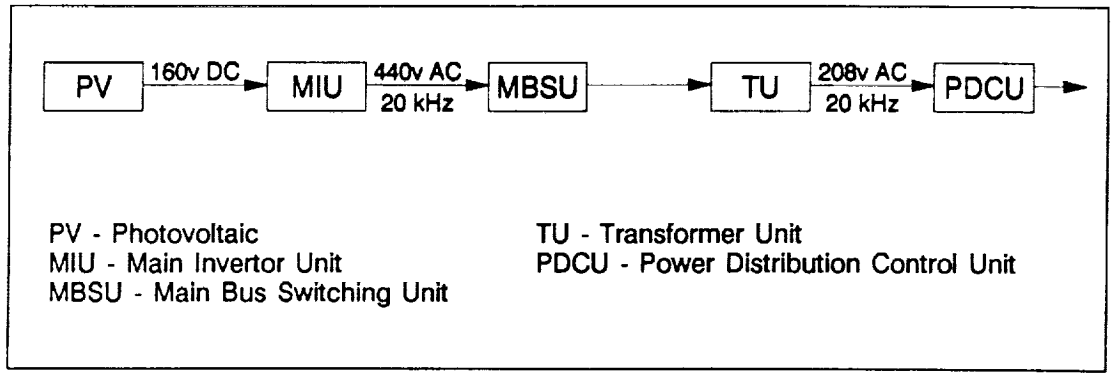

Figure 5: EPS Configuration September, 1987

This configuration was changed in December 1988, because the Europeans and Japanese wanted a DC secondary distribution instead of AC. The baseline then became PV 160v DC generation, 20 kHz, 440v AC primary and 120v DC secondary distribution[24]. With this configuration, DC was converted to AC then back to DC which meant both AC and DC distribution technology had to be developed.

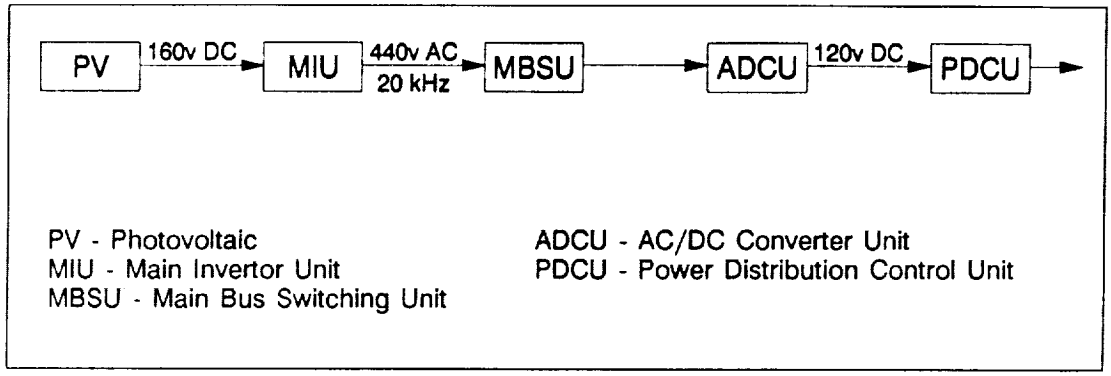

Figure 6: EPS Configuration December, 1988

In a cost-cutting exercise during September 1989, it was decided to get rid of the AC and have a total DC configuration: PV 160v DC generation, 160v DC primary and 120v DC secondary distribution[24].

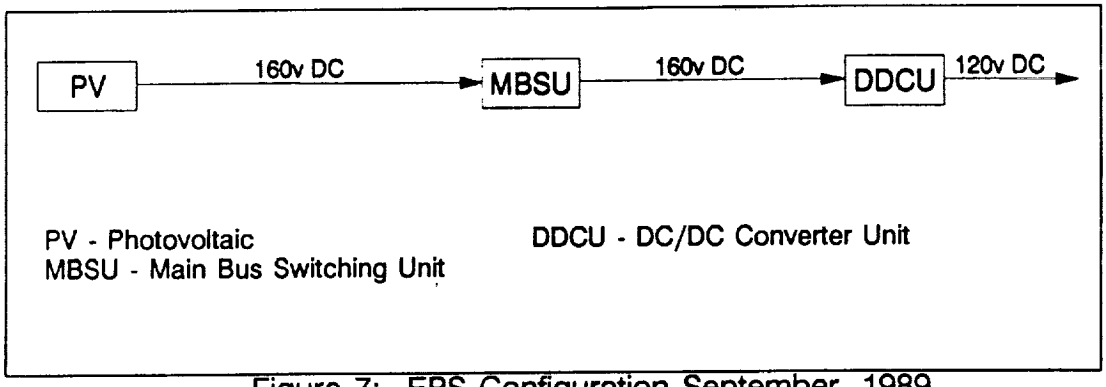

Figure 7: EPS Configuration September, 1989

As of today, the all DC configuration from September, 1989 is still the baseline configuration for Space Station Freedom's electrical power system.

Table 1: EPS Baseline Configuration Changes

DATE	GENERATION	PRIMARY DISTRIBUTION	SECONDARY DISTRIBUTION
September 1984	PV 160v DC	20 kHz, 440v AC	20 kHz, 208v AC
December 1985	PV 160v DC	400 Hz, 440v AC	400 Hz, 208v AC
April 1986	PV 160v DC	20 kHz, 440v AC	20 kHz, 208v AC
May 1986	PV 160v DC SD 1200 Hz AC	20 kHz, 440v AC	20 kHz, 208v AC
September 1987	PV 160v DC	20 kHz, 440v AC	20 kHz, 208v AC
December 1988	PV 160v DC	20 kHz, 440v AC	120v DC
September 1989	PV 160v DC	160v DC	120v DC

Need for Flexibility

The type of generation and distribution being used by an electrical power system affects the behavior of the system and the design of the supporting components. In writing a diagnostic expert system for such a system, the behavior of the system and each subcomponent is described and analyzed. If the distribution or generation design changes in mid stream, this knowledge about the system becomes obsolete and the knowledge engineer has to start the information gathering process over. Ideally, the expert system design would be flexible enough that major changes to the generation or distribution baselines would have minimal affects on the program.

Since 1984, the generation and distribution baseline configuration for Space Station Freedom has changed six times. Any diagnostic expert system applied to this system should be flexible so it can handle these major design changes if any more should come along. TROUBLE III was designed with this type of flexibility and is discussed in the next chapter.

TROUBLE III

Background

Automating the operation of Space Station Freedom's electric power system is rife with technologic opportunities. Expert Systems will be used to schedule power resources as well as monitor and diagnose faults on the power system[8]. TROUBLE III , a diagnostic expert system, has been created to perform fault detection and isolation for the electrical power system of Space Station Freedom.

TROUBLE III is one of the projects of the Power System Facility Advanced Automation Lab located at NASA Lewis Research Center. It is an object-oriented program developed using the commercially available AI shell, ART (Automated Reasoning Tool) from Inference Corporation. ART is a LISP based shell and is installed on a Texas Instruments Explorer II workstation.

Space Station Freedom's electric power system design is still changing. This complicates designing an expert diagnostic system since there is no operating or failure experience. Usually, expert systems are built for working systems for which there are knowledgeable experts. In these cases, the design and specifications are an a priori knowledge base upon which the knowledge engineer builds. What measurements are needed, how each component of the system interacts with other components, and what faults are most likely to occur in given situations are already known to the expert and establishes a good frame work upon which to build a diagnostic program.

Space Station Freedom is different. Without exact design and performance information and with no operating experience it is hard to identify abnormal behavior let alone write an expert system to do it. What is needed is a flexible expert system that can quickly accommodate a design change. Therefore, TROUBLE III has been designed using set-covering to represent failure knowledge, rather than rules or models, to give it the flexibility needed to handle the design changes in the power system.

Design

The objective of TROUBLE III is to apply expert system technologies to develop an automated system for detecting and diagnosing failures and failure causes for the electric power system of Space Station Freedom. TROUBLE III is an object-oriented program which performs the following tasks:

1. Symptom Detection
   Monitors measurements throughout the system and determines what failure symptoms, if any, are present.

2. Reasoned Assumptions
   Rules examine the current state of the system and decide whether each potential failure object should have a reason for inclusion or exclusion when it is matched to the detected symptoms. If the failure hypothesis is to be excluded then no further processing is done.

3. Failure Hypothesis Generation
   Detected symptoms are matched to the predefined failure data to come up with the possible failure hypotheses.

4. Subcause Generation
   The FHRs are linked together to form chains. The chains start with the last failure of a chain reaction and work backwards to a root cause. For each root cause, there is a separate chain. Chains take the form "A failed because of B, B failed because of C; therefore C is a root cause of A". Once the chains are formed, they are ranked in order of their ability to explain the observed abnormalities.

5. Explanation and Justification
   The Justification Module takes any of the chains and justifies to the user why the cause of that chain was selected as a possible root cause for that failure.

6. Human Interactions
   The Output Module displays the requested results to the user. This module can show the detected symptoms, the top level failures and possible root causes (ranked in order of most probable) or the justification for one of the chains. An Input Module allows the user to load a test file or manually change the value of any of the measurement devices for the purpose of testing.

