Application of MODEL BASED REASONING to Diagnosis of Faults in Inertial Navigation Equipment

THESIS

Raymond E. Yost
Captain, USAF

AFIT/GCS/ENG/89-D-19

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/89-D

Application of MODEL BASED REASONING to Diagnosis of Faults in Inertial Navigation Equipment

THESIS

Raymond E. Yost
Captain, USAF

AFIT/GCS/ENG/89-D-19

AFIT/GCS/ENG/89-D

Application of MODEL BASED REASONING to Diagnosis of Faults in Inertial Navigation Equipment

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science (Computer Systems)

Raymond E. Yost, A.S. B.S.

Captain, USAF

December 1989

Acknowledgments

There are four sorts of men:

He who knows not and knows not he knows not:
he is a fool - shun him;
He who knows not and knows he knows not:
he is simple - teach him;
He who knows and knows not he knows:
he is asleep - wake him;
He who knows and knows he knows:
he is wise - follow him

- LADY BURTON, In Life of Sir Richard Burton

Hopefully I have become wiser in conducting, and to no lesser degree in completing, the research associated with this thesis.

I began this thesis under more adversities than have plagued any other project I have ever undertaken. However, thanks to the unrelenting efforts of Lt Col Charles Bisbee, my advisor, and Capt Eric Hanson, my sponsor, I accomplished more than anticipated. I am convinced that the adversities thrust in our path during the initial stages of this thesis were, to a degree, contributors to its success. They instilled in me a perseverance which had dwindled after a rigorous 12 month curriculum. I am indebted to Lt Col Bisbee and Capt Hanson for their encouragement throughout this work. I also wish to extend my appreciation to Professor F.M. Brown for the academic tenacity he exhibited in my earlier graduate courses.

Finally, I wish to thank my parents, Gerald and Anna May Yost, whose love and encouragement over the years have enabled me to accomplish things I never imagined.

Raymond E. Yost

Table of Contents

List of Figures

Abstract

In 1988 Skinner produced the Blended Diagnostic System (BDS) which was an attempt to provide the Aerospace Guidance and Metrology Center (AGMC) with an expert system capable of diagnosing faults in the Dual Miniature Inertial Navigation System (DMINS). Skinner proposed the blending of a traditional rule-based system with a model-based system. The techniques used to perform the model-based reasoning in BDS are however primitive compared to other techniques currently available. This thesis describes the development of a model-based diagnostic system using techniques pioneered by Randall Davis, and substantially more sophisticated than those used in BDS. A diagnostic prototype for the DMINS was developed which provides a more thorough and consistent diagnosis than does Skinner's model-based system.

The models in our prototype were created using the Intelligent Diagnostic Expert Assistant (IDEA) software developed by AI Squared Inc., of North Chelmsford MA. IDEA is based on extensive research by Davis at the Massachusetts Institute of Technology (MIT).

Application of

MODEL BASED REASONING to

Diagnosis of Faults in Inertial Navigation Equipment

I. Introduction

This thesis describes a research project which investigates the application of model-based reasoning to the diagnosis of faults in Inertial Measuring Units (IMUs). At least one documented attempt has been made to apply model-based techniques to the diagnosis of faults in IMU's (26). Though successful, the resulting system was implemented using rule-based expert system technology and as a result only dealt with the structural aspects of the IMU. True model-based systems incorporate both structural as well as behavioral characteristics of the problem domain. This thesis presents a prototype system which diagnoses faults in Inertial Measuring Units using a model-based approach incorporating both structural and behavioral knowledge of inertial navigation equipment.

This chapter covers background material related to this research effort. It includes a brief description of the fundamental issues inherent in expert system development and in particular how these issues affect fault diagnosis. In addition, this chapter introduces the particular problem to be addressed as well as the relevant research conducted to date. Finally, a brief description of the methodology to be followed in conducting this research is presented.

1.1 Background

In recent years the AI community has seen an increase in the development of diagnostic systems, both rule-based and model-based. Expert Systems for diagnosing everything from infectious blood diseases (25),to liver cholestasis (4), to malfunctions in diesel electric locomotives (27), are now commonplace. A predominant number of AI based diagnostic systems in use today may be classified as heuristic rule-based systems. The relatively few model-based systems discussed in the current literature have typically been implemented from scratch (27) or in rule-based expert system shells ill-equipped to capture both the structural and functional aspects of the problem domain (26). The substantial number of rule-based diagnostic applications has, however, provided invaluable insight into some of the inherent limitations of traditional expert systems.

In the classic rule-based expert system, the developer's primary goal is to represent the heuristics an expert uses to solve a specific problem in such a manner as to make that knowledge available to other technicians. Representing the expert's knowledge is accomplished by coding fault manifestations into IF..THEN constructs. For instance, if an expert has encountered a problem, P, that occurs when evidence A, B and C exist, a rule-based system would encode that as follows:

```
IF fact A
   fact B
   fact C
THEN
    problem P
```

This strategy allows the expert system to codify the knowledge required to identify

any given anomaly. In essence, rule-based diagnostic systems tend to rely on failure histories to predict future failures. This is one of the most important aspects of expert systems. A problem arises, however, when a fault occurs that has never been previously encountered. Rule-based systems are unable to diagnose these unanticipated failures since their knowledge only covers faults explicitly represented in the rules. This phenomenon results in incomplete coverage of the particular problem domain and has often created problems in the field of diagnosis. It is this requirement, that the expert system developer be aware of all expected fault behaviors, which has prompted researchers to investigate alternative diagnostic strategies. One such strategy is model-based diagnosis, often referred to as deep-reasoning or diagnosis based on "first principles".

Model-based diagnostic systems attempt to alleviate the above scenario by taking a different approach altogether. Rather than enumerating all possible faults, model-based systems reason from a model of the system or device under consideration. By comparing observed system behaviors with behaviors predicted by the model, the reasoning mechanism identifies points where the predicted value and observed value differ. These differences, or discrepancies, are subsequently used to guide the search for faulty components. This eliminates the need to explicitly identify all possible faults and is more likely to provide more complete fault coverage than current rule-based systems (5).

In addition, while rule-based diagnostic systems may use knowledge relating to device structure and function to solve problems, such knowledge is often inextricable from the problem solving heuristics themselves. This information, regarding structure and function, may be implicitly embedded in the rules but is often transparent to the user. Understanding the knowledge contained in the rules is often extremely difficult. This makes the subsequent

system maintenance even more so.

The model-based approach tends to be relatively device independent as opposed to the highly device dependent nature of the rule-based approach. This device dependency stems from the fact that rule-based systems are aimed at identifying explicit faults. Similar devices may exhibit different failure characteristics thereby requiring a unique rule set for each separate device. Small changes to the device can also lead to dramatic changes to the existing rule base (9). The impact of these device changes on system performance is often difficult to determine and minimize, regardless of how subtle the change. In the model-based approach, because the actual physical structure of the device is represented, changes to the device can be more readily incorporated into the diagnostic system. It is this device independence which may allow model-based systems to diagnose failures in a general class of devices, since specific failure characteristics need not be coded.

Since the fundamental concept of rule-based expert systems is the acquisition of expert knowledge, such systems often demand a substantial amount of experience in terms of the types of problems being diagnosed. This requires a considerable amount of time in acquiring that troubleshooting experience prior to developing a useful system (22). Model-based system advocates believe a reduction in this acquisition time could be achieved if efforts were focused on capturing structural and behavioral knowledge of the particular problem domain. By using engineering documentation such as circuit schematics, maintenance manuals and manuals describing the system's theory of operation, much of the information required to construct a model can be obtained without the perpetual use of knowledge engineering interviews characteristic of traditional expert system development. Using design data obtained from engineering documentation, a device model is created and

used to predict device behaviors. The reasoning strategy compares the predicted behaviors with observed device behaviors. Differences between the behaviors indicate a fault and are used to determine what components could have contributed to the fault.

This is not to imply that experts are not needed, it simply means that we are more concerned with the proper functioning of a given system rather than how an expert goes about identifying specific malfunctions. Alleviating the need to have uninterrupted access to a system expert should speed the development process.

In summary, the application of model-based diagnosis could minimize the lengthy knowledge acquisition process inherent in rule-based system development, and provide more complete fault coverage. Furthermore, diagnostic systems could conceivably be developed in conjunction with the actual device, thereby providing an immediate diagnostic capability upon completion of the device. It is this characteristic which has led to further research into model-based systems and the tools to support their development.

1.2 Problem Statement

The system chosen to investigate model-based diagnosis is the Dual Miniature Inertial Navigation System (DMINS). The DMINS is an inertial navigation system used on fast attack submarines, oceanographic survey ships and aircraft carriers. Currently, DMINS inertial measuring units (IMUs) found to be defective at sea are shipped to the Aerospace Guidance and Metrology Center (AGMC), located at Newark AFB, OH, for service. Maintenance of these units is an expensive and time-consuming mission for the technicians and engineers at AGMC. In an attempt to enhance the diagnostic capability of the technicians, and to capture the repair expertise of these personnel, AGMC decided to investigate the

applicability of expert system technology in the depot level repair of IMU's.

Initial research in the application of artificial intelligence techniques to the diagnosis of faults in the DMINS inertial measuring unit resulted in a system called the Blended Diagnostic System (BDS), developed by Capt James Skinner as a 1988 Masters Thesis during his enrollment at the Air Force Institute of Technology (AFIT) (26). The aim of BDS was twofold:

1. to explore the feasibility of implementing expert system technology to the depot level repair of IMUs.
2. to investigate the blending of a rule-based diagnostic system and a model-based diagnostic system.

The resulting system was very well received by the engineers at AGMC and is currently being completed for future integration into the depot level repair of the DMINS (24). However, the model-based portion of Skinner's system only dealt with the structural aspects of inertial measuring units. That is to say it only possessed rudimentary knowledge of how the IMU components are connected. Actual functional relationships among components and behavioral knowledge of individual components were not included. Hence when using the model-based portion of the system, BDS would always initiate a point-by-point search through the connections of the IMU components. This is one of the most primitive search techniques available and exhaustively enumerates all model components as possible suspects. Other techniques are available which are more sophisticated and efficient. In addition, BDS provided little insight as to what characterizes a good output versus a bad output. Since the ultimate goal of expert systems is to provide a consistent, intelligent, diagnostic capability it is imperative that such a system possess knowledge concerning valid

behaviors. To simply ask, **"Is the output of circuit X good?"** leaves significant room for misinterpretation. The resulting system should be a central repository of diagnostic and training information.

The intent of this research effort is to extend the work begun by Skinner by investigating the applicability of a full model-based system for use in the repair of the dual miniature inertial navigation system. Specific attention was focused on determining the appropriate level of abstraction necessary to conduct the diagnosis, and to provide a system suitable for novice technicians as well as veteran experts. Keeping with this goal the prototype exploits the use of high resolution color graphics in displaying information and requests to the technician.

A newly developed software tool called the Intelligent Diagnostic Expert Assistant (IDEA) was used in developing the prototype. IDEA was developed by AI Squared,Inc. of North Chelmsford Massachusetts, and is based on the research efforts of Dr Randall Davis of the Massachusetts Institute of Technology (19).

1.3 Scope and Approach

Primary sources of technical information for the development of the prototype were the DMINS organizational and depot level repair manuals. The organizational level manual presents a functional decomposition of the DMINS system which was used in scoping this research to a manageable level. The functions relating to the Inertial Measuring Unit (IMU) include the following:

- 115vac 400Hz Power Distribution

- +28v, +42v Power Distribution Function
- +/- 6v, 24v, 48v Power Distribution Function
- +/- 12v Power Distribution Function
- Platform / Gyro Temperature Control Function
- Frequency Standard Function
- Gyro Speed Control Function
- Gyro Torquing Function
- Reference 400Hz Function
- Platform Torquing Function
- Velocity Meter Function

The Reference 400Hz and Frequency Standard functions were chosen for implementation in the prototype. These functions include a relatively high degree of complexity and possess interesting features relating to circuit feedback making them noteworthy candidates in this research. In addition, the four power-distribution functions were incorporated to the degree that the user has the option of specifying whether or not the power distribution networks could have caused the fault being diagnosed. If not, the power distribution infrastructure is assumed operational and the system will not inquire about it. The Dual Miniature Inertial Navigation System (DMINS) Inertial Measuring Unit (IMU) can be decomposed into 38 shop replaceable units (SRUs) as illustrated in Figure 1. The technician is responsible for isolating the fault to one or more of these 38 SRUs. Initially, only those components composing the functions targeted for implementation will be represented in the model. Those modules are listed in Figure 2.

Shop Replaceable Unit (SRU)	SRU Designator
BandPass Filter and Shift Register (X-Y)	A1
BandPass Filter and Shift Register (Y-Z)	A2
Precision Torquing Driver (X)	A3
Precision Torquing Driver (Y)	A4
Precision Torquing Driver (Z)	A5
Platform Electronic Switch	A7
Shorting Plug	A8
Precision Current Network	A9
Stable Platform Assembly	A10
Displacement Gyroscope(X-Y)	A10A3
Displacement Gyroscope(Y-Z)	A10A4
Velocity Meter(X)	3A10A7
Velocity Meter(Y)	3A10A8
Resolver Buffer Electronic Control Amp	3A10AR1
Gyro Buffer Electronic Control Amp(X-Y)	3A10AR5
Gyro Buffer Electronic Control Amp(Y-Z)	3A10AR6
DC Amplifier(X-Y)	AR1
DC Amplifier(Y-Z)	AR2
Synchro Signal Buffer Amplifier	AR3
Gyro Cage Amplifier	AR4
Thermoelectric Signal Amplifier	AR5
Gyro Temperature Controller	AR6
Gimbal Cage Amplifier	AR7
Platform Signal Amplifier	AR8
Platform Electronic Control Amplifier (roll)	AR9
Platform Electronic Control Amplifier (pitch)	AR10
Platform Electronic Control Amplifier (azimuth)	AR11
Gimbal Rate Electronic Control Amplifier (roll)	AR12
Gimbal Rate Electronic Control Amplifier (pitch)	AR13
Gimbal Rate Electronic Control Amplifier (azimuth)	AR14
640KHz Power Supply(X-Y)	PS1
640KHz Power Supply(Y-Z)	PS2
Power Cube	PS3
400KHz Power Supply 1	PS8
400KHz Power Supply 2	PS7
Triangle Generator and Case Rotation Power Supply	PS9
4.8KHz Power Supply	PS10
Frequency Standard	PS11

Figure 1. DMINS SRU Configuration

Shop Replaceable Unit (SRU)	SRU Designator
Power Cube	PS3
400KHz Power Supply 1	PS8
400KHz Power Supply 2	PS7
Triangle Generator and Case Rotation Power Supply	PS9
Synchro Signal Buffer Amplifier	AR3
4.8KHz Power Supply	PS10
Displacement Gyroscope(X-Y)	A10A3
Displacement Gyroscope(Y-Z)	A10A4
Velocity Meter(X)	3A10A7
Velocity Meter(Y)	3A10A8
Frequency Standard	PS11
Capacitors	n/a
Transformers	n/a

Figure 2. SRUs Targeted for Implementation

The basic methodology to be followed in developing the prototype is as follows:

1. Determine the level of abstraction necessary to adequately diagnose IMU failures.
2. For the SRUs targeted for implementation, develop and encode behavioral models at the appropriate level of abstraction.
3. Use the development environment in IDEA to encode the connectivity of the modules.
4. Develop and integrate a user interface to the diagnostic prototype.

1.4 Overview

A great deal of the current research in AI based diagnosis is aimed at the development of model-based systems as well as the methodologies and tools necessary for their development (19). Chapter 2 describes some applications of artificial intelligence in the field of diagnosis and summarizes the state of expert systems. Chapter 3 explains the concept of "First Principles" and explains the model-based strategy used in this prototype. Chapter 4 provides a detailed description of the DMINS system and the current

troubleshooting philosophy employed by the Aerospace Guidance and Metrology Center. Chapter 5 documents the bulk of this research and describes the actual development stages of the prototype, whereas Chapter 6 discusses some problems experienced during that development. Finally, Chapter 7 summarizes the research and provides some ideas for future work.

II. Review of Literature

2.1 Introduction

In recent years the AI community has seen an increased interest in the development and application of diagnostic expert systems. The primary focus has been on the use of rule-based systems, though interest has begun to shift toward alternative strategies. This chapter surveys diagnostic expert systems, progressing from the more popular rule-based applications to comparatively new model-based systems.

Webster defines diagnosis as: "the art or act of identifying a disease from its signs and symptoms" (13). However, in past years, this term has also been widely applied to electrical and mechanical equipment and components. Waterman states that diagnosis uses: "situation descriptions, behavior characteristics, or knowledge about component design to infer probable causes of system malfunctions" (27). This latter definition forms the basis for a number of diagnostic expert systems in use today, which as noted by Merritt are " ... some of the most visible and easily justifiable applications in todays computerlandscape "(21). Not only are such systems the most visible applications of artificial intelligence (AI), they also represent one of the first breakthroughs in transitioning AI from the research centers to the commercial sector. For this reason much attention has been focused on the successful fielding of such applications. The objective of this chapter is to survey some typical diagnostic expert systems and to provide some insight as to the direction of their development.