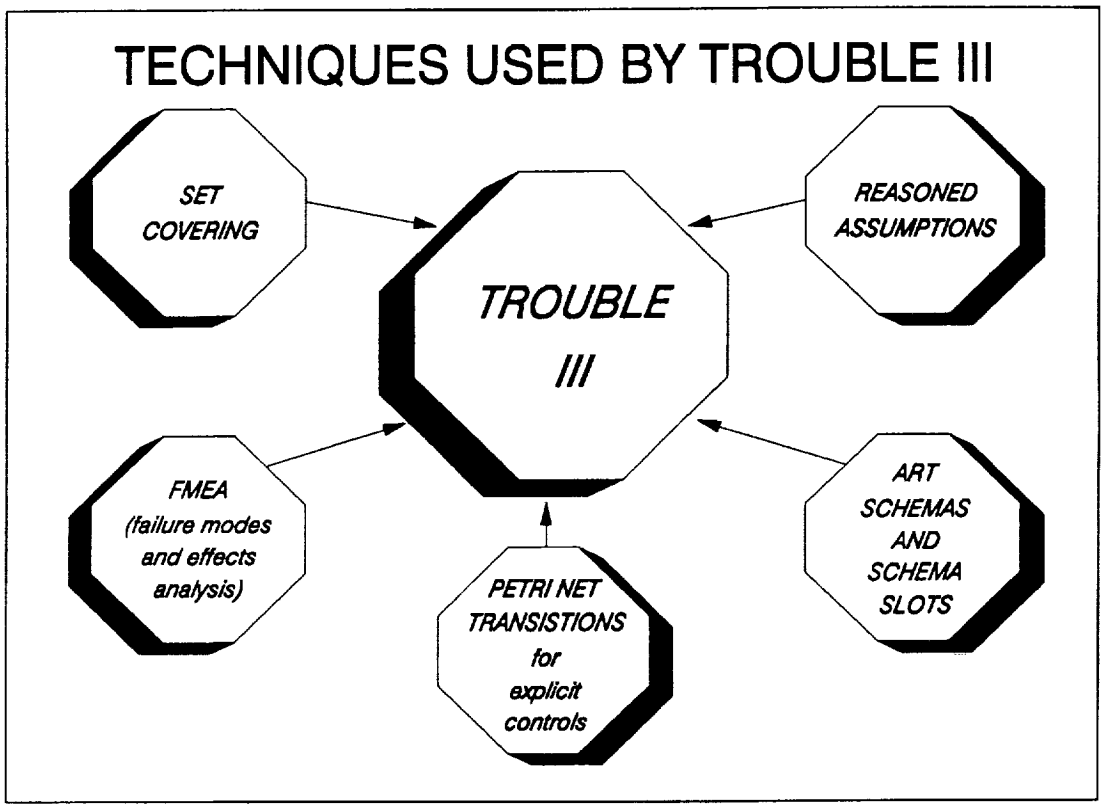

Figure 8: TROUBLE III Techniques

Set-Covering

In looking for a design approach for TROUBLE III, rule-based, set-covering and model based approaches were investigated to see which was best suited for Space Station Freedom's evolving power system. There are many examples of diagnostic expert systems in the literature that use either a rule-based (APEX, ADEPT, STARR, AMPRERES) or a model-based (PERF-EXS) architecture. However, examples of diagnostic systems using a set-covering architecture seem to be absent from the literature even though set-covering is especially well suited to diagnostics[11].

A rule-based solution was considered but was not used due to its lack of flexibility in handling a changing system. Rule-based designs, like APEX, STARR, AMPERES, have the failure knowledge encoded in the rules. Each rule looks for one specific set of conditions. If all the conditions are met then that rule's failure mode becomes a possible failure hypothesis. The same condition can appear in several failure rules. If the system changes,

as is the case with Space Station Freedom, the programmer must go through the complicated interconnections of the rules and change the conditions and resulting failures to correspond to the new design. This makes rule-based design very domain specific and not a good choice to be used with Space Station Freedom's dynamic evolving power system.

Davis and Hamscher[5] make a good argument for the use of a model-based architecture. Unfortunately, model-based systems are best suited for an existing system since knowledge about the internal structure of each component, the connections of each component and behavioral description of each component is needed. With Freedom's power system still changing this information is not available. Making a good model of the system is already a difficult task, but adding the complexity of modeling an evolving system makes a model-based architecture unattractive for Space Station Freedom.

In set-covering, knowledge about the failures is stored as data with the rules being independent of this failure knowledge. With this technique, variable data (symptoms) is compared to variable date (failure modes). This gives the expert system the flexibility to adept to changes in the system without having to rewrite rules.

With flexibility being the key in selecting an architecture for TROUBLE III, a set-covering approach was selected. With this architecture, updating the failure knowledge database is the only change needed to adapt TROUBLE III to the new design.

Transitions

A production system's architecture employs a set of rules, a global database and an inference engine that performs the recognize-act cycle of match, conflict resolution and rule-firing[23]. TROUBLE III is not strictly a rule-based system but does perform the recognize-act cycle. Therefore, TROUBLE III is not a pure production system according to the definition but since it behaves like one, it will be considered a production system in this discussion.

In a pure production system, rules are in a sequential list and are evaluated one at a time according to the order of the list. When all the conditions of a rule are found true, the rule is executed and the rule evaluation begins again at the top of the rule list. Most large production systems, including TROUBLE III, can be divided into sub-systems (modules) that perform well defined tasks. If the programmer exerts control over when each sub-system's rules fire, the sub-systems will behave like sub-routines or functions in procedural code.

Explicit control over the flow of the program is advantageous to a production system that is modularized. Since a production system evaluates the entire list of sequential rules, the more rules there are the slower the execution is. Therefore, the production system will run faster if the set of rules it is evaluating is only a subset of the whole set of rules. Also, many times certain tasks need to be performed before others. In a pure production system where all rules are eligible for execution all the time, implementing "rule priority" correctly becomes difficult. Where as having explicit control over the rules makes rule prioritization a simple task. Therefore, for efficiency and correctness a structure is needed that adds explicit control to production systems.

Zisman[23] encountered this problem in his work with asynchronous concurrent production systems for independent event driven office procedures. Zisman suggested that Petri net transitions be used to model the interaction and temporal relationships between the asynchronous concurrent events.

TROUBLE III is not a pure asynchronous concurrent production system like Zisman's office procedure but it does have the same need for explicit control over rule-firing. TROUBLE III expands Zisman's work to a non-asynchronous concurrent production systems. Like Zisman, TROUBLE III uses Petri nets to dynamically extract from the set of system rules those rules which are relevant given the present system state. Transitions represent a process or task; since these processes are described by productions (rules), the transition is really a "home" for the set of rules describing the process[23].

Transitions control which set of rules are active in TROUBLE III. Each module has its own transition object which appears as one of the conditions in each rule in the module. Transitions have the value inactive, active, or fired. Petri nets, in the form of rules, control the status of each transitions. If all the conditions are satisfied for a transition's rule, that transition becomes active, otherwise it remains inactive. More than one transition can be active at a time but only one can be fired. Conflict rules are used to resolve the conflict of which transition to fire when multiple transitions are active. If only one is active, it immediately fires. All rules associated with a fired transition constitute the active rule set. TROUBLE III's inference engine continually cycles through the active rule set evaluating its rules. Only rules in the active rule set are evaluated. As transitions are fired and become inactive, the active rule set changes.

TROUBLE 3 expanded Zisman's work to include nesting of transitions. In TROUBLE III's design, rule prioritization was needed inside some of the modules. To handle this, new sets of transitions were implemented inside of

modules thus creating a nested Petri net transition structure.

By using Petri net transition, TROUBLE III was able to break its design into modules and have explicit control over rule-firing. This reduced the number of active rules which in turn reduces the processing time.

Failure Modes and Effects Analysis

The failure knowledge database can be obtained by performing a failure modes and effects analysis (FMEA) of the system to be diagnosed. FMEA is a method of identifying possible system failures and their consequences.

In a FMEA, an expert on the system describes the ways each system component can fail. Each of these failures is called a failure mode and is a qualitative description of the failure. For example, "No discharge current" would be one of the possible failure modes for a battery charge discharge unit (BCDU).

The expert then describes all possible causes for each failure mode. "No charge on Battery" is an example of one of the possible causes for the BCDU failure mode "No discharge current". Each unique failure-mode-cause pair in the FMEA is used as a separate failure object in the failure database. The symptom that a failure mode manifests is also identified in the FMEA. It too is stored in the failure database.

The FMEA is complete when all of the failure modes, causes and symptoms have been described for each system component. All this information collectively represents the known failure knowledge of the system. Once this information is entered into the database it is used by the failure rules to generate the failure hypothesis.

Detection Module

To detect failures TROUBLE III searches for evidence of possible failures (symptoms) using measurements taken throughout the power system. Each measurement has its own unique data object. This object contains many characteristics such as: the unique name of the device, the type of device (voltmeter, ammeter etc.), the measuring unit, the present measurement value, the device's purpose and its location. These attributes are defined in advance and do not change, with the exception of the measurement value. During each sampling period, each measurement's value slot is updated to reflect the most current information about the system's state.

The detection module converts the quantitative measurements into qualitative

symptoms that describe the system's performance. Detection rules monitor the measurements and look for predefined symptoms of abnormal behavior. During the FMEA process, a symptom associated with each failure mode was described. Each symptom has a detection rule. If the measurement object's attributes satisfy all the conditions of the rule, then the rule asserts the corresponding symptom to the system. If the conditions are not satisfied, then the symptom does not exist and cannot be reasoned about by the rest of the program. The detected symptoms give a qualitative description of the current state of the system.

Each symptom is itself a data object. The information stored about each symptom includes: its unique name, its description of the symptom (i.e. low output), its list of measurement devices used, its location of the measurement device, and a time stamp to tell when the symptom was detected.

The power system is broken down into many subsystems and components each with its own measurement devices and detection rules. Since measurements in one subsystem may be used by a detection rule in another, all measurement objects are stored in one file. The detection rules are grouped according to subsystem or component and stored in their own file. In this way a maintenance programmer can quickly and easily find the code that needs to be changed.

Once the system's measurements have been read in and the failure symptoms have been detected, a list of possible failures for the detected symptom(s) is generated by the Failure Hypothesis Module.