Numerous strategies have been applied to fault diagnosis, some more successfully than others. To gain an appreciation for the applicability of model-based diagnosis it is

worthwhile to familiarize ourselves with these traditional strategies.

2.1.1 Diagnostics. One of the earliest strategies used to determine the cause of a system malfunction was the use of diagnostic programs. These programs were run to ensure that a given system or device was capable of performing all of its intended functions. The problem with diagnostics however is twofold. First, diagnostics, as noted by Davis (5), do not perform diagnosis, rather they do verification. Diagnostics are designed solely to ensure that a device or system does what it is supposed to do. Traditionally this strategy resorts to exhaustively testing a device to verify correct operation. In diagnosis, on the other hand, we are presented with an entirely different problem, namely, given an observed anomaly, what caused it. Second, diagnostics tend to be more a practice in test case generation rather than diagnosis. Diagnostics are designed to test a given device to ensure the absence of faults. This requires the device be exercised in all conceivable and often inconceivable ways in order to detect all possible failures. Accomplishing this feat, however, is no easy task. For instance, at what point does one assume that *all* possible faults have been identified? In addition, once identified, how do we determine what faults to include in the diagnostics, and once this determination is made how do we know we have covered the most important faults? To ensure thorough coverage, diagnostic programs presuppose the complete enumeration of expected faults. Inevitably diagnostics are unable to cover all possible faults. Writers of diagnostic programs traditionally concern themselves with handling a restricted set of faults. In practice it is often more efficient and effective to work from the anomaly to the underlying faults, rather than to apply diagnostics and exhaustively test the device. This notion of symptom-driven diagnosis forms the basis for

what has been termed model-based reasoning, which, opposed to traditional diagnostics, is aimed at determining the cause of an existing fault.

2.1.2 Fault Dictionaries. Explicit enumeration of anticipated faults is also fundamental to the creation of fault dictionaries. The objective is to simulate a device for each way *a specific component* or *set of components* may fail. Each simulation results in a pattern for how the *entire device* would fail, given that components fail in predicted ways. The consequence of this strategy is a list of anticipated faults along with the associated device symptoms. The resulting list can be indexed and queried to provide possible failing components, given an observed anomaly.

2.1.3 Fault Trees. Fault trees provide a systematic approach to fault diagnosis. The goal is to construct a decision tree which leads a technician through a sequence of actions to diagnose a fault. The result of each step is used as the guiding characteristic in selecting subsequent steps and forms the basis for the entire diagnostic process. One problem with this approach is that the knowledge used to develop the fault tree is inaccessible. The resulting decision tree offers little information as to why a particular diagnosis was made. The tree can be analyzed to ascertain the path of a particular diagnosis, but the knowledge concerning the development of that particular path, and in fact the entire tree, is inextricable. After the tree is developed no information exists as to why decisions inherent in the tree are what they are.

2.1.4 Expert Systems. The last approach to be discussed is the diagnostic application of artificial intelligence (AI), specifically expert systems. Traditional rule-based

systems have been developed by accumulating the experience of experts in a particular field in the form of rules. Diagnostic rules are empirical associations among fault symptoms and the underlying faults. These rules are then made available to other technicians thereby providing expert assistance in failure diagnosis. Numerous diagnostic applications of rule-based systems, however, have uncovered some limitations. One primary drawback is the sizable amount of troubleshooting experience required with a device prior to the development of a useful system. Accumulating this experience can introduce years into the development cycle of a diagnostic application, often resulting in a device being well on its way to obsolescence before the application is completed.

The remainder of this chapter presents some typical expert system applications. In addition, rule-based expert systems are compared and contrasted to model based systems in order to establish a foundation for subsequent discussions.

2.2 Rule-Based Expert Systems

A predominant number of diagnostic expert systems in use today fall into the category of rule-based systems as defined earlier (27). Waterman outlined several major application areas where expert systems seem to be particularly successful, interesting and hence very popular. Though far from complete, the list does provide insight into the diversity of the many application arenas open to expert system technology. Some primary application areas are:

COMPUTER SYSTEMS	MEDICINE
ELECTRONICS	METEOROLOGY
ENGINEERING	MILITARY SCIENCE

2.2.1 MYCIN. One of the first and most widely publicized diagnostic expert systems is MYCIN. Prior to the development of MYCIN, the artificial intelligence community was often bombarded with criticisms that only trivial problems were being solved. In the early 1970's a group at Stanford University decided to undertake a relatively complex problem, in the hope of advancing the state of expert systems. The result was MYCIN, a rule-based diagnostic system aimed at diagnosing infectious blood diseases and recommending treatment (17, 27).

The determination of the type of treatment required is highly involved and takes into consideration a variety of factors such as the type of infecting agent and its susceptibility to specific medications; the site of the infection; the infected patients weight, age, physical condition, metabolism; or whether the patient is currently taking other medication which may neutralize the effects of any recommended medication. Each of these factors alone is relatively manageable, however it is the complex interaction among the factors which makes the diagnosis difficult and often beyond the mental capacity of the physician in charge. MYCIN provides a consistent capability to collate these parameters of interest and arrive at logical conclusions based on production rules created with the assistance of an expert in the field of blood diseases and antibacterial medications.

2.2.2 NEOMYCIN. The more intelligent offspring of MYCIN, NEOMYCIN, also provides diagnosis and treatment recommendations of infectious blood diseases as well as meningitis. The primary advantage of NEOMYCIN is in its separation of the inference strategy and the expert knowledge regarding blood diseases and medications. This strategy of separating the reasoning mechanism and the domain knowledge was a major contributor

to the development of other expert systems, and led to the development of expert system shells. MYCIN was subsequently stripped of its knowledge of blood diseases leaving only the inference mechanisms and knowledge representation schemes. The resulting system was renamed EMYCIN for Essential MYCIN. Systems such as EMYCIN are appropriately called expert system *shells* because they lack knowledge regarding any specific problem. They do, however, possess all the appropriate knowledge representation and reasoning mechanisms. Expert system shells can be populated with knowledge of a particular problem domain without the user being overly concerned with the actual reasoning mechanism behind the shell.

2.2.3 PUFF. The PUFF expert system was one of the first systems developed using EMYCIN. The development was accomplished by populating the EMYCIN shell with knowledge concerning pulmonary function disease. Developed at Stanford University, PUFF diagnoses obstructive airway diseases (OAD) in patients, by interpreting data from respiratory tests. PUFF is similar to MYCIN in that it uses the same inference mechanism namely, a rule-based, exhaustive backward chaining strategy with uncertainty. Its primary contribution to the state of expert systems was in its ability to effectively increase the performance of a lab technician rather than replace him, despite the relatively small size of the resulting system. Whereas MYCIN only reached the research prototype stage, PUFF is currently diagnosing the presence and severity of OAD's at the Pacific Medical Center in San Francisco (2).

2.2.4 HEADMED. Developed in the mid 1970's HEADMED was primarily an expert system designed to assist physicians in the diagnosis of a wide range of psychiatric

disorders (18). Using knowledge about neuroses, behavior disorders, substance abuse, schizophrenia and other major psychological disorders as well as the Minnesota Multiphasic Personality Inventory (MMPI), HEADMED was also able to recommend psychiatric treatments, both drug related and therapeutic.

In addition to providing assistance in the diagnoses and treatment of medical diseases expert systems have been widely used in the diagnoses of system malfunctions, both electrical and mechanical.

2.2.5 DELTA (Diesel-Electric Locomotive Trouble-shooting Aid). DELTA was designed to assist in the diagnosis of diesel-electric locomotive failures. The AI techniques employed in this effort are not significantly different from those used in the medical diagnostic expert systems discussed previously. DELTA is a rule-based system employing a combination of backward and forward chaining inference strategies. One interesting point about DELTA is that it is highly integrated with other application packages. This integrated environment includes the ability to:

1. identify specific locomotive components
2. locate specific locomotive components
3. classify replacement parts
4. display repair procedures via video disks
5. print hardcopies of repair procedures

This integrated environment provides a total system description of General Electrics' diesel-electric locomotives and has been used not only by experienced repair technicians, but also as a training aid by those less experienced.

2.3 Model-Based Expert Systems

As expressed by Newquist, expert system applications are as prevalent as "pretzel vendors on New York street corners" (20). Our objective in this section is to delineate between the so called pretzel vendors or rule-based diagnostic expert systems and the emergence of the second generation diagnostics, namely model-based systems. Whereas traditional expert systems perform diagnosis using a set of facts, along with production rules representing empirical associations among those facts, model-based systems attempt to diagnose problems by exploiting a system's structure and function. By reasoning about differences between the way a system should function and the way it is functioning, model-based systems alleviate the need to explicitly identify all possible faults. It is believed that such an approach may be more cost effective since the information required to develop a device model is often the actual design documentation used in manufacturing the device. No experience in troubleshooting specific device failures is required. Model-based techniques may also provide better fault coverage than rule-based systems which only encode information about known failures. Although some success has been derived from systems and methodologies based on rules, these systems have simultaneously demonstrated their limitations. With the increased popularity of rule-based diagnostic expert systems emerged the understanding that not all problem domains were treatable either implicitly or efficiently by such systems. Formidable amounts of research and development at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology (MIT) into alternative diagnostic methodologies has spurred interest in what has been referred to as model-based diagnostics, or deep reasoning.

Researchers at MIT have described a system that diagnoses problems by relating observed system behaviors to predictions about the system's correct behavior (5, 15). Rather than explicitly encoding the relationships among parameters of interest in rules, model-based systems diagnose failures by using structural and functional design data in the form of a model. Any discrepancy between the behavior predicted by the model and the observed behavior is indicative of a fault. The system uses the discrepancy to determine what components may have contributed to the fault.

Numerous systems have attempted, and in some instances succeeded, in demonstrating the applicability of model-based reasoning to diagnosis. Some of the more popular applications are as follows:

2.3.1 FELIX (CAT scan diagnostic advisor). FELIX is a diagnostic expert system which reasons about malfunctions in CAT scan equipment, not by using empirical associations, but rather a structural understanding of the CAT scanner itself. FELIX was the first system implemented in the model-based representational software tool called the Intelligent Diagnostic Expert Assistant (IDEA). IDEA is the recently developed product of AI Squared Inc., a firm implementing the work of Dr. Randall Davis, who has conducted significant research into the diagnosis of faults based on a system model rather than past experiences (14). FELIX is an interactive system requiring a technician to input fault symptoms and/or error codes, and which subsequently diagnoses CAT scan problems. FELIX may request further observations in order to reduce the discrepancies it must consider.

CAT scanners cost from $500,000 to $2 million, thereby making downtimes quite

expensive. Prior to the development of FELIX the average repair time for a scanner was measured in hours. Experience thus far with FELIX, indicates a repair time of between 20 and 30 minutes (14). FELIX has also made it possible for less experienced technicians to perform at or near the level of senior technicians.

2.3.2 Automated Fault Handling of a Satellite Electrical Power System. This was a development project at Ford Aerospace & Communications Corporation of Sunnyvale, California using yet another model-based software development package called PARAGON. Implementation of a model-based system was warranted because a rule-based system would only provide assistance in those situations explicitly formulated in its knowledge base. With a model-based approach Ford Aerospace felt a wider coverage of satellite malfunctions could be achieved thereby prolonging satellite missions and extending their service life.

This particular system interprets satellite telemetry data and warns of impending satellite shutdown due to power deviations. The system determines probable causes of the power fluctuation and recommends a sequence of command instructions to correct the situation and avoid the shutdown. Timely interpretation of satellite telemetry data is critical in identifying deteriorating satellite operation. With the implementation of the model-based expert system all the knowledge that could be put to bear on a problem can now be consistently applied to all problems and subsequently save satellite missions (3).

2.3.3 Fault Isolation System (FIS). FIS is an expert system, developed as a research project at the US Naval Research Laboratory, that assists repair technicians in trouble-shooting problems with electronic equipment. Using FIS and documentation describing the structure and function of a particular piece of electronics, a knowledge engineer

can construct a model of that equipment. One aspect in particular which makes FIS an interesting subject of discussion is its ability to reason qualitatively from a functional representation of the equipment rather than from a numerical simulation. FIS exemplifies this by using qualitative values such as bad and ok for binary signals, rather than relying on electrical characteristics. FIS also incorporates *a priori* component failure probabilities (23).

As an ongoing research project FIS is being extended to investigate ways to add further capabilities such as:

- the use of quantitative relationships in conjunction with qualitative relationships
- the automatic deduction of qualitative relationships based on quantitative relations
- the use of an extended explanation facility

2.3.4 Blended Diagnostic System (BDS). This system was developed by Skinner (26) and was an investigation into the blending of a rule-based diagnostic system and a model-based system. The problem domain was the Dual Miniature Inertial Navigation System (DMINS) used on fast attack submarines, oceanographic survey ships, and aircraft carriers. Skinner implemented BDS in S1, a rule-based expert system shell, and by using an object oriented approach was able to model the connectivity of the components composing the DMINS. The resulting system was demonstrated, tested and subsequently delivered to the Aerospace Guidance and Metrology Center (AGMC) located at Newark AFB, OH. which has responsibility for repairing the navigation units (26).

As mentioned previously, rule-based systems are ill-equipped to capture the structural and functional characteristics of complex devices. Skinner's system was adequate in

capturing the expert's knowledge but the model-based portion was primitive compared to techniques currently available. Chapter 5 includes a discussion of BDS and its model-based strategy.

2.4 Summary

Rule-based expert systems are extremely popular for a variety of problem- solving applications. With this increased popularity has evolved an awareness of the limitations of this technology. Work in the AI community on model-based diagnosis has grown out of a desire to move away from these weaknesses and toward a system which more closely approximates the troubleshooting process. The intent of this research is to apply a model-based software development tool to the diagnosis of faults in the Dual Miniature Inertial Navigation System.

III. Model Based Diagnosis

3.1 Introduction

Often, to determine why a system or device is malfunctioning, it is beneficial to understand how the device behaves when operating properly. If a technician understands the normal operating characteristics of a system, he can use that information to determine the cause of system failures. Most technicians routinely construct mental models of devices they are troubleshooting. This phenomenon has led researchers to hypothesize that certain diagnoses could be accomplished using the actual design of a device, in the form of a model.

Generically a model is simply an abstraction of some item of interest. Models range from simple structural models, representing only the connectivity among device components, to computationally intensive numerical models requiring extensive knowledge of the mathematics behind the device. This raises the issue concerning the intended precision and efficiency of the modeling. Consider simulating any device, of some relative complexity, whose model not only captures detailed information of physical laws, such as Ohm's law, but also behavioral aspects of the complex mechanical components in the device. Such a model would be impractical, but emphasizes one of the key perplexities inherent in modeling. At what level of abstraction should the system be modeled in order to obtain a precise diagnosis while not sacrificing diagnostic accuracy or unnecessarily increasing computational time?

Researchers at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology (MIT) have described a system that diagnoses problems by relating observed device behaviors with expected behaviors produced by a model of the device. Model-based

troubleshooting is driven by the interaction of system observations and model predictions. A model is used to predict device outputs which are subsequently compared to observed outputs. Should this result in a discrepancy, a fault is indicated and the discrepancy is traced back through the device structure to create a list of possible malfunctioning components, often referred to as a *suspect list.* Additional observations are requested to prune the list and identify the malfunctioning component.

AI Squared, Inc. of North Chelmsford, MA has developed a modeling tool based on this concept pioneered by Professor Randall Davis of MIT. The following chapter describes the techniques used in IDEA and in the prototype developed in this research.

3.2 Constraint Networks

The following sections build upon a basic understanding of constraint networks and their application in model-based diagnosis. To create a common framework for these subsequent discussions a brief description of constraint networks and their application is provided.

While there are many ways to represent the structure and function of a given device or system, the technique exploited here is that of a constraint network. "Constraint networks provide a means of combining primitive constraints in order to express more complex relations"(1). In developing a constraint network we define device components as objects and behaviors as a set of logical propositions. The logical propositions represent the relationships among component terminals. Object terminals have the ability to "hold" values that may be propagated to other terminals. The propagation is defined by "primitive constraints which state that certain relations hold between" object terminals (1). By

connecting objects we create a "constraint network". The resulting device representation allows a qualitative simulation of the device. To understand the fundamental principles of constraint networks it is best to explore a simple example as illustrated in Figure 3.