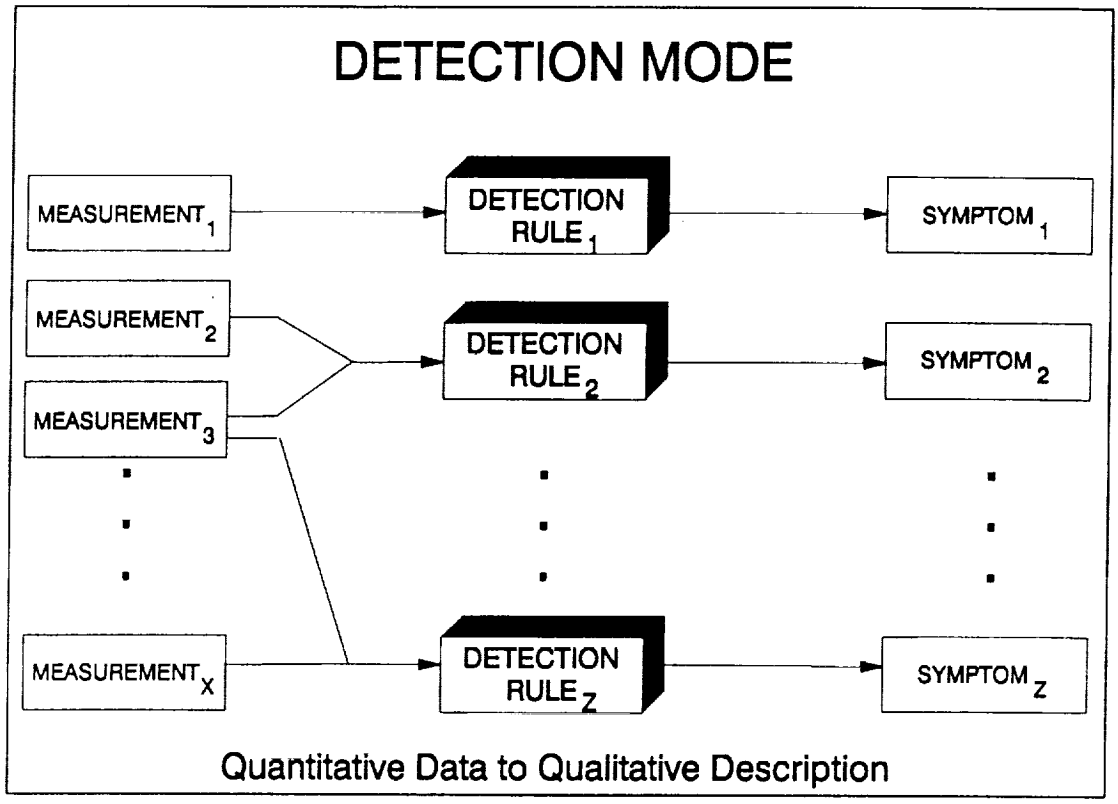

Figure 9: Detection Module Logic

Failure Hypothesis Module

The failure hypothesis module generates a list of possible failures to explain the symptoms identified by the detection module.

The essence of TROUBLE III's set-covering technique is encapsulated inside this module which matches the detected symptoms to the predefined failure knowledge. Two rules are used to take each symptom detected, search the objects in the failure database, and select all failure objects whose symptom slot matches the symptom in question. If these matched failure objects have a reason for inclusion or if a reason for exclusion has been defeated then the rules create a failure hypothesis record (FHR) for that failure mode (the reasons are maintained by the reason module). These rules are very general, matching variable symptom data to variable failure knowledge data. No failure knowledge is encoded in the rules.

Failure hypothesis records are objects that contain specific facts about possible failure modes and their causes. When the FHR rule fires, it combines information from both the symptom object and the failure object to generate a new FHR. The FHR's facts include the failure mode, failure

cause, failure symptom, information on the failed device, the parent FHR, chain number and time stamp which indicates when the FHR was created.

The FHRs taken together represent all known failures that can explain the detected abnormal behaviors. These hypotheses are reviewed to determine the most probable cause for the failure.

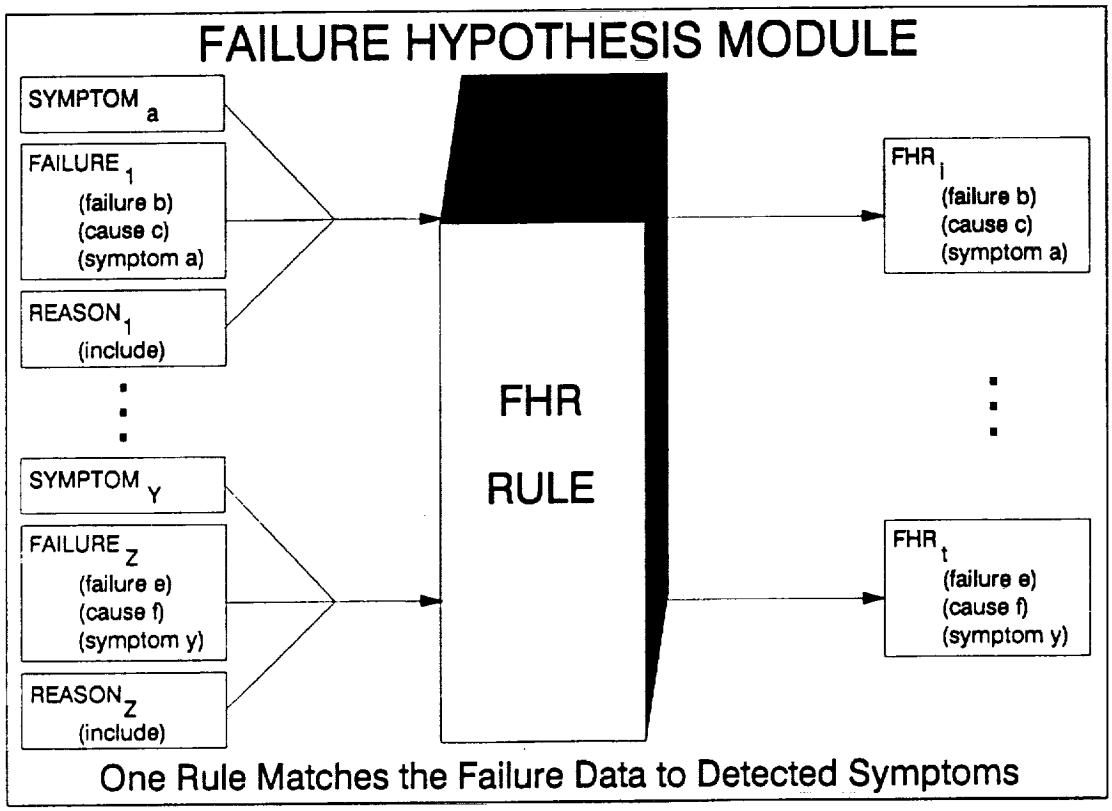

Figure 10: Failure Hypothesis Module Logic

Reason Module

Doyle[10] discusses a methodology, Reasoned Assumptions, to express uncertainty in terms of the non-statistical notions of typicality and defeasibility. The reason module employs Doyle's reasoned assumptions methodology to reduce the number of FHRs that can be created thus reducing the work TROUBLE III must do to determine the most probable cause. As mentioned previously, a failure object whose symptom has been detected must have a reason for inclusion or a defeated reason for exclusion before it can generate a failure hypothesis record. By posting the reason's status in each failure object, the reason module is able to control which failure objects become

FHRs. This limits the number of hypotheses considered while still generating enough to identify the most likely hypothesis.

In Doyle's work, the normal or initial conditions were expressed as rules of typicality which may be individually defeated if circumstances warrant. These rules were considered as default reasons. Default reasons were used in concert with other reasons expressing special cases, exceptions or other overriding conditions. TROUBLE III incorporates Doyle's methodology by creating unique reason objects for every failure object. These reason objects have slots named: failure mode, symptom, cause, reason, status and because. The failure mode, symptoms and cause slots are used to link it to the appropriate failure object. The reason slot is used to define the initial type of reason. If the associated failure is a normal failure then the reason is a default reason and has a value of inclusions. If the failure is a special case or exception, the reason value is set to exclusion. The status slot's value is maintained if the reason value is still believed or defeated if not believed. The because slot stores the reason in English for the justification module.

A set of maintenance rules are activated after the detection task and before the failure hypothesis generation task. These rules look for special circumstances which will make an normal failure unlikely or an exception failure a likely candidate. When these conditions exist, the rules change the reason status to defeated. If those conditions no longer hold, the status is changed back to maintained.

Failure objects whose corresponding reason objects have values of inclusion maintained or exclusion defeated will become an FHR if their symptom is detected. Failure objects whose corresponding reason object has values of inclusion defeated or exclusion maintained will not become FHRs even if their symptom is detected.

After every detection cycle, reason rules are used to check the status of the reason objects to limit the number of eligible causes while making sure all potential candidates are included.

Chaining Module

Usually, each failure in the FMEA has a subcause and the subcause has a subcause and so forth. Each will produce a failure hypothesis record. Although at time of creation these FHRs look independent, they are actually interconnected by the subcause relationships. The chaining module is responsible for creating chains (lists of FHRs) which show how the FHRs are interconnected.

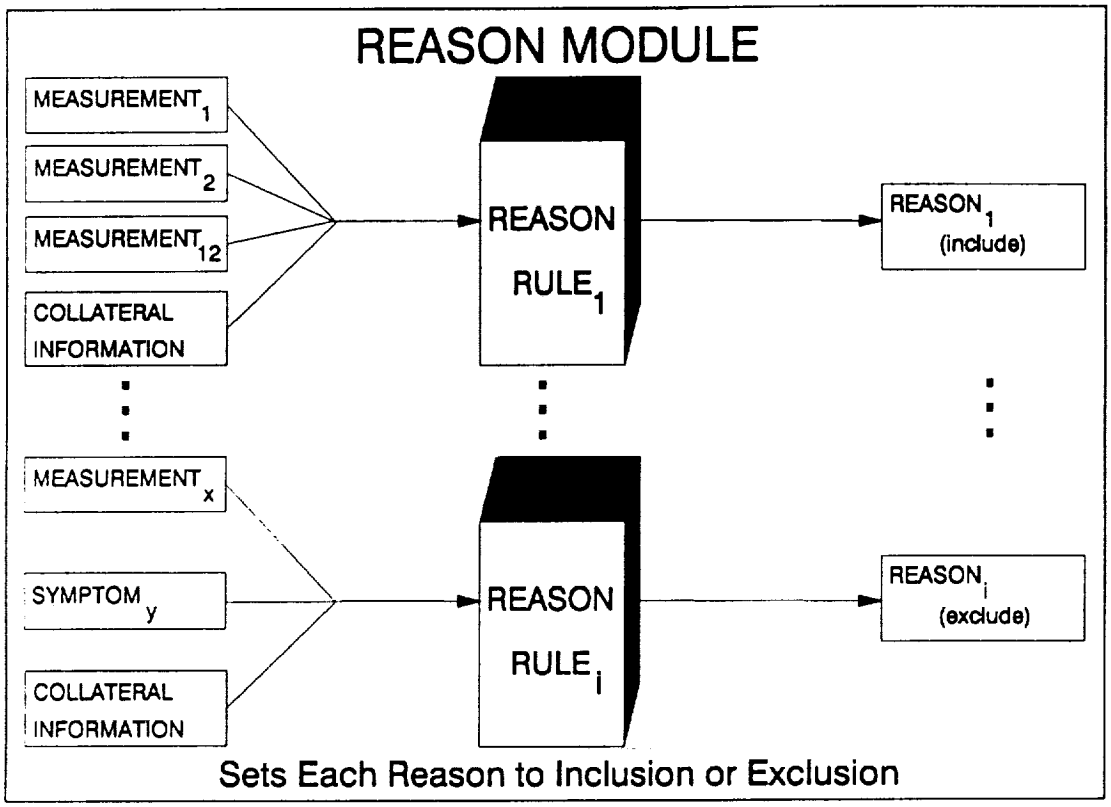

Figure 11: Reason Module Logic

After the failure hypothesis module has generated all the possible FHRs, the chaining module is activated to connect the FHRs into failure chains. FHRs can be members of more than one chain. All failure chains taken together form the failure hypothesis tree. Each chain represents a path from the top node of the failure hypothesis tree down to a root cause. Each of these paths represents a possible explanation for the observed abnormalities.

The first step in the chaining module is to create all the parent links. A FHR is the parent of another if the failure cause of the parent is the failure mode of the other. For example, given these FHRs:

```
(FHR 1                          (FHR 2
     (failure-mode A)                (failure-mode B)
     (failure-cause B                (failure-cause C)
     .                                              .
     .                                              .
     .                                              .
                    )                               )
```

FHR 1 is the parent of FHR 2 because the cause of FHR 1 (B) is the failure mode of FHR 2.

When a parent is found, TROUBLE III stores the unique FHR number of the parent in the parent slot of the child's FHR object. A FHR can only have one parent but can be the parent of many children. Once all of the parent nodes have been identified, failure chains are created by starting with a bottom node and adding it's parent to the chain list then adding the parent's parent and so forth until a top level failure is found.

The bottom node of a chain is the root cause of the chain. A root cause has no subcause. Root causes are identified by finding the FHRs which are not parents of any other FHR.

A top level failure is a failure that does not cause any other failure. They are identified by finding the FHRs with no parent. It is the end of the line and, in most cases, is a general failure. Top level failures are usually very noticeable to the system operator and are the most serious type to clear because they affect a large portion of the system. A voltage loss on a distribution line is an example of a top level failure while a failed open relay could be its root cause.

Usually, if a root cause is identified and cleared, all other failures in the chain will also be cleared. The failures in the middle and at the top most likely have occurred as a result of the root cause failure. Attempting to clear a middle failure without clearing the root cause is futile because the conditions that caused that failure are still present on the system and will cause the same failure to reoccur. Therefore, the root cause and top failure are of most concern in the failure chain.

Ranking Module

A list of potential failure causes is very useful when trouble shooting. But the process of isolating and correcting the failures is more efficiently accomplished if this list is ranked based on the probability of each failure cause actually causing the failure. In TROUBLE III the ranking module

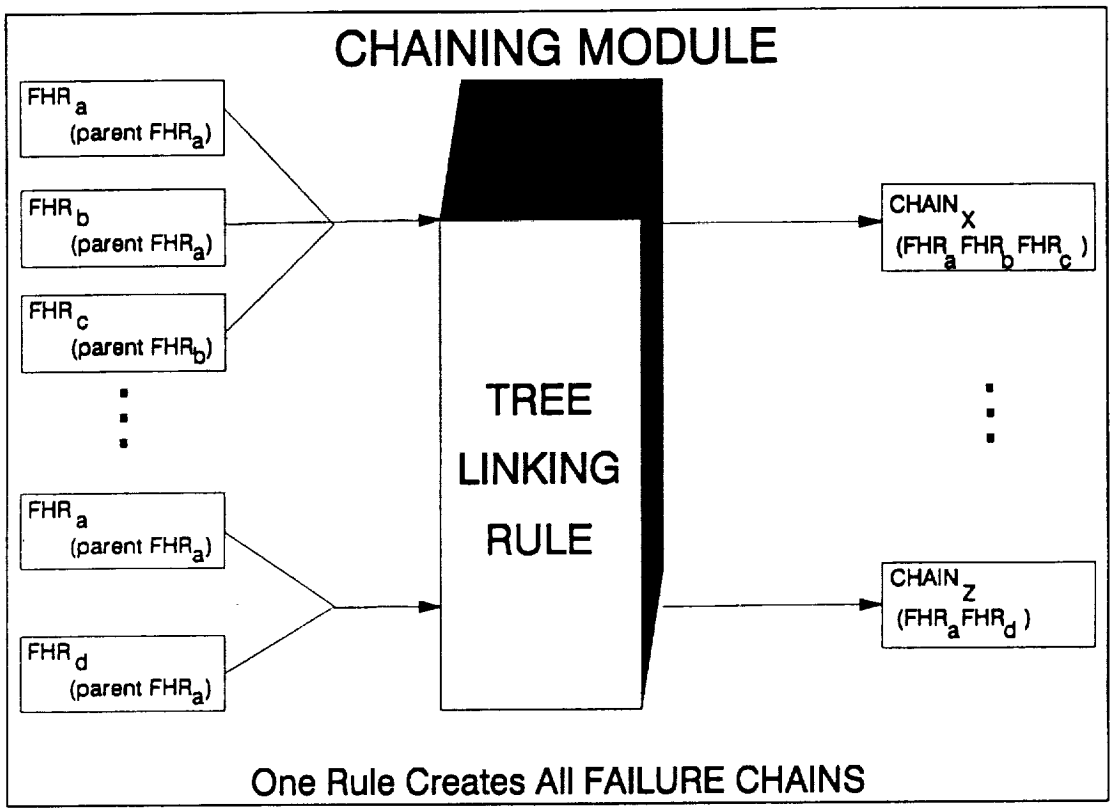

Figure 12: Chaining Module Logic

accomplishes this by ranking the failure chains (the list of potential failure causes) in decreasing order of likelihood.

As of now, TROUBLE III incorporates only one ranking scheme, although future plans include adding more ranking schemes. The ranking scheme used by TROUBLE III is called Related Modes With Different Symptoms (RMWDS). Using this scheme, the number of different symptoms associated with each chain of causes is counted and the ones with the largest number are ranked first. Each failure chain is made up of a list of related FHRs which also makes their failure modes related. Therefore, each chain includes a set of related failure modes. So the task of counting the different symptoms in each set of related modes is accomplished by counting the number of different symptoms associated with each failure chain. The ranking module looks at each FHR in a chain and counts the number of different symptoms present. This number is stored in the chain's RMWDS slot.

The logic behind this ranking scheme is simple: the more evidence for a hypothesis, the more likely that it is a valid one. Each detector in TROUBLE

III generates a single symptom, so the number of different symptoms in a chain is equal to the number of independent pieces of evidence associated with the chain. The probability of a chain's cause being the actual cause increases as the independent evidence associated with the chain increases. Therefore, a large count of different symptoms (RMWDS) in a chain implies a higher probability of that chain's cause being the real cause.

Other ranking schemes include easiest to check and severity of failure. Neither of these have been incorporated into TROUBLE III as of yet. Easiest to check ranks the causes in the order that would be easiest for the operator to verify. The idea being why do a complicated test procedure if a simpler one has the same probability of being at fault. If the easier to check one happens to be the fault, checking it first eliminates the work of checking the difficult one.

Ranking by severity of failure is a scheme that looks at the result of each possible failure cause and ranks them in order of the one that has the severest consequences. This method is based on the idea that severe faults should be tested for first because if they exist they will do more damage. Failure causes that impact small areas can be left unattended until all the other possibilities have been checked more easily than a severe failure cause.

These methods will be incorporated when better system operating knowledge becomes available.

Justification Module

The justification module answers the question of "Why is a particular failure chain included as a possible cause?". It justifies the chain's existence to the user by retracing the logic TROUBLE III went through in creating the chain.

After the top level failure modes and root causes have been displayed to the user, the user may want more information on why a particular failure cause appeared. By entering the justification module the user can obtain the desired information on any of the chains on the screen.

The justification module queries the user for the chain to justify and then proceeds to backtrack through the chain displaying TROUBLE III's reasoning. Starting with the top level failure, this module displays in a natural language format the failure mode, failure cause, failure symptoms, failed device and locations for each link (FHR) in the chain. An example is shown in Figure 15. It shows the justification of why the root cause "DC BUS Voltage VT Failed MAX" is a possible root cause for the failure mode "NO Discharge

Current".

Besides giving the logic justification for each root cause, the justification module also allows the user to believe or disbelieve any of FHRs in a chain. If the operator has good reason to believe that a failure mode and its cause is not a viable candidate, he/she can defeat the reason and remove that FHR from all chains. Once a FHR's reason is defeated, the logic of the rest of the chain falls apart and every root cause stemming from the defeated FHR is no longer a possible candidate. After defeating a FHR's reason, TROUBLE III will redisplay the possible top level failure modes and root causes with all chains dependent upon the defeated FHR eliminated. Any FHR's reason can be defeated, it does not have to be the top or bottom node of the chain. The higher up in the chain the defeated FHR appears, the more chains will be eliminated from contention. Defeated chains are never discarded, just suppressed from being printed.

The user also has the option of reinstating a defeated FHR. When this option is selected a menu appears with every defeated FHR being an option. The user can reinstate any or all defeated FHRs. Once the selections are made the possible failure modes and causes are redisplayed including the ones reinstated.

Input and Output Modules

The input module allows the user to directly alter measurement values. It is only used for developing and testing TROUBLE III's diagnostic logic. When TROUBLE III is directly connected to hardware, input data will be acquired from the hardware rather than from the input module.

The input module has two ways of entering or altering measurement values stored in the database: 1) load a test file or 2) manually enter the measurement number and value one at a time.

If the user selects to enter new data by loading a file, TROUBLE III displays a mouse sensitive menu which lists all possible test case files. Upon selection of a test file, TROUBLE III locates the desired file on the disk and loads the file into its working memory where input rules modify the existing measurement objects to correspond to the data in the test file. Once all of the data has been modified, the input module increments the clock and begins the diagnostic process by activating the detection module.

If the manual input mode is chosen, TROUBLE III keeps prompting the user for the measurement device number and its new value until the user has entered all of his/her changes. For each device number entered, TROUBLE

III locates the corresponding measurement object and modifies it value to correspond to the user's input. After all changes are made, TROUBLE III increments the clock and begins the diagnostic process by activating the detection module.