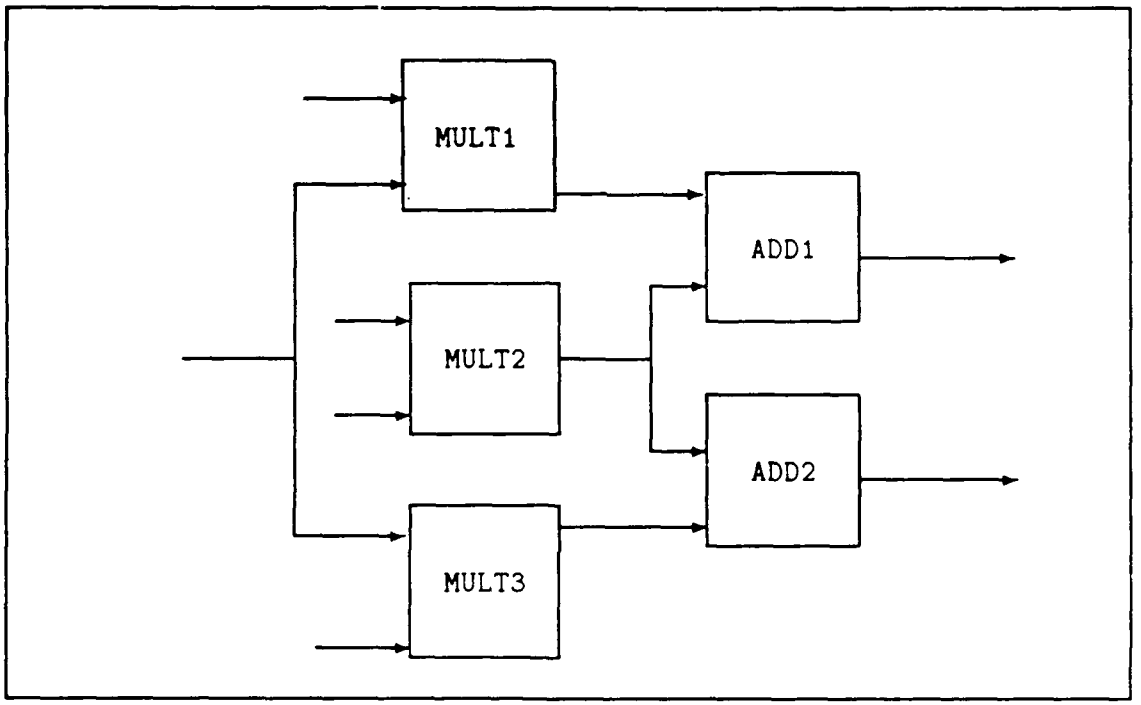

Figure 3. The Canonical Model

This model is often referred to as canonical since it is reduced to the simplest and clearest schema possible. The canonical model is sufficient to illustrate the fundamental concepts used in model-based diagnosis and in particular the techniques used in the Intelligent Diagnostic Expert Assistant (IDEA) software.

The canonical model is an abstract representation of a device, perhaps a simple calculator, in the form of a constraint network. The device is composed of three multipliers and two adders connected as shown. In addition each component has an associated de-

scription of its behavior. These behavioral descriptions are typically sets of expressions that capture the causal relationships among the input/output terminals of the components and encapsulate two different types of behavior. First, we have the concept of *simulation* where values are propagated through the network to determine output values based on input values. Second, we can define behaviors which deduce input values based on other device inputs and outputs. These behavior representations are called *inferences*. For instance, the function of the multiplier is to multiply two input values and produce an output value, and can be represented by the simulation clause illustrated in figure 4. We can also

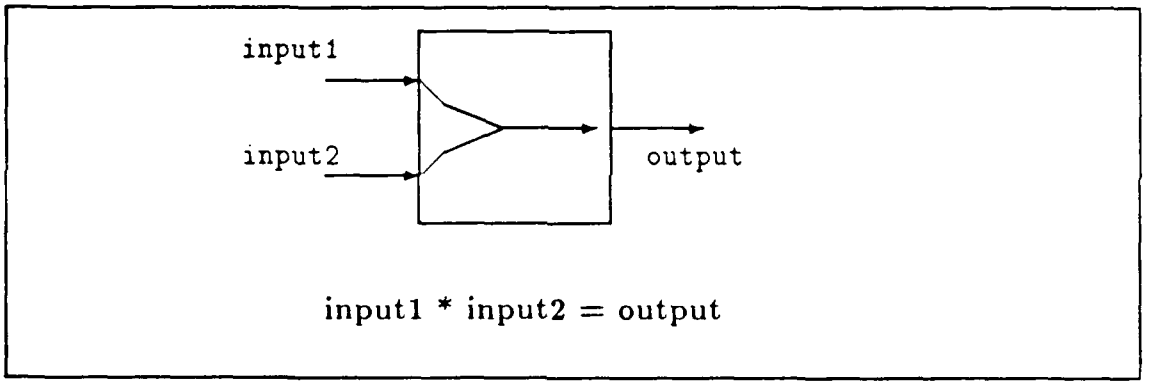

Figure 4. Basic Simulation scheme

develop appropriate inference behaviors for the multiplier as illustrated in Figure 5. These representations fully describe the correct behavior of a simple multiplier. We construct a device model by instantiating component objects, along with their associated behavior representations, and connecting the object terminals. The resulting structure is called a constraint network, and is used to produce a qualitative simulation of the device's behavior. An important feature of this representation is that the behaviors need only be defined once,

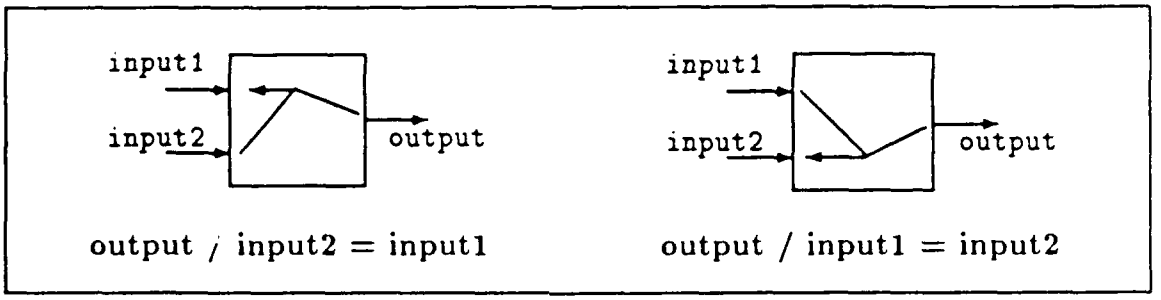

Figure 5. Basic Inference scheme

though the device may have several multipliers. This is because our objective is to model *correct* behavior. Unique multipliers may exhibit different fault characteristics; however, in our modeling this is irrelevant. Any behavior deviating from the correct component behavior represents a fault.

In summary, a constraint network allows us to build a device representation that encapsulates not only the connectivity of device components but also a behavioral description of those components. By creating connections among the components, the resulting network becomes not only examinable but executable. This intriguing combination allows the representation to be queried, to obtain information about the device structure, and to be "run" in simulating the device. It also forms the underlying principles used in the IDEA software.

The algorithm used in this research to perform model-based diagnosis consists of three fundamental tasks (5). Those tasks, as described in the following sections, are:

Hypothesis Generation **Hypothesis Testing** **Hypothesis Discrimination**

3.3 Hypothesis Generation

Hypothesis generation requires an initial device simulation which is accomplished by propagating device inputs through the model. For instance, suppose we provide 2 and 3 as the inputs to the multipliers of our canonical model. The model takes those values and simulates the behavior of the multipliers. The resulting values are propagated to the adders which use their defined behaviors to produce the model outputs. The ensuing propagation results in the network illustrated in Figure 6. After this initial simulation,

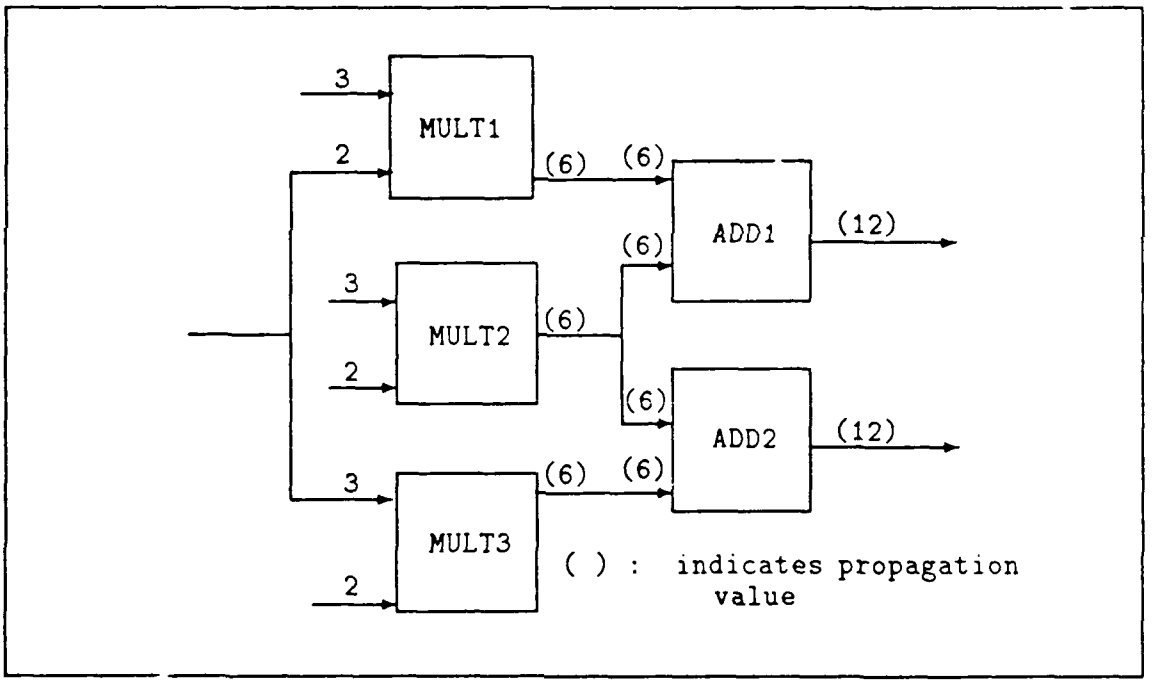

Figure 6. Canonical Model after Initial Simulation

suppose we inform the system that we had *observed* a 10 at the output of ADD1. This would result in a conflict with the simulated value of 12 already at that terminal, and constitutes a *discrepancy*. In hypothesis generation, once a discrepancy between a model

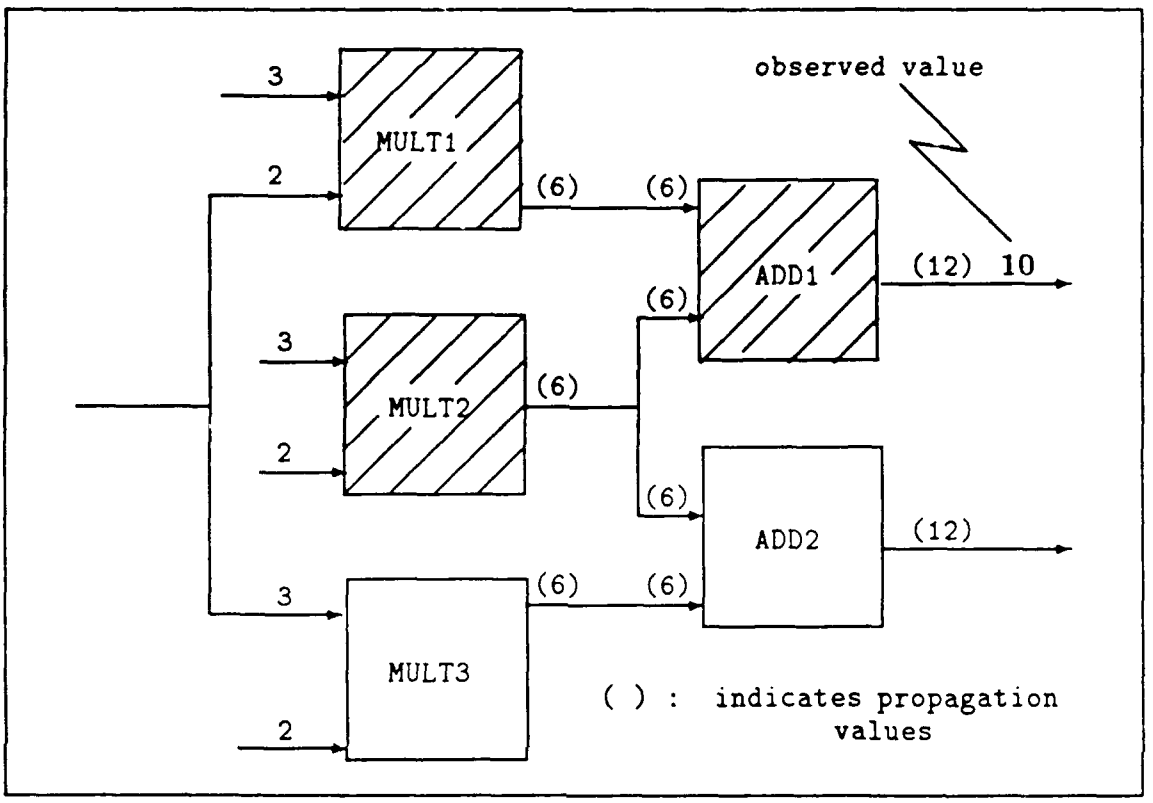

Figure 7. After Initial Candidate Generation

prediction and an observation has been detected, the task becomes one of determining which device components may potentially be causing it. Some fundamental concepts in hypothesis generation are summarized below.

1. To be a suspect, a component must be connected to a discrepancy
2. Not every component input influences each output

The reasoning mechanism traces the discrepancy back through the device structure and concludes that either MULT1, MULT2, or ADD1 could be responsible for the fault, as illustrated in Figure 7.

In compiling our suspect list, the system not only transverses the device structure, but also examines component behaviors. These behaviors can assist in candidate generation by determining what inputs influence the output being traced. Though not exemplified in our canonical model suppose we had a multiplier with three inputs and two outputs as illustrated in Figure 8. In this instance, the simulations are defined such that OUTPUT1

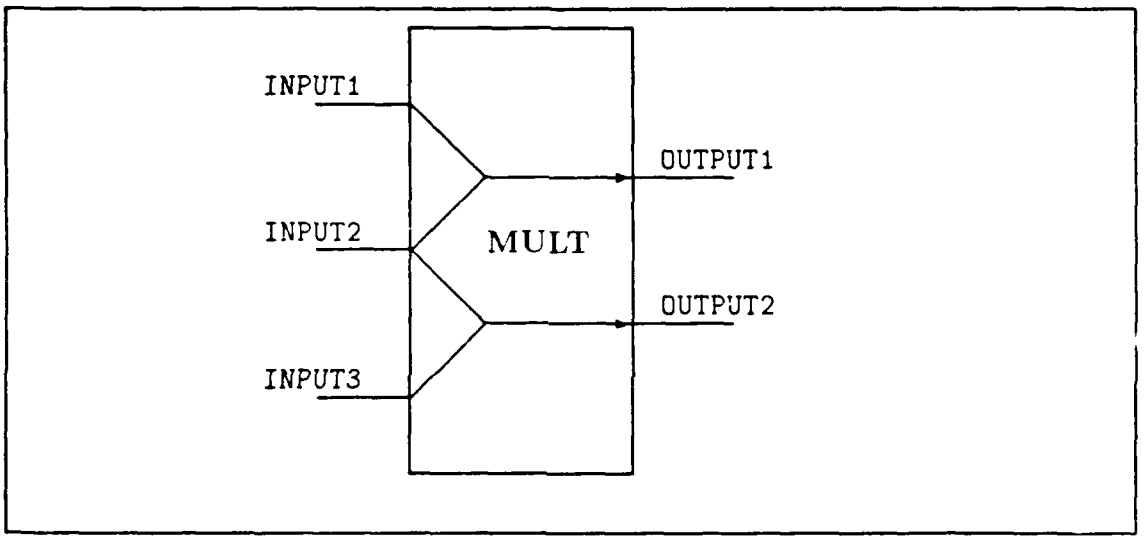

Figure 8. Using Behavior in Suspect Collection

is the product of INPUT1 and INPUT2, and OUTPUT2 is the product of INPUT2 and INPUT3. If a discrepancy is detected at either multiplier output, a pure structural analysis would trace **all** component inputs upstream in collecting suspect components. However, in the event a discrepancy is detected at OUTPUT1 it would be futile to trace INPUT3 upstream since no component connected to INPUT3 could possibly have contributed to the discrepancy. To avoid this inefficiency the system would query the behavior descriptions of OUTPUT2 and trace only the inputs designated in the **From Terminals** slot. The

simulation clause for OUTPUT2 is as follows:

```
SIMULATE
    Terminal        OUTPUT2
    From Terminals  INPUT2, INPUT3
    Perform         set @OUTPUT2 to @INPUT2 * @INPUT3
```

Using this behavior description the reasoning mechanism will only trace inputs INPUT2 and INPUT3 in collecting suspects.

In summary, hypothesis generation entails not only querying the model for components *connected* to the output producing the discrepancy but only those which might have *functionally* contributed to the discrepancy. It is this combination of structure and function which eliminates the need to follow irrelevant input connections upstream while searching for suspect components. Hypothesis generation results in a list of components, called the *suspect list*. After collecting the initial suspect components the system proceeds to prune the list as described in the following section.

3.4 Hypothesis Testing

In hypothesis testing, the fundamental task is to test the components in our suspect list to determine which of them could account for *all* device observations. The technique used in this research is called *Constraint Suspension*. It tests whether or not a given component's behavior is consistent with *all* observations of the device behavior. Basically the process asks the question:

is it consistent to believe that only the suspect is malfunctioning?