The output module is responsible for communicating the results of the diagnostics to the user. By selecting one of the mouse sensitive areas on the bottom of the screen, the user controls what output is exposed on the screen. By interacting with the output module the user is able to 1) invoke the input module, 2) view all symptoms detected, 3) view the FAILURE chains created, 4) invoke the justification module, 5) invoke the belief maintenance procedure or 6) EXIT the program.

Each of these options has its own window to display its results. The output module determines which window is exposed to the user. In this way TROUBLE III can switch back and forth between screens without having to waste time regenerating the outputs. This is especially evident in the detected symptoms and failure chains screen. If TROUBLE III regenerated the outputs for these screens every time the user selected them, a lot of time would be wasted generating output that hadn't changed from the last time. The other screens are cleared each time they are selected because they invoke modules which generate different results each time depending on the user responses to their prompts.

Figures 13-15 show output screens that are the result of running TROUBLE III on Test Case 1. Figure 13 lists all the symptoms TROUBLE III found with the conditions of the test case. Figure 14 displays the top level failures and root cause for the observed abnormalities ranked by RMWDS. Figure 15 shows the justification for the fourth possible cause of Figure 14.

Figure 13: Detected Symptoms for Test Case 1

Figure 14: Ranked Failure Chains for Test Case 1

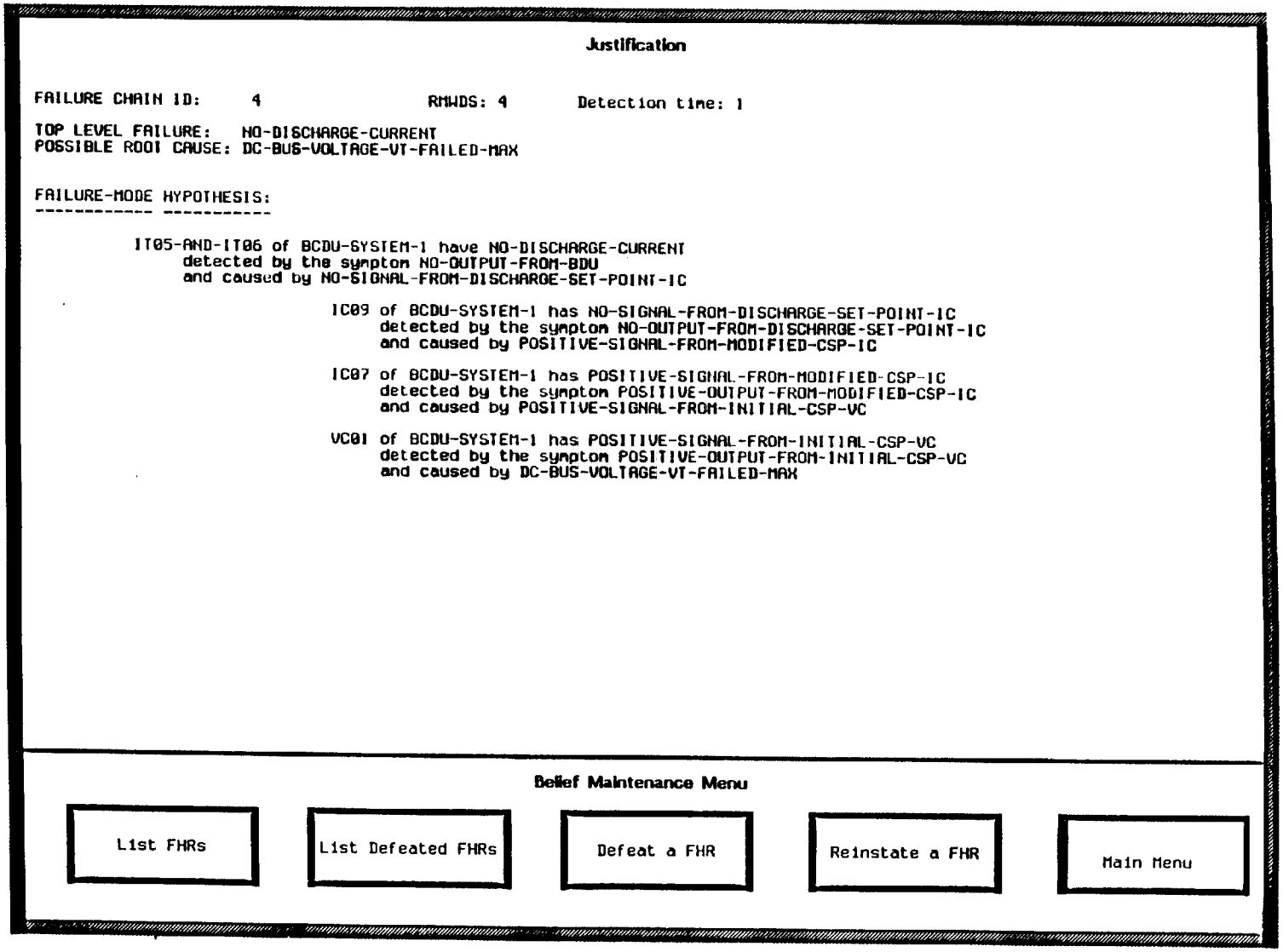

Figure 15: Justification for Failure Chain 4

Implementation

The previous sections have outlined the overall design and logic of TROUBLE III. Each of the modules in the design have been implemented using ART and LISP code. Each module is made up of rules, objects and databases that incorporate the design techniques of set-covering, FMEA, and transitions. After each module was developed and coded, it was tested to verify that it functioned properly. The next section describes the testing procedures.

Testing

Diagnostic expert systems, like TROUBLE III, must gain the user's confidence. TROUBLE III uses a testing method to prove that it is accurate and reliable. The testing method was to test and to debug each module after it was coded.

As each module was coded it was tested to make sure that the results agreed with the expectations. Specific test cases with known results were used as input and the outputs of the module were compared to the known results. If the output agreed, work began on the next module. If the results disagreed, then the logic of the module was reviewed and altered accordingly.

To do this testing, a power system was needed that could be analyzed and the results used in testing. Since the power system being designed by the project contractor was incomplete, James Dolce of NASA Lewis Research Center completed the unfinished control system design using proposed components and configurations[7]. As the actual design evolves, only the database will change not the logic of TROUBLE III's rules.

A failure mode and effects analysis was performed on the system design to determine failures and causes, needed measurements, and corresponding symptoms. This failure knowledge was placed into failure objects, measurement objects, and detection rules. Next, test cases were developed that simulated known failures.

Every module was tested with these cases and has produced the correct diagnosis. The overall system can successfully diagnose failures as defined in the data base for our hypothetical power system.

Future Developments

The next major development in TROUBLE III's future is integrating it with a test bed. For the integration to be successful, issues surrounding data

acquisition must be resolved, a failure mode and effects analysis must be performed on the test bed, symptom detection rules must be developed, and the diagnostic software integrated with test bed control software. Data acquisition issues include determining the rate of acquisition and how to convert the analog measurements into the measurement facts TROUBLE III needs.

Adding additional ranking schemes, as discussed in Chapter 4.2.8, is another enhancement planned for TROUBLE III. Before implementing these, collateral information on troubleshooting and failure probabilities will have to be gathered from an expert and incorporated into the design.

Another development will be to integrate TROUBLE III with the other expert systems being developed in the Advanced Automation Lab of NASA Lewis Research Center. The results of the fault diagnoses of TROUBLE III will be an important input into the load scheduler and security expert systems[9].

DIAGNOSTIC EXAMPLE USING TROUBLE III

This chapter is intended to clarify the function of each module of TROUBLE III by discussing, in detail, an example used in testing. The test system, failure mode and effects analysis and test conditions will be described. Then this chapter will follow the given test conditions through each module and explain how each module reacts.

Test System

The Battery Charger/Discharger Unit (BCDU) used in testing TROUBLE III was also used as the test system for this example. The purpose of the BCDU system is to charge the batteries when the solar arrays produce more electricity than used by the loads and discharge the batteries when the loads use more electricity than is produced. The batteries charge and discharge at a rate that keeps the bus voltage constant.

Figure 16 shows a schematic diagram of the BCDU design. Five parallel batteries, each with their own charger and discharger are connected to the two parallel DC bus lines running from the solar arrays to the loads. The control scheme uses measurements from the solar arrays, the loads, the batteries and the main bus along with constant set points for the

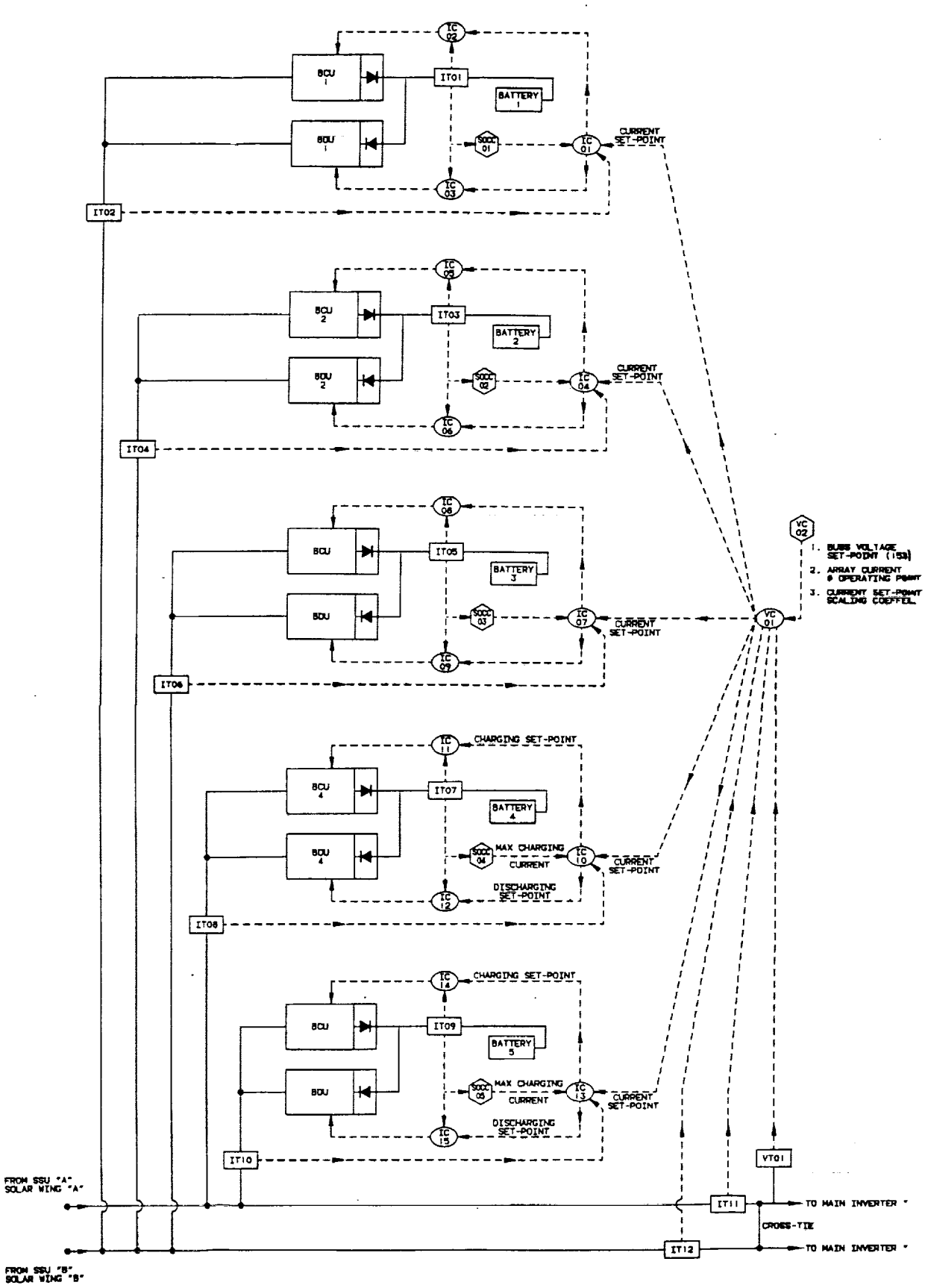

Figure 16: BCDU Schematic

maximum bus voltage and charging current to control the amount of discharging or charging for each battery. Detailed explanation of the control scheme is not important for this example and will not be discussed.

FMEA

A FMEA was performed on the BCDU system to define the failure knowledge used by TROUBLE III in diagnosing the system. For each component of the BCDU system (ie VC01, IT01, IC03)[a] every possible cause and symptom for every possible failure was described. Each unique failure-cause-symptom was stored as a unique failure object. Once all the failure modes and causes were identified, a failure hypothesis tree was created that shows how the failure modes and causes are interconnected.

Figure 17 shows one branch of the failure hypothesis tree created for the BCDU system. Each node of the tree that has children (branches coming out of the right side) is a failure mode and each of its children is a possible cause for that failure. As can be seen, a failure mode can be a cause (child) of another failure mode. In figure 17 the failure mode "NO DISCHARGE CURRENT" has six possible causes, one of which, "NO SIGNAL FROM IC03", is itself a failure mode with four possible causes. In figure 17 specific device names such as IC03, IC01, etc. were used so that the reader could easily locate the failures on the schematic. The real tree is more general so that it can be used for all five batteries. The device names are replaced with the device function. For example IC03, IC06, IC09, IC12 and IC15 are all the same device, just a different battery, so in the tree, IC03 would be replaced with "DISCHARGE SET POINT IC" to make it general enough to include the other four batteries. The failure tree is not stored in TROUBLE III but each node is, so that TROUBLE III can generate all or part of the tree depending upon the state of the system.

[a]VC = Voltage Controller, IC = Current Controller, VT = Voltage Transducer, IT = Current Transducer

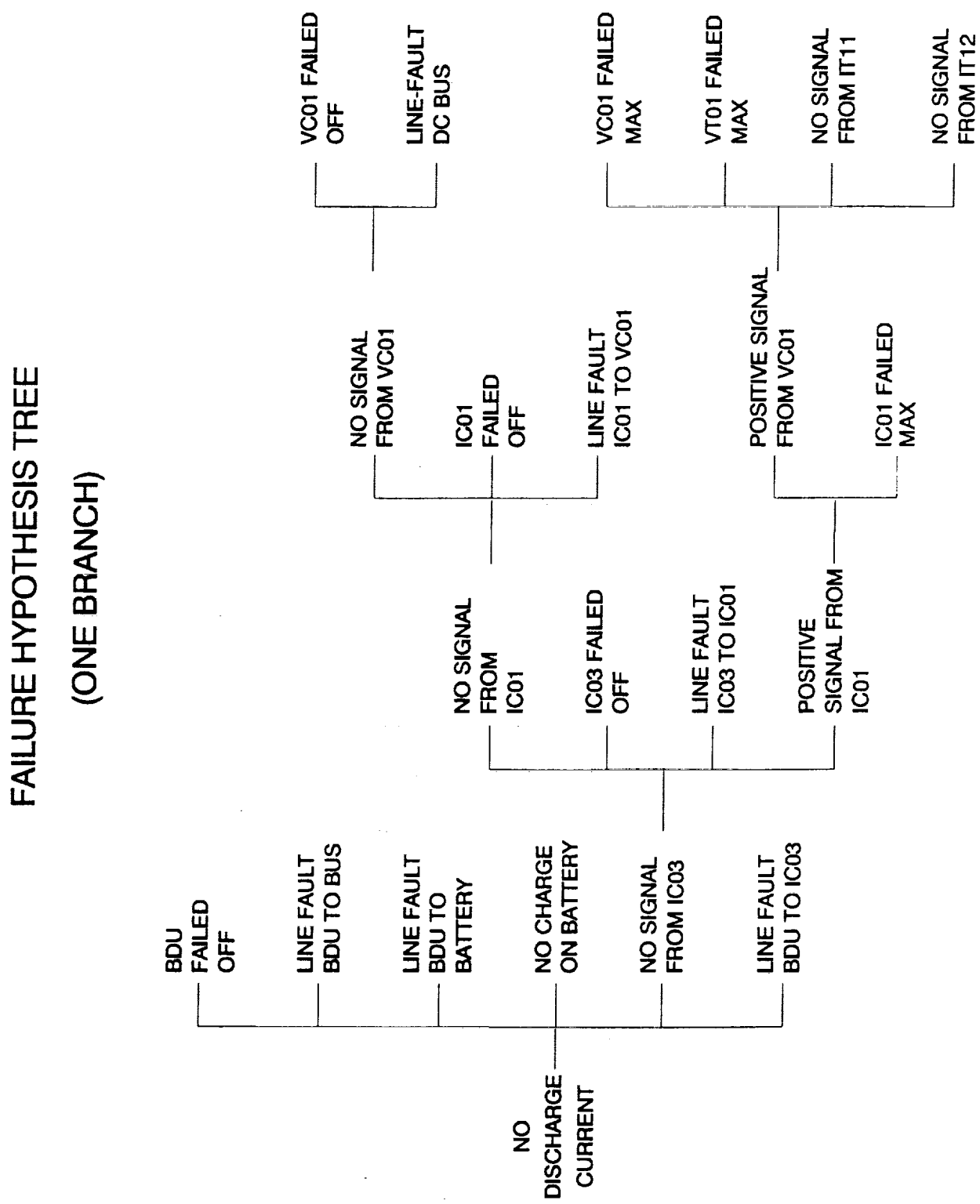

Figure 17: Failure Hypothesis Tree

Detection

The first module in TROUBLE III is the detection module which converts the quantitative measurements into qualitative symptoms that describe the current state of the system.

For this example, the conditions in Table 2 were input into TROUBLE III and stored in the measurement objects. These values were selected in such a way that they will simulate known failures to TROUBLE III. Figure 13 shows the five symptoms that were detected by TROUBLE III when given the test data.

Table 2: Test Conditions

DEVICE	VALUE	MEASUREMENT NUMBER
VC01	22.5	4
IT05	0	17
IT06	0	18
IC07	5.4	19
IC09	0	21

The creation of the symptom "NO OUTPUT FROM DISCHARGE SET POINT IC", as shown by figure 18, demonstrates the function of the detection rules. For this symptom to be present on the system one of the discharge set point ICs has to be sending no signal. Recall from before that IC03, IC06, IC09, IC12, and IC15 are all discharge set point ICs. The test data entered had the measuring device at the output of IC09 reading 0 so detection rule #2 matched this measurement object and created the symptom "NO OUTPUT FROM DISCHARGE SET POINT IC" (Figure 18). In so doing detection rule #2 converted a quantitative measurement (IC09 = 0) into a qualitative symptom (NO OUTPUT FROM DISCHARGE SET POINT IC).

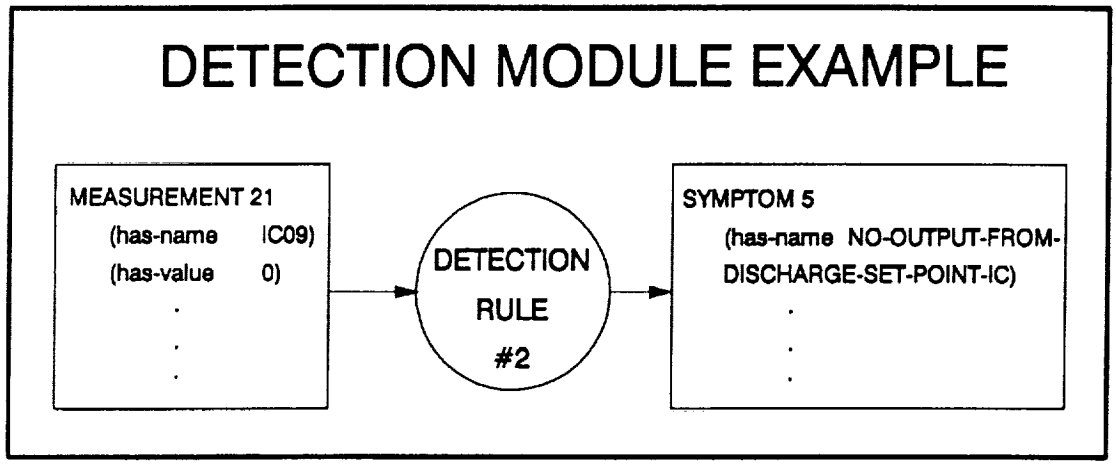

Figure 18: Detection Module Example

Failure Hypothesis Module

The failure hypothesis module is responsible for generating a list of possible failures to explain the abnormal behavior(s) detected by the detection module.

Failure objects have many attributes including failure mode, failure symptom and failure cause. The failure hypothesis module matches each detected symptom to failure objects with that symptom. If the reason object of the matched failure object has value inclusion maintained or exclusion defeated, a failure hypothesis record (FHR) is created (Figure 10).