That is, given the device inputs and the observed outputs, is it consistent to believe that

all the other components in the device are functioning properly, and that only the suspect is broken? This is accomplished by ignoring or "turning off" the behavioral constraints on the suspect, firing all remaining constraints and searching for network inconsistencies. Whether or not a suspect can be eliminated is based on the consistency of the resulting network. The network is considered consistent only if no conflict arises on any component terminal. If the network remains consistent after suspending the constraints for a particular component, then that component remains suspect. In other words, assuming single failures, that component could have failed in a manner consistent with the current device state, which includes the discrepancy, and is therefore a candidate. On the other hand, if the propagation results in an inconsistent network, where a constraint fires which attempts to place a value at a terminal on which there is already a *different* value, then that component can be eliminated. This is because the propagation, after running to quiescence, finds no set of values for the suspect terminals which is consistent with the observed values and the remaining constraints.

Using the canonical model let us perform *constraint suspension* on each of the three components making up our suspect list. If we "turn-off" the constraints associated with ADD1 and fire all remaining constraints, the ensuing propagation would result in a network as illustrated in Figure 9. Notice that no value is propagated through ADD1 since the constraints defining its behavior have been suspended. As can be seen from this figure, if all other components are working properly no conflicts arise anywhere in the network and ADD1 becomes a candidate. This is relatively intuitive since we can see from the device structure that if ADD1 were broken it could cause its output to be 10, or any other value for that matter. Suspects surviving constraint suspension are subsequently referred to as

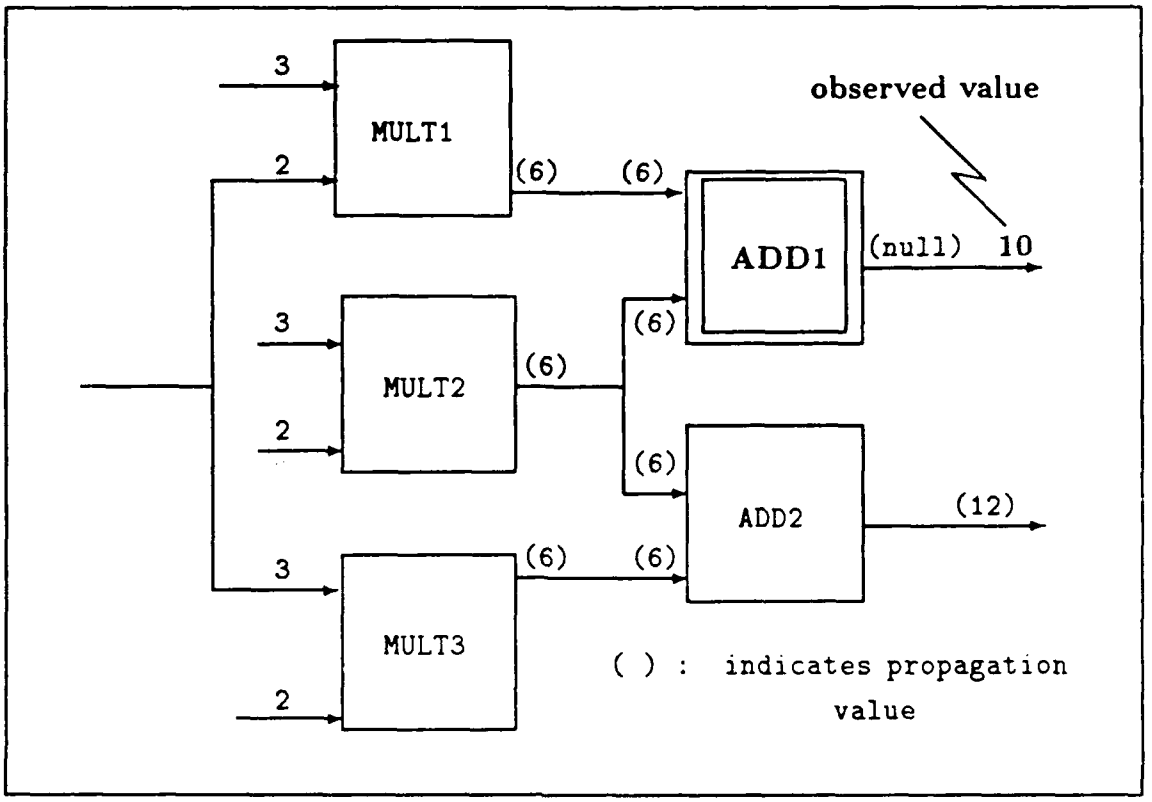

Figure 9. Constraint Suspension on ADD1

candidates. Continuing with this scenario let us "turn-off" the constraints associated with MULT1 and again allow the network to fire all remaining constraints. The propagation results in the network illustrated in Figure 10. The value 4 is a result of an inference by ADD1, but does not result in any conflicts since the constraints of MULT1 are suspended. As was the case with ADD1, no conflicts arise anywhere in this network either. Therefore MULT1 also becomes a candidate.

Finally let us suspend the constraints on MULT2 and, as before, allow the network to fire all remaining constraints. The network resulting from this last propagation is presented in Figure 11. The inference constraints for each of the adders were fired to obtain the input

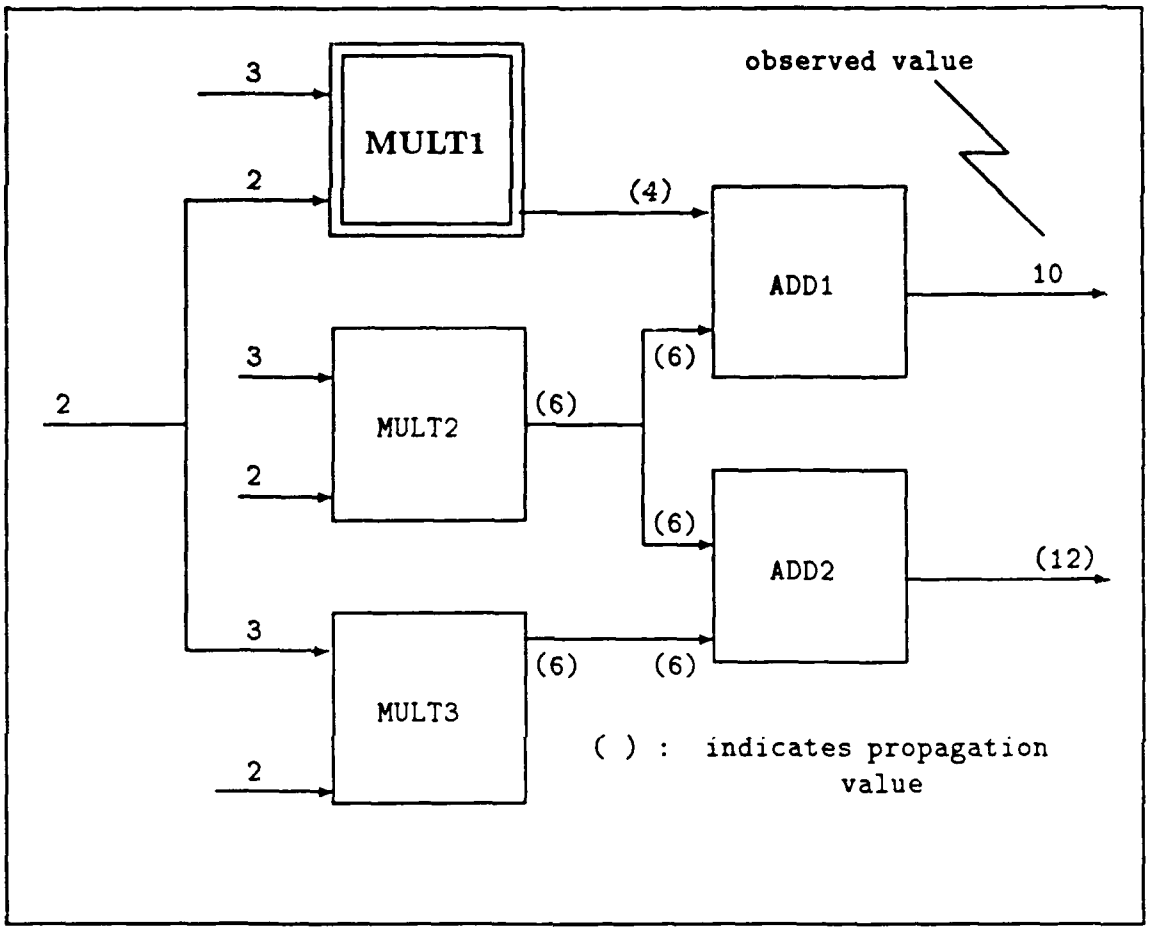

Figure 10. Constraint Suspension on MULT1

value being supplied by MULT2. This resulted in the propagation's attempting to place conflicting values on the same terminal. Since the network is now *inconsistent* MULT2 is no longer considered a candidate and is therefore removed from our suspect list. This is relatively intuitive since if ADD2 is operating properly then it is consistent to believe that MULT2 is also operating properly. This process of constraint suspension continues until all components in the suspect list have been suspended. In this example the resulting list contains only ADD1 and MULT1 and is now referred to as the candidate list.

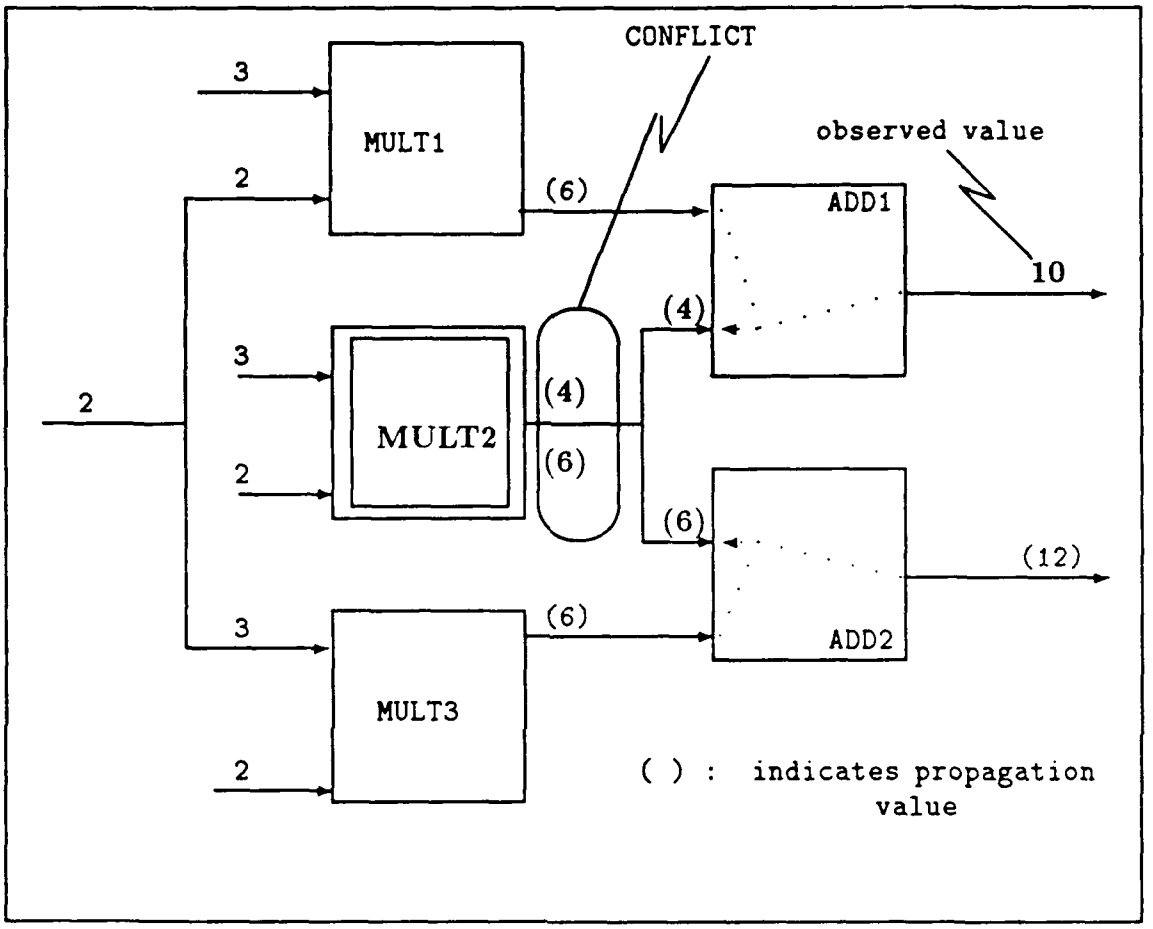

Figure 11. Constraint Suspension on MULT2

One interesting aspect of this process is that the reasoning strategy does not require us to predetermine component failure modes. Our only objective is to determine whether or not a component is doing what it is supposed to do, any other behavior is considered a malfunction.

3.5 Hypothesis Discrimination

The first two stages of this algorithm provide us with an initial list of candidate components, and some pruning of this list. The problem facing us now is how to discriminate between the remaining components. One approach is to gather additional observations of the device's behavior. This can be accomplished by either probing the device or testing device components. Numerous criteria exist in selecting probe points; these vary from random selection, to tracing the discrepancy upstream through the device structure, to using component failure probabilities. The challenge is in selecting an optimal probe while not sacrificing efficiency in computationally intensive probe selection.

The IDEA software provides a capability to create a probe/test selection strategy best suited for the specific application. The developer can create a *strategy clause* which allows the ranking of applicable examination points according to the following sort keys: cost, average reliability, number of modules exonerated. Upon reaching a point in the reasoning process where a probe or test selection is warranted the system will rank all applicable choices according to the sort key contained in the strategy clause.

Test and probe points, collectively referred to as examination points, have associated cost attributes. Cost refers to the associated difficulty of performing a given probe or test. For instance, in our prototype, certain probe points require special test controller cables to be attached to the IMU, while others do not. The cost attribute can be assigned a value corresponding to the difficulty in connecting these cables. If the probe/test selection strategy sort key is *cost*, the resulting list will be ordered according to cost, lowest to highest.

Components also have an average reliability attribute, which allows a measure of the respective components' reliabilities. If the probe/test selection strategy sort-key is *average reliability*, the resulting list will be ordered according to the average reliability of the candidate components. If the candidate list contains a device more likely to fail than some other component, that component would have a higher probability of being probed and/or tested.

The modules-exonerated key refers to the number of candidate components that would be eliminated after conducting a given probe or test. The cost and average reliability keys use the corresponding attributes to sort the examination points. However, if the strategy clause contained the modules-exonerated key, the applicable examination points would be sorted according to the number of candidates remaining upon conducting the probe or test. The selected examination point would ideally result in the smallest candidate list. This allows the diagnosis to more readily reduce the number of candidate components. Conducting a probe or test provides additional information about how the device is actually behaving. The model uses this additional information in generating a new suspect list and repeats the three tasks just discussed. The algorithm concludes when further observations of the device are unachievable or the candidate list contains only a single component. The algorithm encompassing the strategy just presented and used in the IDEA software is depicted in Figure 12.

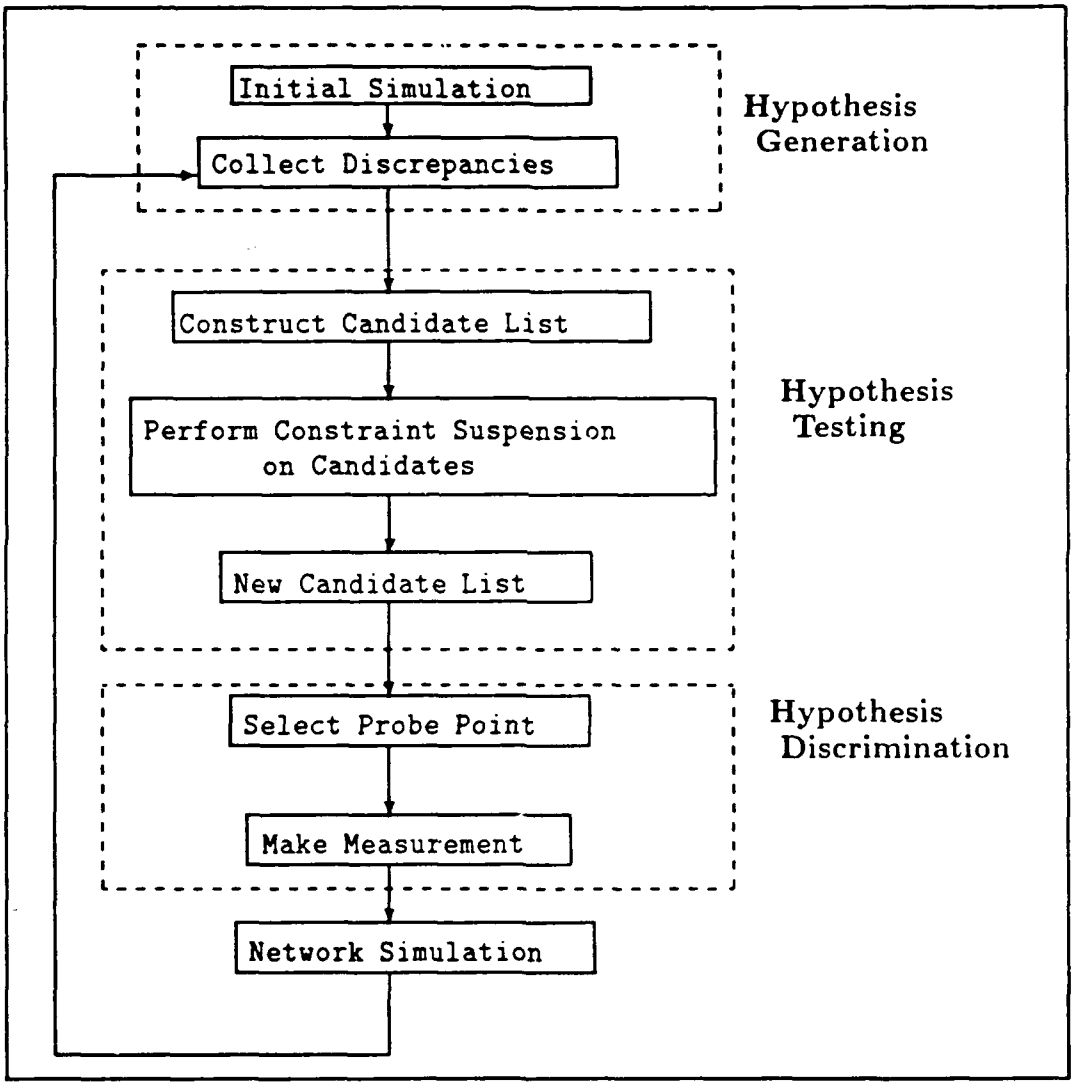

Figure 12. IDEA Reasoning Algorithm

IV. DMINS Testing Philosophy

4.1 Introduction

The Dual Miniature Inertial Navigation System (DMINS), first introduced in 1974, is an inertial navigation system used on fast attack submarines, oceanographic survey ships, and aircraft carriers. The system provides continuously updated attitude (roll and pitch), heading, velocity (north and east), and position (latitude and longitude) data to other ship subsystems (11).

The DMINS comprises the following major assemblies:

Navigation Control Console

Left Blower Assembly

Left Inertial Measuring Unit (IMU)

Left Electrical Equipment Mounting Base

Right Blower Assembly

Right Inertial Measuring Unit

Right Electrical Equipment Mounting Base

At the organizational maintenance level, the inertial measuring unit (IMU) is replaced as an entire unit. The faulty unit is shipped to the Aerospace Guidance and Metrology Center(AGMC) located at Newark AFB, OH. AGMC is the depot level maintenance shop for all inertial navigation equipment.

The IMU contains the sensing instruments (velocity meters and gyroscopes) and support circuitry necessary to sense a ship's acceleration, integrate that data to compute

the ship's velocity and integrate a second time to compute the ship's displacement. Current testing of the DMINS IMU is conducted by automatic test equipment (ATE), driven by an IBM 1800 test controller and test program. The test program cycles the IMU through various test modes required to verify its operation. There are three separate test modes designated Mode A, Mode B and Mode C. Mode A and Mode B tests are described in ensuing sections since they are instrumental in testing the IMU itself. Mode C is a test of the automatic test equipment itself.

4.2 Initial Testing: Mode B

Initial testing of the IMU is referred to as Mode B testing. The purpose of Mode B is to provide an automatic *fault isolation test* on the IMU. Mode B consists of 23 main tests each with varying number of sub-tests, and is run with the IMU mounted on a stationary pier with access provided by an IMU interface control unit (IMUIC). The primary objective is to test for gross IMU failures and to ensure that operational signals are within coarse limits. This phase of testing, however, will not confirm whether or not the IMU will properly navigate.

Mode B provides a computer printout indicating the status of each mode B test as it is conducted. A GO or NOGO flag is supplied indicating whether or not the test passed or failed. The upper and lower signal specifications are also printed along with the value actually measured by the test program. In the event of a test failure the technician can either verify the failure and attempt to further isolate it, by referring to the IMU circuit schematics, or ignore it and proceed with the testing. Upon successful completion of Mode B the IMU is sent to Mode A testing.

4.3 Mode A Testing

The purpose of Mode A is to provide an automatic *functional performance test* which will cycle the IMU automatically through a calibration criteria test and subsequently into a long term navigation performance run. Figure 13 illustrates the progression of the specific Mode B tests listed below.

MODE A Initialization
Velocity Meter Calibration
Gyro Calibration
Self Align
Nav Align
Nav Performance

Whereas in Mode B the technician uses a GO/NOGO printout in his diagnosis, in Mode A he has more flexibility. He can either plot and interpret IMU performance parameters to detect system degradation or use one of the 62 error messages generated by the test equipment. Mode B testing often detects hard failures that exhibit definite signal characteristics. In Mode A, even though the circuitry may be in spec as implied by the successful completion of Mode B, the IMU may have trouble properly functioning over long periods of time. To determine whether or not an IMU will operate within its performance specifications requries a minimum of 70 hours of Mode A testing. Troubleshooting errors in Mode A testing requires a firm understanding of the performance parameters and their expected behaviors over time. Capt Skinner, in his initial research, relied solely upon the ATE error messages to initiate the diagnosis. Current research efforts by Capt Dan Florian at the Air Force Institute of Technology are being conducted to investigate the

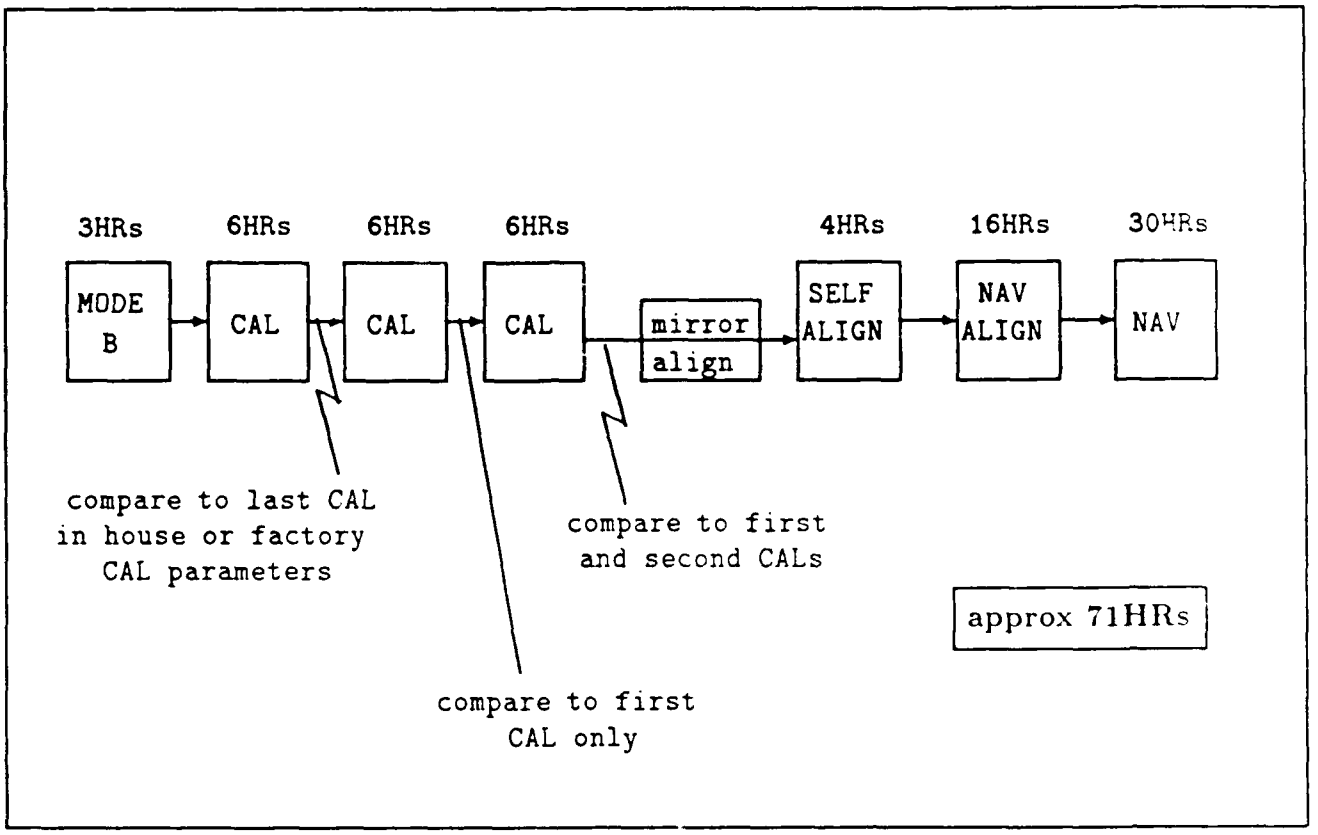

Figure 13. DMINS Mode A Test Scenario

use of performance parameters in providing further expert information in the diagnosis of IMU failures.

4.4 Integration of Prototype

Model-based reasoning is most applicable to diagnostic scenarios in which technicians construct mental models of the devices or systems they are troubleshooting. Using this premise we decided to focus our prototype development efforts toward assisting AGMC technicians in the diagnosis of Mode B test failures. This decision was based on the fact

that technicians almost always use the IMU circuit schematics when troubleshooting Mode B failures. Our prototype provides a consistent capability in utilizing the IMU circuit structure in that diagnosis.

V. *Prototype Development*

This chapter documents the development of the DMINS Inertial Measuring Unit (IMU) models and the resulting diagnostic prototype. The primary motivation for this work was that current model-based reasoning techniques are significantly better than those used by Skinner (26). We felt that by employing more advanced techniques, we could build a diagnostic system for the DMINS that provided a more thorough and consistent diagnosis than is provided by BDS. The introduction explains some of the major factors inspiring this research.

5.1 Introduction

Skinner proposed a blending of rule-based and model-based techniques in an attempt to draw upon the strengths of both. Rule-based systems tend to be extremely well suited in capturing general rules of thumb or heuristics, based on accumulated experience. Model-based systems on the other hand require no trouble-shooting experience, but instead diagnose failures by reasoning from an abstract model of the device. Though a novel approach, the model-based portion of Skinner's prototype was primitive compared to techniques currently available. The model, due to limitations of rule-based systems, incorporates only the connectivity of the IMU shop replaceable units (SRUs). No information was encoded concerning the functional relationships among individual SRUs, nor among terminals of specific SRUs. Consequently, running the model always initiates a point-by-point search for the fault (26). Skinners prototype also relies heavily on generic

"prompts" in requesting user inputs. For instance, the following prompt is displayed to request the status of the 4.8KHz output signal from the Frequency Standard Function.

Is the output of the Frequency Standard Function good?

This obscure prompt leaves significant room for misinterpretation, resulting in inconsistent diagnoses. What a novice technician perceives as a good output may be considerably different from what an expert technician considers good. All prompts provided by BDS are of the same format and simply ask whether or not the output of a given component or function is good. These prompts should provide more detailed information to allow for consistent and meaningful diagnoses.

In reviewing BDS we noticed an elaborate mechanism was devised to handle multiple component outputs, which could become extremely cumbersome. As previously mentioned, the capability of rule-based systems to encode device structure and function is limited (5). This limitation is exemplified in BDS. The fundamental algorithm used in BDS to conduct a model-based diagnosis is illustrated in Figure 14. Basically the algorithm chains through a simple graph of components tracking "BAD" inputs. If a given component has a bad output the algorithm checks all component inputs. If a bad input is detected, that input is traced further upstream in searching for the bad component. If no bad input is found BDS flags the current component as the suspect. The one characteristic inherent in BDS and indeed most model-based systems is the incorporation of a hierarchical diagnostic strategy. Before the current suspect is flagged as the actual malfunctioning component BDS first determines whether or not the suspect has any sub-components. If it does, a model is

```
START component(x) has a bad output.
   If component(y) is a bad input to component(x)
     then diagnose component(y);
   else if component(x) has subcomponents
           then build the subcomponents and
                diagnose the first subcomponent;
            else component(x) is the faulty component.
```

Figure 14. Model-Based Reasoning Algorithm used in BDS

constructed of those components and the diagnoses continues as before. The algorithm concludes only when a component is reached that has a bad output, all good inputs and no sub-components.

In developing BDS, Skinner devised an ingenious strategy to handle multiple component outputs. Basically, a unique control strategy had to be created for each individual output of a given function or device. As stated by Skinner this requirement

> ...stems from the fact that the algorithm is restricted to single-output components. Multiple-output components must be modeled by creating instances of the component, one for each separate output. (26)

This issue becomes especially significant when we realize that not all component outputs were included in the model portion of BDS. For instance, BDS represents the four IMU power distribution functions with one component, whose sole output designates the IMU power supply. Our model, however, explicitly represents each of the power outputs, only a fraction of which are listed here:

```
+28 volts                  -24 volts
+24 volts No.1             +24 volts No.2
26vac 400Hz at 120deg      26vac 400Hz at 0deg
400Hz 14vac                400Hz 115vac
```

This approach, along with component behaviors, allows fault manifestations to be traced back to specific power supplies. In fact a faulty 6 volt power source may exhibit certain device output discrepancies not seen with other power failures. This representation allows a clearer picture of how specific signals affect device performance. Using the approach proposed by Skinner would require a lengthy control structure for each output of the power distribution function. This quickly causes a significant increase in the amount of code and a proportional decrease in the understandability and maintainability of the system. It is exactly this phenomenon which Davis and Hamscher allude to in their contribution to Exploring Artificial Intelligence (9). Though it is often argued that rules can be written to capture knowledge about device structure and behavior, Davis and Hamscher feel that this is indeed the strongest argument *against* using rules. In fact they quite emphatically state that the

> ...relevant knowledge concerns structure and behavior. Given that, we ought next to ask what representations are well suited to capturing that information, and what representations offer us leverage in thinking about that knowledge. Rules, whether as empirical associations or viewed simply as if/then statements, offer us little or no help in thinking about or representing structure and behavior, or in using such descriptions to do diagnosis. Most fundamentally, they do not even lead us to think in such terms.(9)

Our investigation into alternative techniques for performing model-based diagnosis originated in an attempt to address these shortcomings. The ensuing sections describe the

steps followed in developing a symbolic constraint network representation of the DMINS IMU and its implementation as a diagnostic assistant.

5.2 *Methodology*

Chapter 1 introduced the basic methodology used in developing our prototype. A more detailed discussion of the approach is formulated below:

- **Initial Project Layout**
  - Separate independent sub-systems
  - Develop appropriate block diagrams for each sub-system identified
  - Determine valid functional characteristics for each interconnection
  - Assign a symbol for each interconnection
  - Break any feedback loops
- **Modeling Behaviors**
  - Determine the level of abstraction necessary to adequately diagnose IMU failures
  - For the SRUs targeted for implementation, develop and encode behavioral models, at the appropriate level of abstraction
  - Use the development environment in IDEA to encode the connectivity of the modules
- **Modeling Characteristics**
  - Create an examine clause for each applicable input/output
- **Create User Interface**
  - Develop and Integrate a user interface to the diagnostic prototype

5.2.1 Initial Project Layout. The identification of model boundaries was achieved by partitioning the overall DMINS system into independent sub-systems. Fortunately, the DMINS organizational level technical manual (11) provided this functional decomposition,

resulting in the identification of 19 functional groupings 11 of which directly involve the IMU. These functions include the following:

1. **115vac 400Hz Power Distribution**
2. **+28v, +42v Power Distribution Function**
3. **+/- 6v, 24v, 48v Power Distribution Function**
4. **+/- 12v Power Distribution Function**
5. Platform / Gyro Temperature Control Function
6. **Frequency Standard Function**
7. Gyro Speed Control Function
8. Gyro Torquing Function
9. **Reference 400Hz Function**
10. Platform Torquing Function
11. Velocity Meter Function

The Reference 400Hz and Frequency Standard functions were chosen for implementation. The power distribution functions were incorporated in our models as appropriate. The next step was to develop block diagrams for each of these functions, which are illustrated in Figures 15 and 16 respectively.