In the test case being discussed, the five detected symptoms produced 18 possible FHRs to describe the observed abnormal behavior. Figure 19 shows the failure mode and cause of each of the FHRs created. Every failure mode in Figure 19 appears as a parent node and every cause appears as a child node in the failure hypothesis tree (Figure 17). Figure 19 uses the general device purpose instead of the device name that Figure 17 uses.

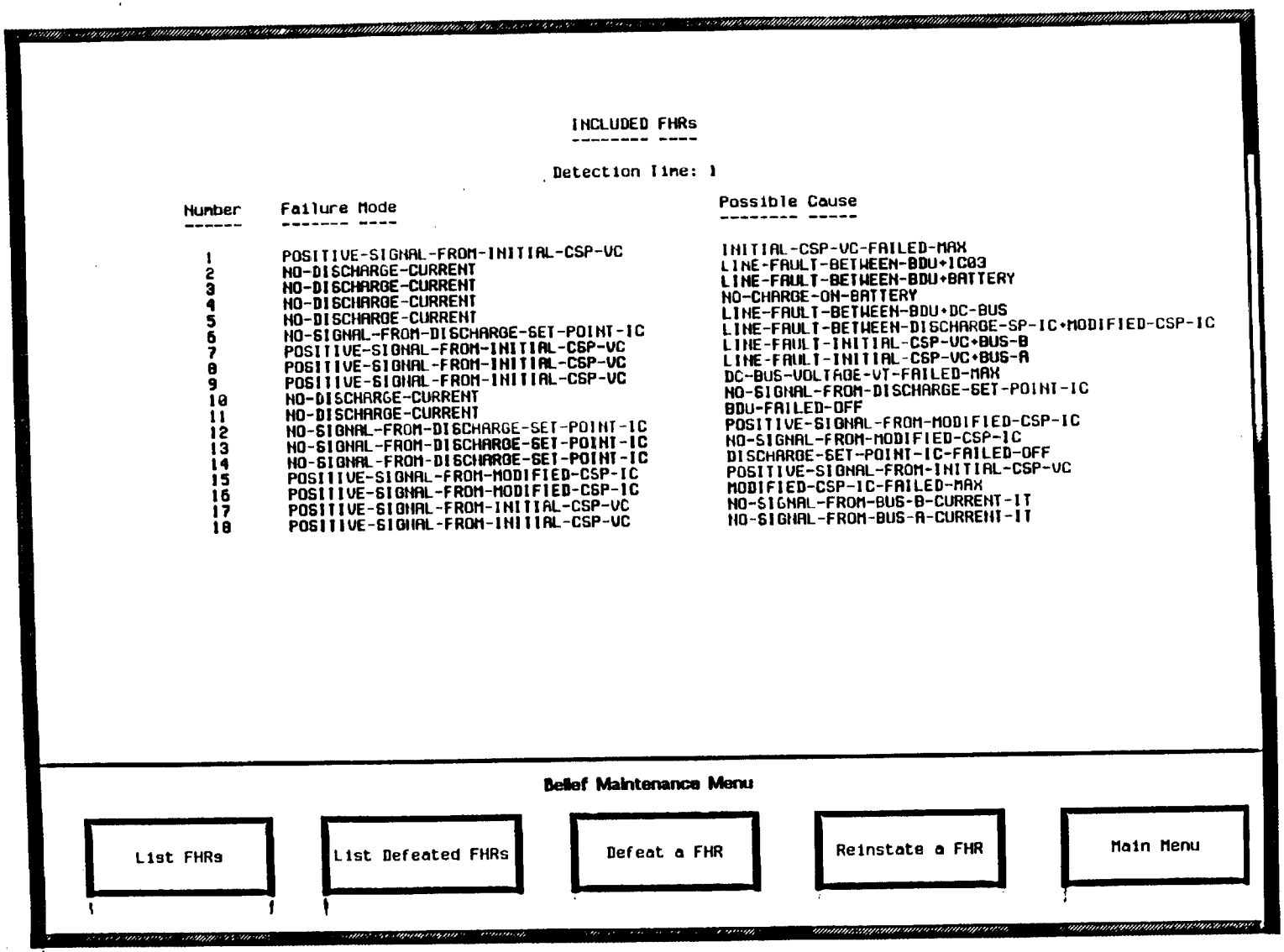

Figure 19: Listed FHRs

Concentrating on the symptom "NO OUTPUT FROM BDU" the FHR rule can be demonstrated. In this example the symptom "NO OUTPUT FROM BDU" is the failure symptom for the failure mode "NO DISCHARGE CURRENT". Therefore, this symptom matched all six causes for the failure mode "NO DISCHARGE CURRENT". Figure 20 shows how the FHR rule takes this symptom and failure to create the FHR with the cause "NO CHARGE ON BATTERY".

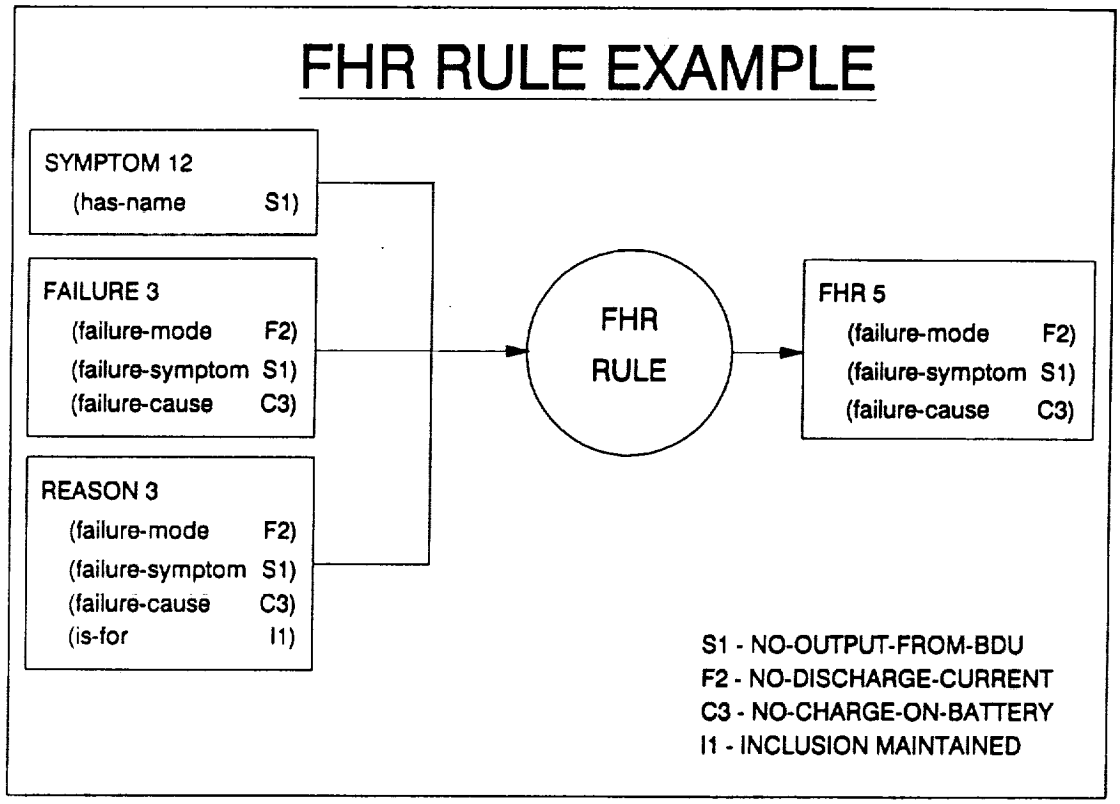

Figure 20: FHR Rule Example

Chaining Module

The chaining module is responsible for taking the FHRs and linking them together to form the portion of the failure hypothesis tree that describes the current state of the system. The chaining module creates links between each

parent and child in the tree, then uses these relationships to form the chains (lists of FHRs) that go from a top level failure to a root cause. A root cause is a cause that has no children.

Figure 21 shows how two chains in this example were created. Given the 18 FHRs of the example, the chaining module produced 15 possible failure chains or root causes for the failure mode "NO DISCHARGE CURRENT" (Figure 14). These correspond to the different paths in the failure hypothesis tree from the top failure MODE down to each root cause. In this example the part of the tree stemming from "NO SIGNAL FROM IC01" has been pruned away because IC07 (IC01's counterpart) has a value so the failure mode "NO SIGNAL FROM IC01" is not detected and the tree from that point on is not developed in the FHRs. Figure 14 gives the 15 possible failure chains ranked according to related modes with different symptoms.

CHAINING EXAMPLE

Given these FHR's . . .

FHR 1
(failure-mode NO-DISCHARGE-CURRENT)
(cause NO-SIGNAL-FROM-IC03)

FHR 2
(failure-mode NO-SIGNAL-FROM-IC03
(cause NO-SIGNAL-FROM-IC01)

FHR 3
(failure-mode NO-SIGNAL-FROM-IC03)
(cause IC03-FAILED-OFF)

FHR 4
(failure-mode NO-SIGNAL-FROM-IC01)
(cause IC01-FAILED-OFF)

These chains would be formed:

	CHAIN 1	CHAIN 2
Chain links	FHR1-FHR3	FHR1-FHR2-FHR4
Top level failure	NO-DISCHARGE-CURRENT	NO-DISCHARGE-CURRENT
Possible root cause	IC03-FAILED-OFF	IC01-FAILED-OFF

Figure 21: Chaining Example

Justification Module

The justification module justifies a user selected chain's existence by displaying the logic used by TROUBLE III in creating the chain.

Chain 4 was selected to be justified in this example and the results appear in Figure 15. If one compares the justification of chain 4 to the failure hypothesis tree, they would see that the justification module displays a branch of the tree, from top failure to root cause, and explains in English how "DC BUS VOLTAGE VT FAILED MAX" could be a root cause of the failure "NO DISCHARGE CURRENT".

Looking at Figure 14 also shows how chain 4 got a RMWDS of four. If one counts the number of different symptoms that appear in the justifications, they will come up with four RMWDS.

Belief Maintenance

The belief maintenance module allows a user to defeat or reinstate any FHR. If the user desires to see the chaining results if a selected FHR never existed, he/she can defeat that particular FHR causing all chains containing that FHR to disappear. This has the effect of pruning the failure hypothesis tree at the desired node.