The next task was to determine valid functional characteristics for each interconnection in the block diagrams. This entailed determining how each component's inputs and outputs contribute to the overall functioning of the IMU. Understanding the valid characteristics of a specific interconnection allowed us to assign it a meaningful symbol. This knowledge also provides the basis for generating component behaviors. Since our objective was to develop and implement a model with only the necessary abstraction to perform the required diagnosis, we decided to begin at a relatively high level of abstraction. Therefore, we assigned symbols such as 'good, 'bad ,'high and 'low to each SRU

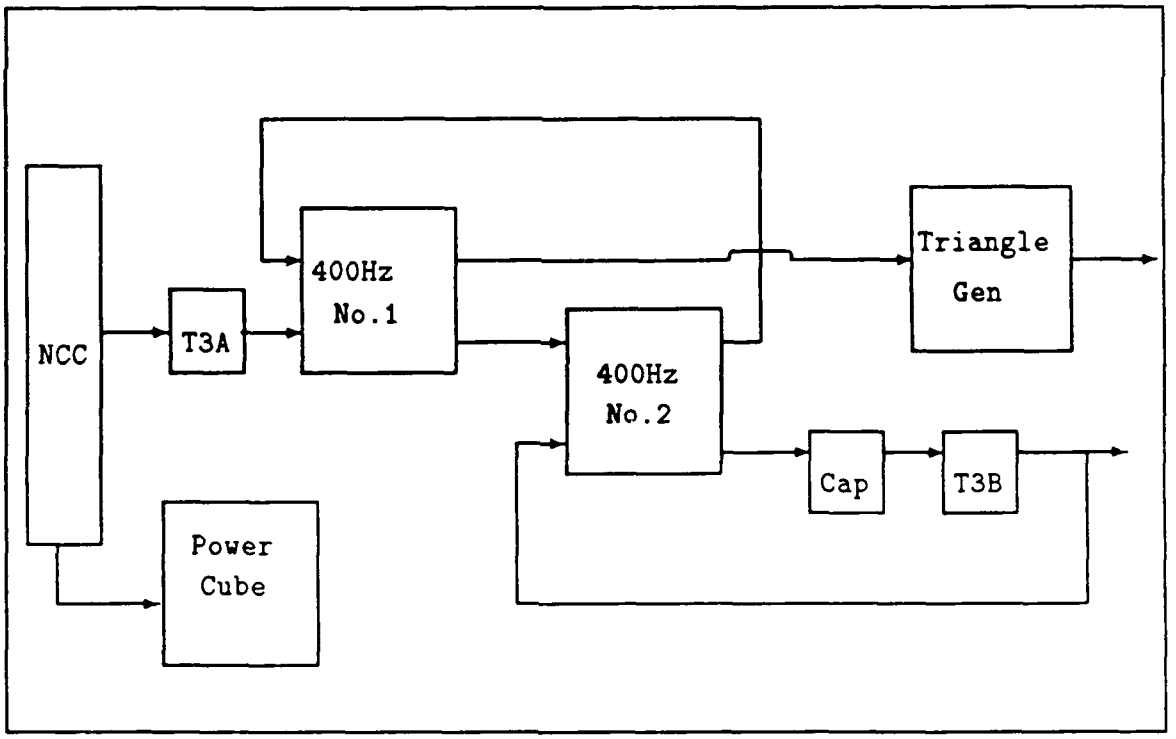

Figure 15. 400Hz Reference Function Schematic

terminal as appropriate. Figure 17 illustrates the symbols assigned to the terminals of the 400Hz power supply #1. It was our feeling that since the technicians use characterizations such as these in describing observed signals we could use the same representations in our propagation scheme. This behavioral abstraction, in addition, allowed us to suppress the detailed electrical characteristics of the IMU circuits and gain some simplification. This allowed us to concentrate on the diagnosis rather than on detailed component operation.

Some discussion on implementing the actual navigation equations in our model did take place. However, it was agreed that since system observations are primarily qualitative, such as whether or not the IMU platform is level, and not mathematical, that the model

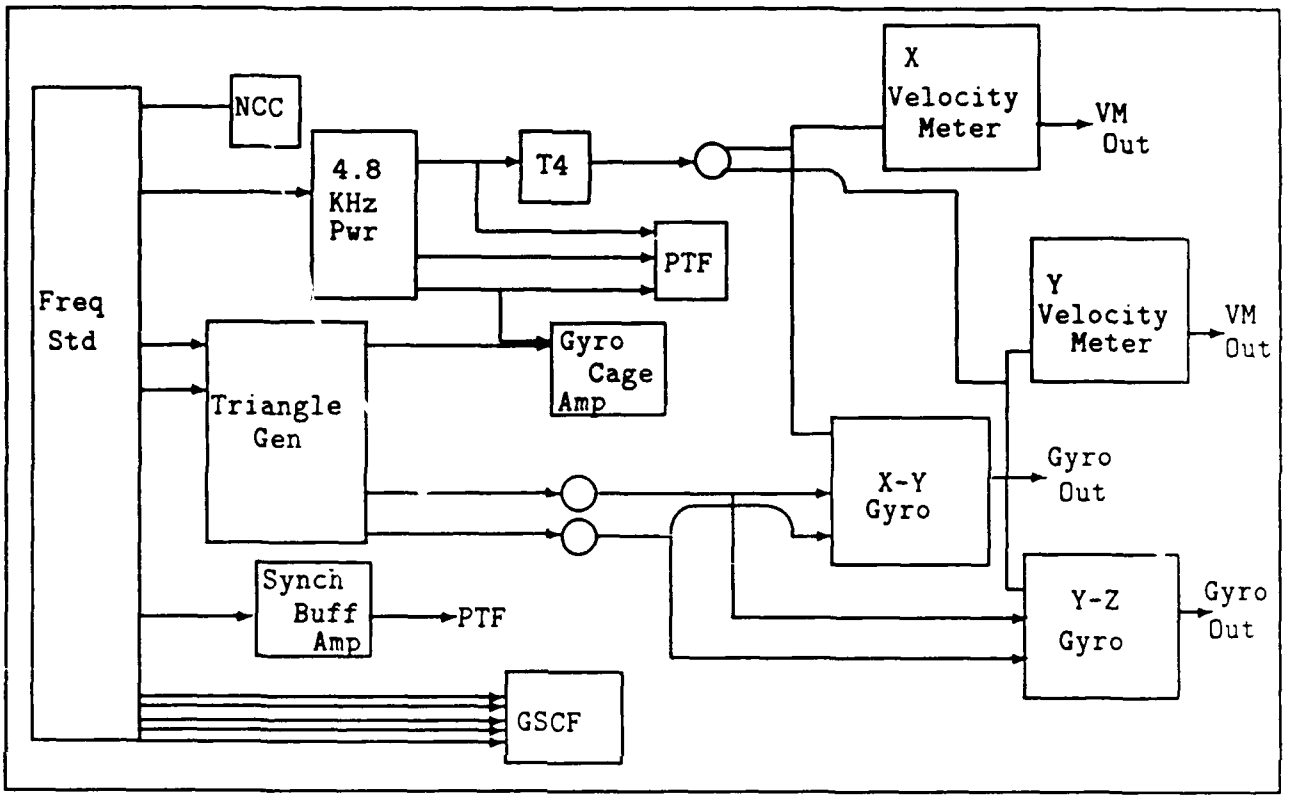

Figure 16. Frequency Standard Function

should characterize that. Whereas a mathematical model would provide a more complete description, representing the circuit knowledge as logical propositions, or constraints, and using a constraint propagation scheme provided a computationally reasonable solution. In practice, propagation of constraints seems to provide a good compromise between completeness and computational expense. Two primary considerations prevented further investigation into modeling the actual navigation equations.

1. The objective was to provide a diagnostic aid closely reflecting that of an experienced technician.

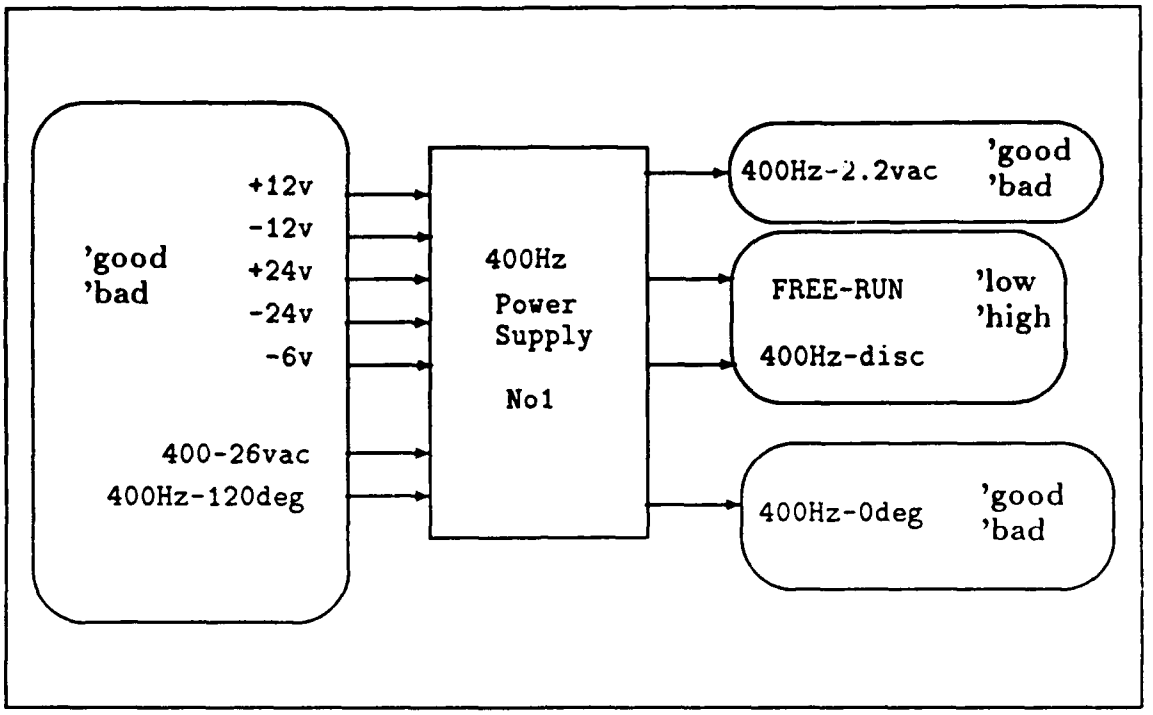

Figure 17. Terminal Symbols for the 400Hz power supply No2

2. The IDEA software chosen for implementation lacks the sophisticated mathematical capabilities to handle the differential equations necessary to model inertial navigation.

5.2.2 Modeling Behaviors. One research premise explored in this phase of the development was that a considerable amount of model development could be accomplished without the perpetual interviews characteristic of traditional expert system development. With this in mind a substantial amount of "knowledge engineering" was conducted using only the DMINS organizational and depot level repair manuals. The organizational level repair manual was invaluable in developing the functional block diagrams and the depot level manual (12) was instrumental in developing the behavioral expressions. Experts were consulted in clarifying technical details; however, the primary source of information was

the DMINS technical orders.

After each component terminal had been characterized by a set of symbolic constraints we described the behavior of each of the SRUs using causal relationships among the terminals. Considerable time was spent in determining how best to characterize these component behaviors. Ultimately these expressions were obtained by tracing internal component circuitry and determining signal dependencies. The depot level repair manual was used to determine how component inputs influence outputs and subsequently to compile abstract behavioral expressions for the components. These expressions encapsulate both the capability to simulate expected output values based on component inputs, and to infer expected input values using other component inputs and outputs. For instance, the simulation clause in Figure 18 is from the prototype and represents the causal relationship between the power inputs (-12v, +12v, +6v), the 400Hz 26vac input signal (400-26VAC) and the 400Hz discrete output signal of the 400Hz power supply #1.

Basically, the discrete output (400HZ-DISC) indicates whether or not the 400Hz used in the IMU is in synch with the ships 400Hz. If the powers are all 'GOOD, and the 26vac 400Hz input reference is 'GOOD, and the circuit is operating properly, then the discrete output should be 'HIGH, indicating that the signal is in synch. If the 400Hz 26volt input is 'BAD and the powers are 'GOOD then the discrete should be 'LOW indicating that the 400Hz signal, used by the IMU, is being supplied by an internal oscillator and is not in synch.

A further example of this can be seen in the inference clause illustrated in Figure 19, also from the prototype, which represents the relationship between the 400Hz 120-deg

```
SIMULATE

TERMINAL            400HZ-DISC
FROM TERMINALS      NEG12V-IN, POS12V-IN, 40026V-IN, POS6V-IN
PERFORM             TRUTH TABLE

     @NEG12V-IN @POS12V-IN @POS6V-IN @400-26VAC @400HZ-DISC
       'GOOD      'GOOD      'GOOD     'GOOD       'HIGH
       'GOOD      'GOOD      'GOOD     'BAD        'LOW

COMMENT: low indicates P1-C > 31vRMS or < 20vRMS. This causes
          the gimbal torquer motors to be disabled
```

Figure 18. Simulate Clause Example

input signal, the power inputs and the 400Hz 0-deg output also of the 400Hz power supply #1. This representation permits the reasoning mechanism to infer the value that should be at the 400Hz 120-deg input terminal if the values at the power inputs and the 400Hz 0degree output are known. Inferences are of particular use since they eliminate the need to probe by allowing the determination of certain signals based on inputs and outputs.

Appendix B contains the behavioral descriptions of the SRUs implemented in this research.

5.2.3 Modeling Characteristics. Upon completing the initial behavioral expressions it was necessary to identify points accessible to the technician from which further information could be collected. This led to the identification of probe point categories as follows:

- IMU Interconnect Console (IMUIC) test panel

```
INFER

  TERMINAL          400HZ-120
  FROM TERMINALS    NEG12V-IN, POS12V-IN, NEG24V-IN, POS24V-IN
                    400HZ-0DEG
  PERFORM           TRUTH TABLE

@POS12V-IN @NEG12V-IN @POS24V-IN @NEG24V-IN @400HZ-0DEG @400HZ-120
   'GOOD      'GOOD      'GOOD      'GOOD      'BAD       'BAD
   'GOOD      'GOOD      'GOOD      'GOOD      'GOOD      'GOOD
```

Figure 19. Inference Clause Example

- NCC Test Panel
- NCC jumper panel
- SRU circuit card edge connectors
- Extender Card accessible

This list is ordered according to the ease with which certain probes may be accomplished. It is easier for a technician to probe a point on the IMUIC test panel than to place an extender card in the IMU to access a signal. Therefore, the above ordering is used when selecting probe points. In addition to actual signal probes, information can also be gathered by testing individual components. For example, it is customary for a technician to simply remove and replace a suspect circuit card in an attempt to exonerate that card. Therefore, some cards were designated as testable rather than allowing their terminals to be examined. This led to other dilemmas concerning the questionable health of replacement components which will be discussed in chapter 6. In either case, the key

point is that characteristics are a means of obtaining further system observations, and can be either signal probes or actual tests.

The only remaining objective in the initial development was to develop a user interface, which is presented next.

5.2.4 Create User Interface. Because the user interface is critical to a successful application, this research integrated high-resolution color graphics to enhance the usability of the system. Particular attention was given in developing a user interface that enhanced the ease with which technicians could interact with the prototype. Fortunately, IDEA provides interface routines which allowed us to create custom user interfaces. The interface developed for the prototype includes graphical representations that accompany system requests for signal characteristics. For instance, as alluded to earlier, Skinner's system would solicit user information with naive phrases such as "Is the output of the Frequency Standard Function good?" In an effort to provide further guidance and continuity our prototype displays a color graphic depicting what the specific Frequency Standard output signal being examined is expected to look like. For example, the frequency standard generates 10 unique outputs. Therefore, our model explicitly represents each of these and provides an appropriate graphical depiction. In addition, textual information is displayed to provide exact signal specifications, probe points, warnings and other relevant data to the user. Should the user request further assistance, a help file can be called up to provide further graphic and/or textual information.

5.3 Implementation

This phase integrated the IMU component behaviors, characteristics and user interface screens into working prototype models. The IDEA software provided considerable mouse driven menu support in this effort, making the actual integration relatively straightforward. Currently the system begins with a display of problem classes. This is a list of typical problems a technician may encounter and is illustrated in Figure 20. Selection of

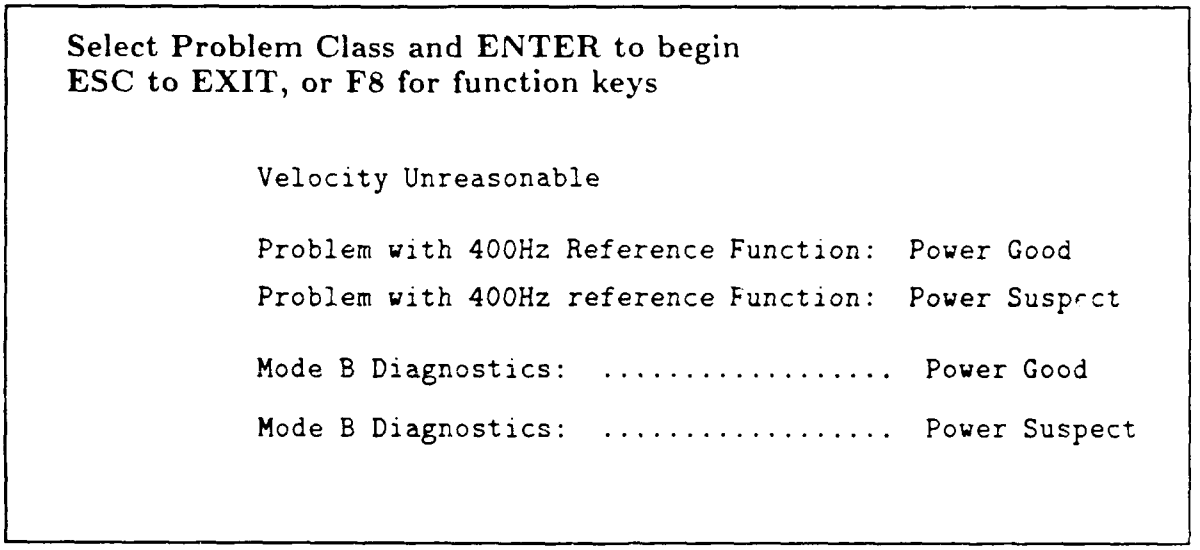

Figure 20. Initial Problem Class Screen

a problem class does not cause the instantiation of any models, but is used only to categorize related problem symptoms. Current problem classes illustrate the various ways we envisioned the system being used. Users can initiate a consultation either by test equipment error messages, specific function problems or mode B test failures. Upon selecting a problem class, the system displays a list of associated symptoms. It is the selection of a particular symptom that results in the instantiation of a model, and the subsequent

diagnosis. For instance, should the technician select either of the Mode B Diagnostic options, the system will respond with the screen depicted in Figure 21. This screen permits the selection of the mode B test which has failed and causes an ensuing diagnoses of that failure.

The Velocity Unreasonable is one of the 62 error messages generated by the test equipment and if selected will cause the running of the top level IMU model which will assist the technician in isolating the problem to one of the 11 major functions.

The 400Hz Reference Function option provides a diagnosis of the 400Hz reference. The tags, Power Good and Power Suspect, indicate whether or not the technician wants to assume the health of the power distribution circuitry. If the power is assumed good the resulting diagnosis will not query the user about power sources, as it would had the power suspect option been selected.

5.4 Summary

The resulting system is capable of diagnosing failures within the 400Hz Reference Function, as well as the Frequency Standard Function. Compared to Skinner's model-based system, this prototype is more complete in that it incorporates nearly all component connections and therefore provides a more thorough diagnosis. In addition, Skinner noted that since his model was a simple graph the system always initiated a point-by-point search. Our prototype is not as procedurally restricted as BDS appears to be. Using the reliability of components in the suspect list, as well as the cost of performing component probes and tests, our system dynamically adjusts the search based on the current candidate list.

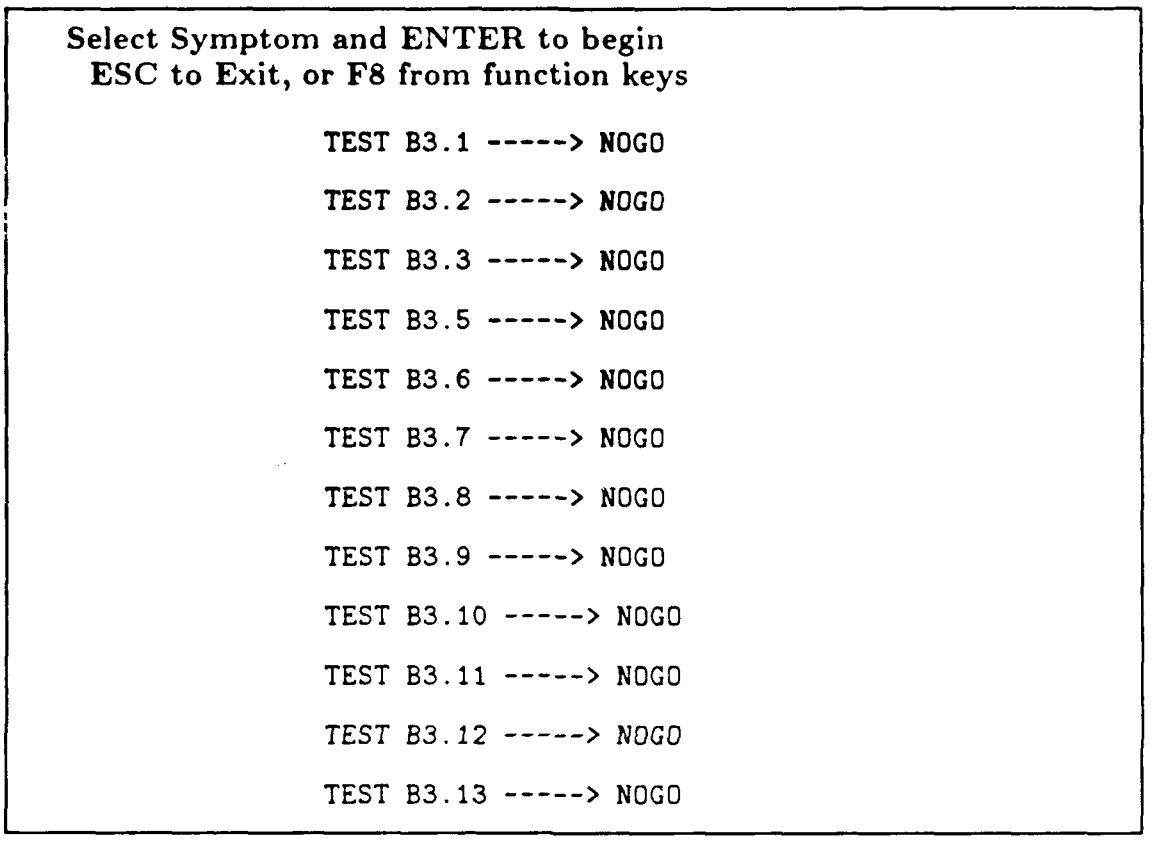

Figure 21. Fault Symptom Screen for Mode B Diagnostics

Though BDS is highly procedural it does demonstrate several innovative techniques in dealing with the limitations of rule-based systems in model-based diagnostics. By creating object attributes such as **RISK** and **TESTCOST** Skinner was able to explicitly prioritize the search. RISK was added as a measure of the likelihood of a component's being at fault based on its mean time between failure (MTBF). This attribute could be assigned HIGH, MED, or LOW accordingly. TEST COST was added as a measure of the difficulty to test a given component, and could take on the same qualitative values as RISK. By explicitly designating the initial level of the search as HIGH the system will only ask questions about

critical components. If after testing all critical components the fault is not identified, a medium level search would be conducted. If after testing all medium RISK components the fault is not identified, a low level search is conducted.

The bottom line is that BDS is highly procedural in nature. Any subsequent changes to the configuration of the IMU would warrant significant modifications to BDS. The models in our prototype are more explicit and would permit changes to be more readily incorporated. Whereas Skinner had to create additional attributes to explicitly control the search, our system inherently takes component reliability, test cost and other factors into consideration to dynamically control the search.

VI. Problems Encountered

This research provides valuable insight into problems with using constraint propagation as an underlying method in performing model-based diagnosis. Though not insurmountable these problems do provide us with interesting challenges for future work.

6.1 Feedback Problem

The primary obstacle in developing our prototype was the treatment of numerous feedback loops inherent in the inertial measuring unit (IMU). The constraint network representation of the feedback loops c used propagation problems and erroneous diagnoses. This section describes some of those problems and the methods used to solve them.

The 400Hz reference function generates two primary signals used in the IMU, as well as various status signals. The first is a 400Hz 40v peak-to-peak square wave generated by the triangle generator/case rotation power supply. This signal is fed to the platform torquing function and is used by the gimbal rate amplifiers as a demodulation reference. The mode B diagnostics monitor this signal with test B3.8. The second signal generated by this function is a 400Hz reference sine wave that is also fed to the platform torquing function. This se. ond signal is utilized as a phase reference (See Figure 24).

Before proceeding with this discussion it is imperative that the signal dependencies within the two 400Hz power supplies be further clarified. A portion of the problems encountered are a consequence of the unique dependencies in these two components. Figure 22 illustrates a slightly more detailed representation of the 400Hz power supply #2. As illustrated, this SRU is composed of two functionally independent circuit groupings. The

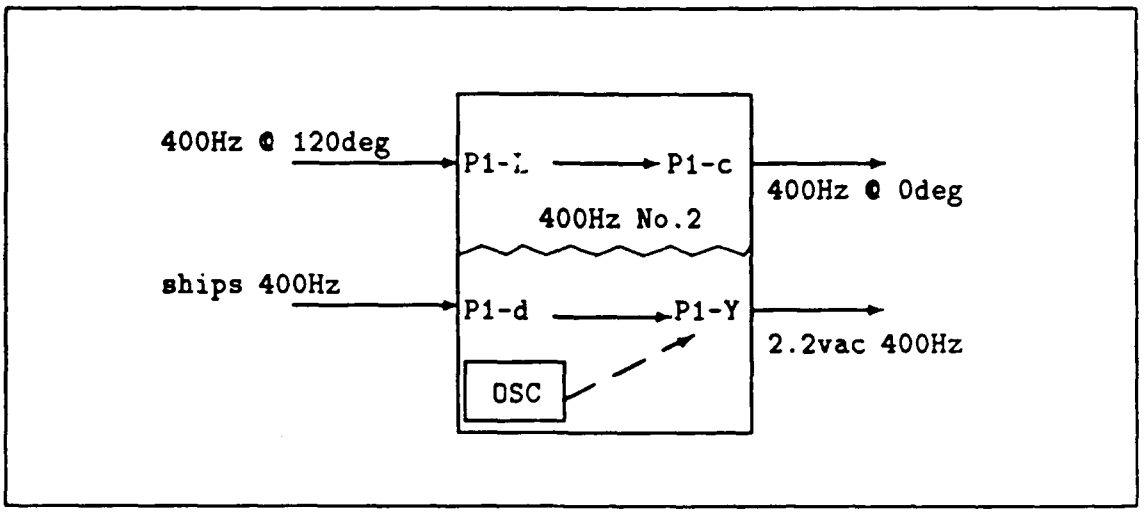

Figure 22. Signal dependencies in 400Hz Power No.2

bottom circuits sense the amplitude of the ship's 400Hz reference input at P1-d. If the amplitude of the reference is greater than approximately 18.5vRMS then the output at P1-Y is a function of the reference input at P1-d. If the input is below approximately 18.5vRMS the output at P1-Y is a function of an internal oscillator. Therefore the simulation clause for P1-Y only depends on the input signal at P1-d and power.

The top circuit has as its input an amplified and phase shifted version of the output at P1-Y. The output at P1-Y is fed to the 400Hz power supply #1 where its amplitude and phase are corrected and subsequently fed to P1-L (see Figure 24). This does not constitute a feedback loop. The output at P1-c of the 400Hz power #2 is dependent only on the input at P1-L, and therefore the simulation clause references only that signal and power. The actual simulation clause for P1-c, 400Hz-0deg is supplied in apprendix B section B.5.

Figure 23 illustrates the 400Hz power supply No. 1. In this SRU the signal at P1-H

is a direct feed from P1-a. Therefore the simulation clause for P1-H only depends on the input at P1-a. The output at P1-d is dependent not only on the input at P1-J but also on the signal at P1-a, which is feedback from the output of P1-d after step-up transformer T3-B (See Figure 24). The transformer output is used as excitation for gimbal position resolvers on the platform and both the amplitude and phase of this voltage must be closely controlled. Consequently the simulation clause for the output at P1-d refers to both signals at P1-a and P1-J, as well as the power sources. After developing an initial model of the

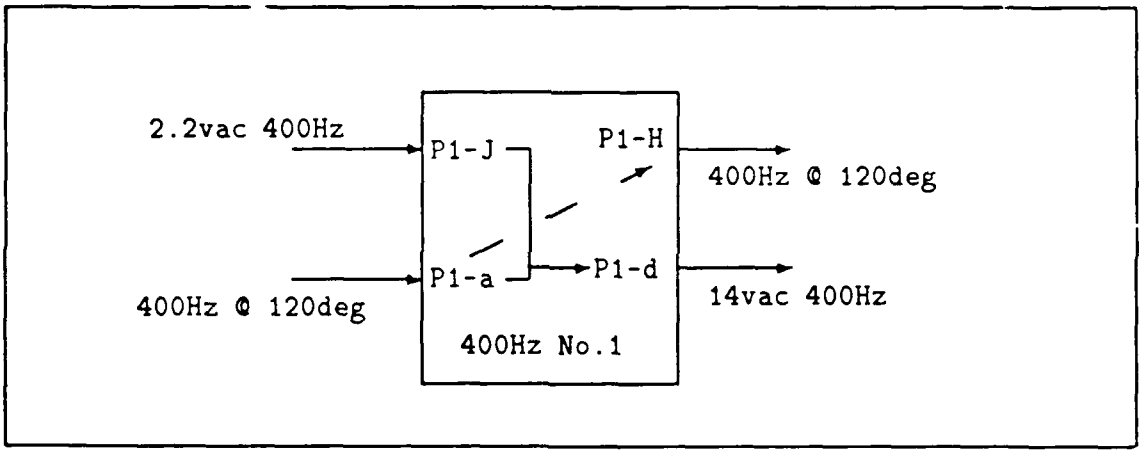

Figure 23. Signal Dependencies in 400Hz Power No.1

400Hz reference function we had a representation as illustrated in Figure 24. The diagram shows the existing feedback used to control both the gain and phase of the 400Hz sine wave generated by T3-B. IDEA provides a technique to handle this feedback by breaking the loop and inserting a module designated FEEDBACK. Our initial understanding of this feature led us to believe that IDEA would detect the feedback and provide an appropriate diagnosis. Subsequently, however, we learned that the feedback module was only used to prevent IDEA from entering an infinite loop during the constraint propagation.

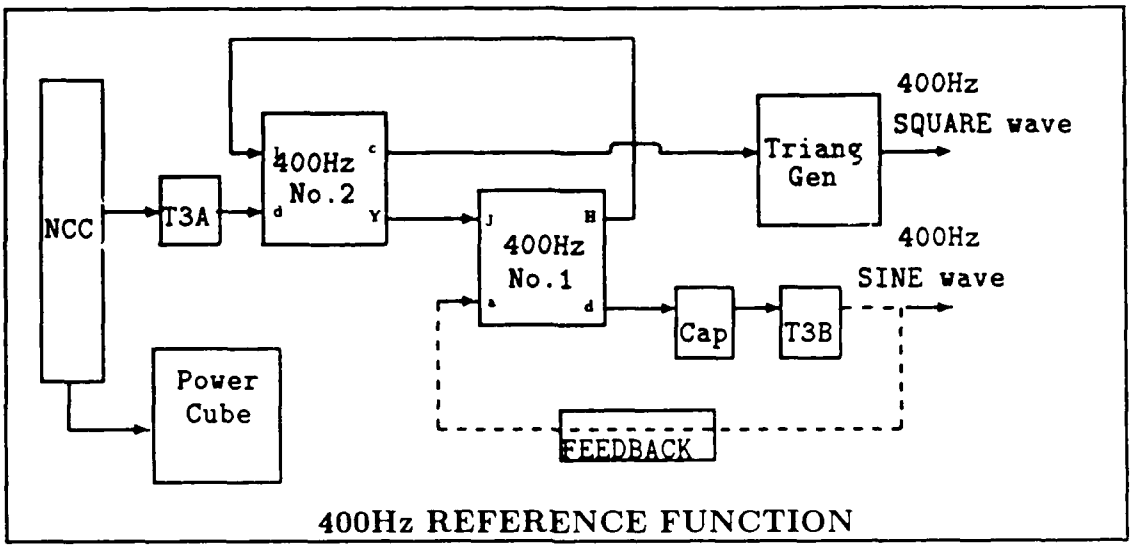

Figure 24. Initial 400Hz Model Representation

After breaking the feedback loop we defined a problem class to diagnose 400Hz square wave anomolies. If the user was experiencing a problem with the 400Hz square wave a model was instantiated with the output of the triangle generator set to 'BAD. At this point all of the modules are suspect since a functional and structural trace of the circuitry leads to all components. The diagnosis began with the system requesting the user to verify the bad triangle generator output. It then queried the user concerning the status of the 400Hz sine wave in an attempt to determine whether the feedback loop was contributing to the bad square wave output. If the user responded with 'GOOD the resulting propagation would correctly detect a discrepancy as illustrated in Figure 25. Through constraint suspension the system would subsequently eliminate, from the candidate list, the components composing the feedback loop . The system would then lead the user through a successful diagnosis of the remaining components. However, if the user responded to the