In this example, FHR 10 was defeated and Figure 22 displays the resulting chains. Comparing this to the failure hypothesis tree one can see that defeating FHR 10 has the same affect as pruning the tree at the "NO SIGNAL FROM IC03" node. Only the five chains that do not contain that failure mode are displayed. If FHR 10 was reinstated all the original chains would reappear.

Figure 22: Defeated Chains Output

CONCLUSION

This thesis has described an object oriented expert system TROUBLE III, which is intended to automate the fault diagnostics of Space Station Freedom's power system.

It has been shown that this type of automation is desirable in a space based system to reduce the number of needed personnel, increase reliability and speed up response time to faults.

The design of TROUBLE III was complicated by the fact that Space Station Freedom's power system design keeps changing. Adapting an expert system to this dynamic, evolving system requires a flexible design. Three types of expert systems were studied: rule-based, set-covering, and model-based. A set-covering approach was selected because it offered the needed flexibility that rule-based and model-based systems lack. In set-covering the flexibility is gained because failure knowledge about the system is stored in databases rather than encoded in rules.

Besides using a set-covering approach, TROUBLE III makes use of the following techniques: Failure Modes and Effects Analysis (FMEA), Petri net transitions, reasoned assumptions and Related Modes With Different Symptoms (RMWDS). FMEA is used to produce the known failure database. Transitions control the flow of the program by determining which module to make active, while reasoned assumptions and RMWDS are methods that reduce the work necessary to determine the most probable cause.

Besides expanding Zisman's work with Petri net transitions, an important outcome of TROUBLE III's development was that it demonstrated it was possible to combine the strengths of set-covering, FMEA, Petri net transitions, and reasoned assumptions into one diagnostic expert system. These techniques are not new ideas individually but combining them all together is unique.

TROUBLE III's main strength is its quick adaptability to any system changes. By performing a FMEA on the new system and changing the detection rules, TROUBLE III can be adapted to handle diagnostics on any system. Failure knowledge not being hard coded into rules means no diagnostic rules need be altered.

As with any rule-based or set-covering expert system, the main limitation of TROUBLE III is that it can only detect faults that have been described to it by a human expert. It cannot make inferences. If a fault appears on the system that was not described by the human expert, TROUBLE III has no

knowledge of that fault and it will go undetected. The failure knowledge of TROUBLE III is limited by the human expert. The more comprehensive the human expert is in identifying the possible failures and causes of the system, the better TROUBLE III becomes.

Another limitation of TROUBLE III is that it will not detect incipient faults, only static faults. TROUBLE III does not have the capability to monitor the performance history of each component in an attempt to predict and negate possible failures before they occur. It can only detect and isolate faults that have already occurred on the system.

A prototype battery charge-discharge control system was designed to test TROUBLE III. A FMEA was performed on this design with the consequent failure modes and causes being stored in a failure database. A set of detection rules was written to detect the symptoms of these failure modes. Test cases were developed to simulate certain failure modes and the test results were compared to the predicted results to verify the rule's logic.

TROUBLE III is a functioning diagnostic expert system but its development is not complete. Several improvements are planned to make TROUBLE III a better system: integrating TROUBLE III with a test bed, adding additional ranking schemes, improving the graphics, and interconnecting TROUBLE III with other expert systems.

Even though TROUBLE III has been designed specifically to handle Space Station Freedom's dynamic, evolving power system, it is not limited to this specific application. The flexibility that had to be incorporated in the design to handle the changing power system makes TROUBLE III easily adaptable to any diagnostic system on Earth or in space.

REFERENCES

1. Biglari, H. and Cheng, C. and Vachtsevanos, G.; "Fault Tolerant Intelligent Controller For Space Station Subsystems"; 1988 IECEC Proceedings of the 23rd Intersociety Energy Conversion Engineering Conference; Denver, CO.; July 31 - Aug 5 1988; Vol. 3 pp 313-318

2. Cardozo, E. and Talukdar, S.; "A Distributed Expert System For Fault Diagnosis"; IEEE TRANSACTIONS on Power Systems; Vol. 3, No. 2; pp 641-646; May 1988

3. Culbert, C. and Boarnet, M.; "AI Applications for the Space Station"; Elsevier Science Publishers B.V. (North-Holland); Robotics 4; pp 35-40; 1988

4. D'Ambrosio, A.; "PERF-EXS, An Expert System For Diagnosing Power Plants: Design Criteria and Implementation Details."; Proceedings 3rd Annual Expert Systems Conference and Exposition; Detroit, MI; April 4-6, 1989; pp 83-94

5. Davis, R. and Hamscher, W.; "Model-Based Reasoning: Troubleshooting"; M.I.T. Artificial Intelligence Laboratory, Cambridge, MA.; July, 1988

6. Dolce, J.; "An Integrated Approach to Space Station Power System Autonomous Control"; IECEC '87 Proceedings of the 22nd Intersociety Energy Conversion Engineering Conference; Philadelphia, PA; Vol. 1; pp 499-508; August 10-14, 1987

7. Dolce, J.; Private Communications; August, 1989

8. Dolce, J. and Faymon, K.; "Automating the U.S. Space Station's Electrical Power System"; Optical Engineering; Vol. 25, No. 11 pp 1181-1185; November, 1986

9. Dolce, J. and Mellor, P. and Kish, J.; "Automated Electric Power Management and Control for Space Station Freedom"; 25th Intersociety Energy Conversion Conference; Reno, Nevada; August 12-17, 1990

10. Doyle, J.; "Methodological Simplicity in Expert System Construction: The Case of Judgements and Reasoned Assumptions."; The AI Magazine; Summer 1983; pp 39-43

11. Fesq, L. and Stephan, A.; " Advances in Spacecraft Autonomy Using Artificial Intelligence Techniques"; TRW Space and Technology Group, Redondo Beach, CA

12. Freeman, K. and Walsh, R.; "Concurrent Development of Fault Management Hardware and Software in the SSM/PMAD"; 1988 IECEC Proceedings of the 23rd Intersociety Energy Conversion Engineering Conference; Denver, CO.; Vol 3; pp 307-312; July 31 -August 5, 1988

13. Lee, S. and Lollar, L.; "Development of a Component Centered Fault Monitoring and Diagnosis Knowledge Based System for Space Power System"; 1988 IECEC Proceeding of the 23rd Intersociety Energy Conversion Engineering Conference; Denver, CO.; Vol.3; pp 377-382; July 31 - August 5, 1988

14. Leinweber, D.; "Expert Systems in Space"; IEEE Expert; pp 26-36; Spring 1987

15. Milne, R.; "Strategies for Diagnosis"; IEEE Transactions on Systems, Man, and Cybernetics"; Vol. SMC-17, No 3; pp 333-339; May/June 1987

16. Omarali, Z. and Savage, G.; "Efficiency Considerations in the Design of Expert Diagnostic Systems"; Proceedings 3rd Annual Expert Systems Conference and Exposition; Detroit, MI.; pp 15-24; April 4-6, 1989

17. Šobajić, D. and Pao, Y.; "An Artificial Intelligence System for Power System Contingency Screening"; IEEE Transaction on Power Systems; Vol. 3, No. 2; pp 647-652; May 1988

18. Teren, F.; "Space Station Electric Power System Requirements and Design"; IECEC '87 Proceedings of the 22nd Intersociety Energy Conversion Engineering Conference; Philadelphia, PA; Vol. 1; pp 39-47; August 10-14, 1987

19. Yoon, W. and Hammer, J.; "Deep-Reasoning Fault Diagnosis: An Aid and a Model" IEEE Transactions on Systems, Man and Cybernetics; Vol. 18, No. 4; pp 659-675; July/August 1988

20. Walls, B.; "STARR: An Expert System For Failure Diagnosis In a Space Based Power System"; 1988 IECEC Proceedings of the 23rd Intersociety Energy Conversion Engineering Conference; Vol. 3;

pp 303-306; July 31 - August 5, 1988

21. Wang, C., Zeanah, H., Anderson, A., Patrick, C., Brady, M., and Ford, D.,; "Automatic Detection of Electric Power Troubles (ADEPT)"; NASA, Marshal Space Flight Center; Huntsville, AL.; 1988

22. Weeks, D.; "Automation of the Space Station Core Module Power Management and Distribution System"; NASA, Marshal Space Flight Center; Huntsville, AL.; 1988

23. Zisman, M.; "Use of Production SYstems for Modeling Asynchronous Concurrent Processes"; in *Pattern-Directed Inference Systems*; Waterman, D.A. and Hayes-Roth, F.; eds.; Academic Press; New York; 1978; pp 53-68

24. Unpublished Document; "Chronology of Space Station Freedom Power Distribution System"; Space Station Freedom Electricla Systems Division, NASA Lewis Research Center; Cleveland, OH; 1990

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR - 187113		

4. Title and Subtitle
TROUBLE III: A Fault Diagnostic Expert System for Space Station Freedom's Power System

5. Report Date

6. Performing Organization Code

7. Author(s)
David B. Manner

8. Performing Organization Report No.
None (E - 6188)

10. Work Unit No.
488 - 51 - 03

9. Performing Organization Name and Address
Sverdrup Technology, Inc.
Lewis Research Center Group
2001 Aerospace Parkway
Brook Park, Ohio 44142

11. Contract or Grant No.
NAS3 - 25266

13. Type of Report and Period Covered
Contractor Report
Final

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135 - 3191

14. Sponsoring Agency Code

15. Supplementary Notes
Project Manager, Pam Mellor, Electrical Systems Division, NASA Lewis Research Center, (216) 433-8328.

16. Abstract
Designing Space Station Freedom has given NASA many opportunities to develop expert systems that automate onboard operations of space based systems. This paper describes one such development, TROUBLE III, an expert system that has been designed to automate the fault diagnostics of Space Station Freedom's electric power system. TROUBLE III's design is complicated by the fact that Space Station Freedom's power system is evolving and changing. TROUBLE III had to be made flexible enough to handle system changes with minimal changes to the program. Three types of expert systems were studied: rule-based, set-covering and model-based. A set-covering approach was selected for TROUBLE III because it offered the needed flexibility that was missing from the other approaches. With this flexibility, TROUBLE III is not limited to Space Station Freedom applications, it can easily be adapted to handle any diagnostic system.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Expert systems Faults Diagnosis	Unclassified - Unlimited Subject Category 18

19. Security Classif. (of the report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	54	A04

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161