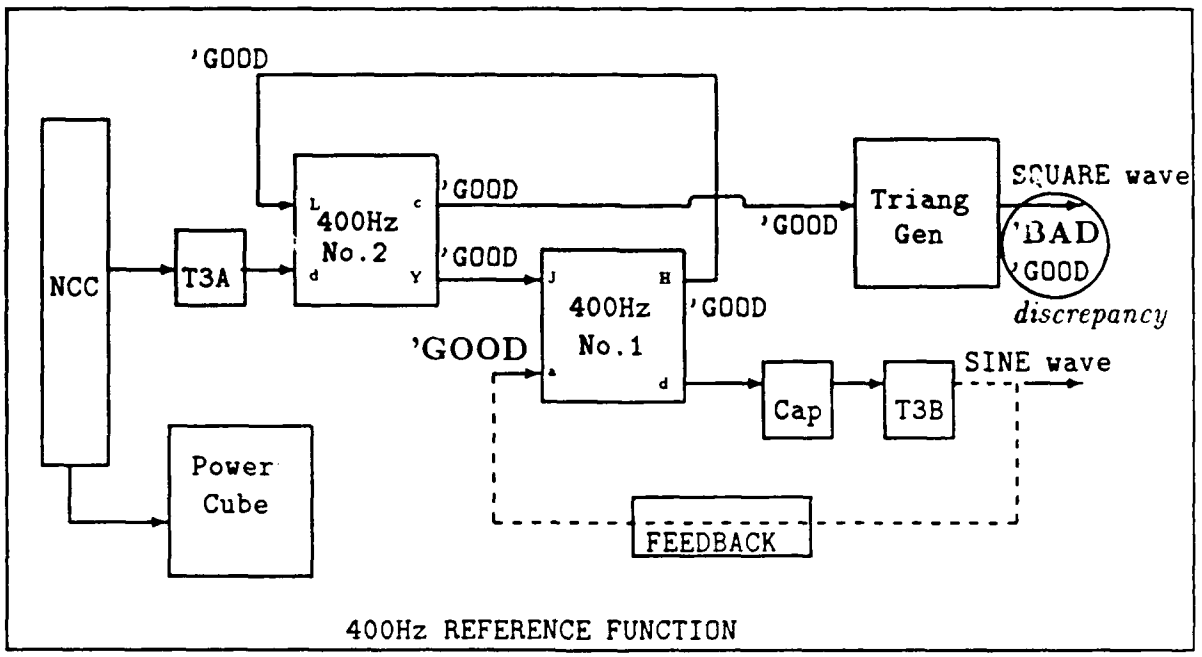

Figure 25. Correct Operation

system's query, concerning the 400Hz sine wave, with 'BAD the system would indicate that the 400Hz reference was functioning properly. Since the main objective in our use of constraint propagation is the detection of differences between actual system behaviors and model predictions we assumed that the system was failing to detect any discrepancies. Further investigation verified this, leading us to hypothesize that the original terminal symbols were not precise enough in expressing the feedback loop. However, as alluded to earlier, our original intentions were to initially describe the signals with relatively fundamental symbols. We fully expected to have to increase the expressiveness of our notations to adequately describe the device behaviors. In fact, the extremely primitive symbols prevented the system from identifying the feedback problem. As illustrated in Figure 26 the 'BAD sine wave symbol eventually propagates to the output of the triangle generator

which already has a 'BAD symbol and therefore **no discrepancy** would occur. Our initial

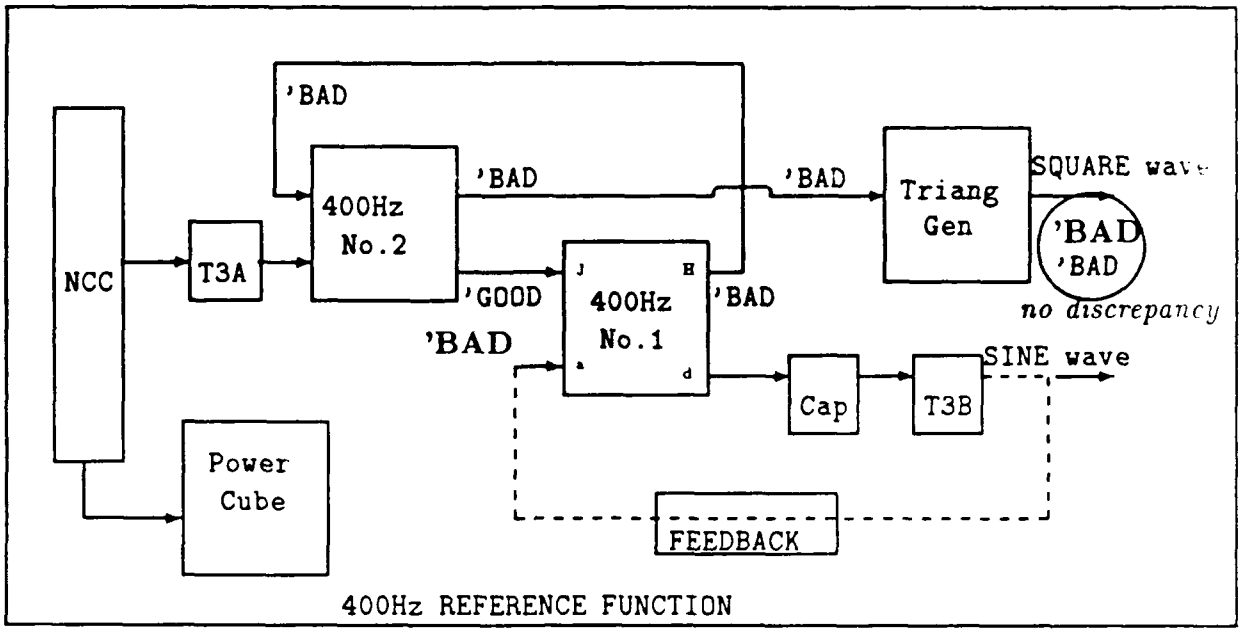

Figure 26. Inappropriate Operation

reaction was to extend the symbol list for the components in the feedback loop to allow a discrepancy to be detected for the preceding scenario. The symbol list for terminals **a** and **d** of the 400Hz #1 were extended to 'TOO-LOW, 'TOO-HIGH and 'GOOD. Subsequent queries concerning the transformer output now required a more precise characterization. If the signal was bad the technician would determine whether the signal was not being amplified enough, thereby resulting in a signal 'TOO-LOW, or was being amplified too much thereby resulting in a signal 'TOO-HIGH. The simulation clause for **d** was subsequently changed to propagate either 'TOO-LOW, 'TOO-HIGH or 'GOOD depending on the signals at **a** and **J**.

Upon incorporating these changes we performed another test using a bad triangle

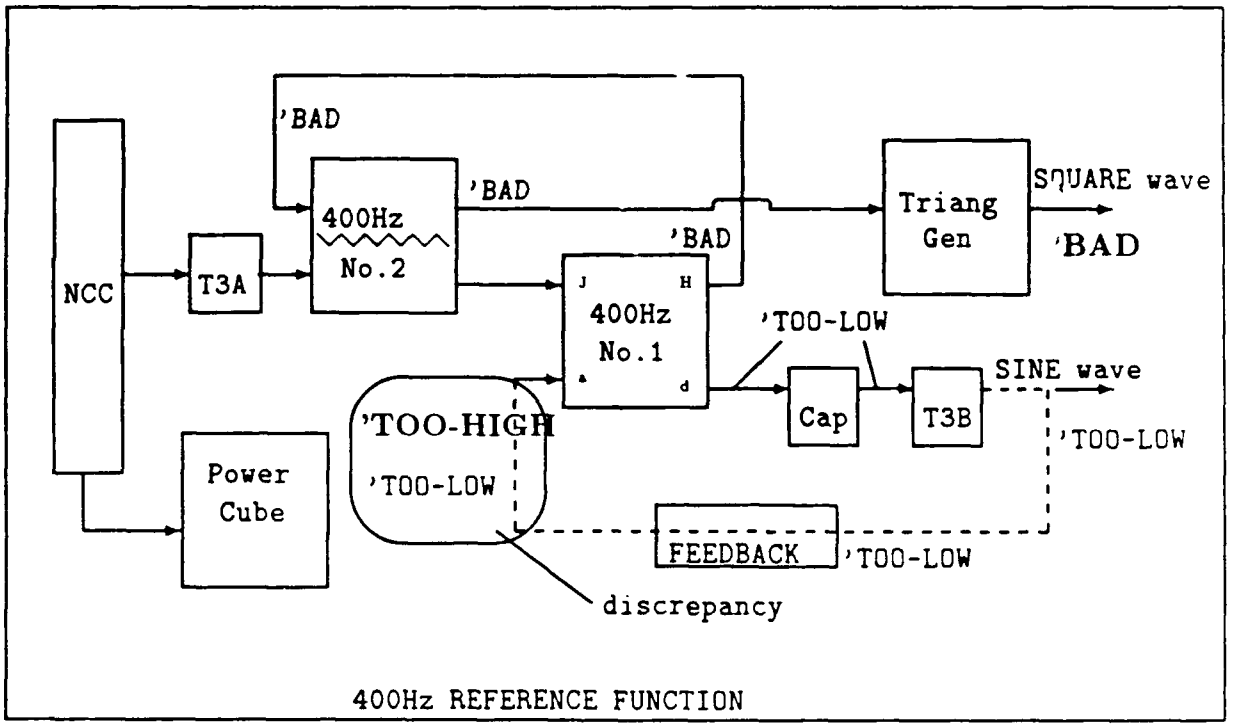

Figure 27. Expected Propagation with extended symbol list

generator output and a 'TOO-HIGH output at T3-B which was preset at **a**. The ensuing propagation was **expected** to be that illustrated in Figure 27, with the discrepancy occurring at **a**. This discrepancy would have been sufficient to cause the feedback components to be included in the candidate list. However, the system again indicated that the 400Hz reference was functioning properly. Subsequent analysis indicated that the propagation was stalling as diagramed in Figure 28. The user response of 'TOO-HIGH, for the transformer output, was set at terminal **a** where the simulation clause for **d** would use it to propagate a 'TOO-LOW. This symbol was then propagated through the capacitor and transformer T3-B but **not** the feedback module! Ensuing phone conversations with engineers at AI Squared Inc. resulted in the following discovery. During the initial constraint propagation

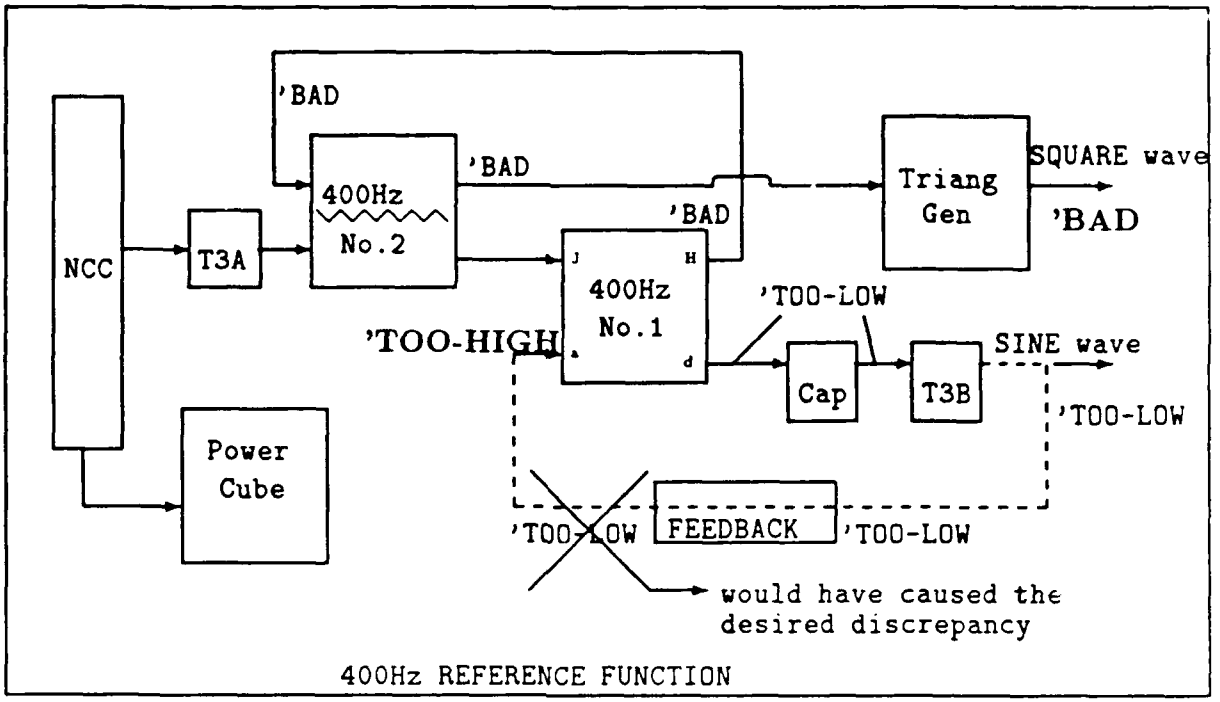

Figure 28. Extended Symbol List Propagation

the FEEDBACK module is turned off. This subsequently prevents the 'TOO-LOW from propagating through and creating our discrepancy. We were told that in past applications, feedback problems were identifiable prior to a diagnostic consultation. This allowed the models to be instantiated in a manner that explicitly caused the feedback components to become candidates. Unfortunately, the DMINS IMU incorporates a significant number of complex feedback circuits. IMU failures, caused by bad feedback circuits, are never easily attributable to those circuits. As illustrated in Figure 28 the initial suspect list, for the bad triangle generator output, consisted of every component in the 400Hz reference function, including the feedback components. This prevents us from identifying the feedback as the sole suspect prior to a diagnostic scenario.

At this point we were faced with two alternatives. The first solution, as well as the easiest, was to remove the entire feedback circuit and replace it with one module to designate the feedback components. If a subsequent diagnosis resulted in this module, representing the feedback circuit, a help file could be displayed to assist the technician in diagnosing the feedback. However, our inability to represent the actual circuit configuration using this method compelled us to disregard it as a viable solution. In order to more closely represent the actual circuit configuration and to include all circuit components in the model we opted to remove the feedback module and to alter the model slightly. Rather than explicitly representing the feedback loop, the two modules composing the loop were made dependent modules to the 400Hz power supply #1. The resulting representation of the 400Hz reference function is illustrated in Figure 29. The simulation clause for **H** was altered to include the values at **J** and **d** resulting in a simulation clause of the following form:

```
SIMULATE
  Terminal          H
  From Terminals    a  J  d
  Perform           set  H  to  J
```

This design propagates the value at J to H and the dummy terminal and eventually arrives at the output of transformer T3-B which allows a discrepancy to occur. Even though the actual simulation clause does not explicitly utilize the values at **a** or **d**, placing them in the From Terminals slot allows the components supplying those inputs to be declared testable. If a discrepancy occurs at the output of T3-B the capacitor and transformer

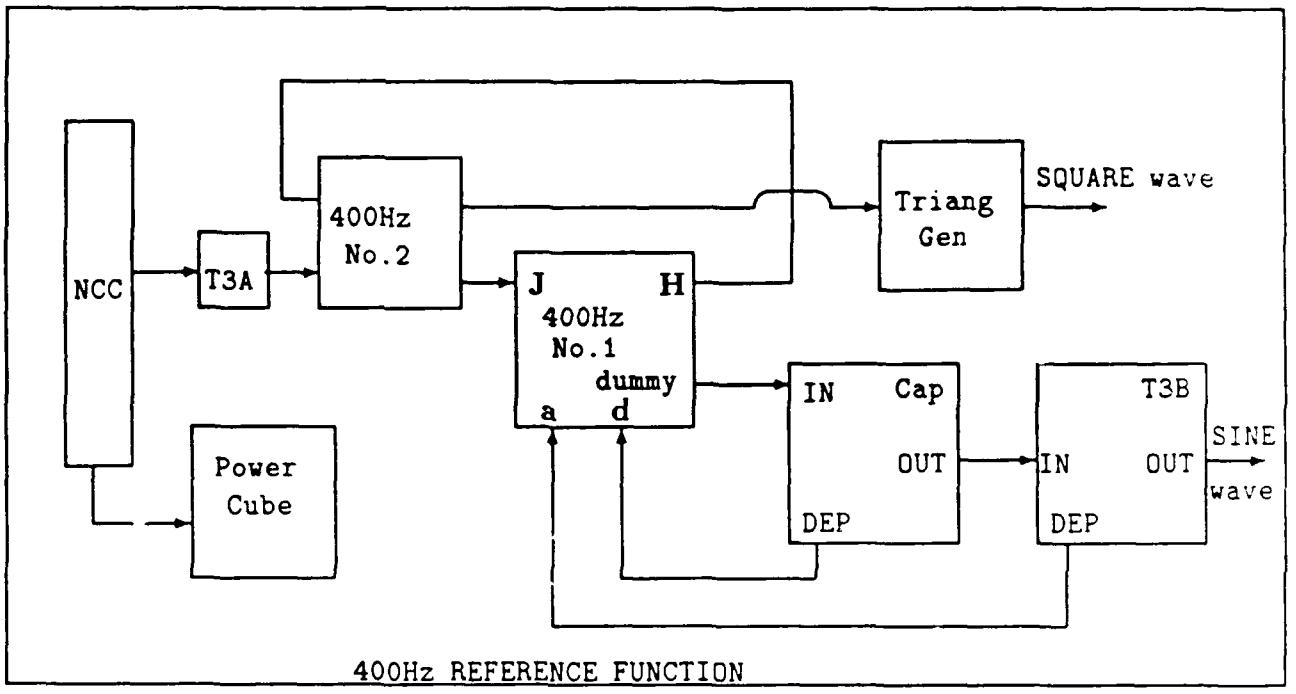

Figure 29. Final Representation of the 400Hz reference

become suspects, and can subsequently be tested. The symbol lists for **a** and **d** were changed back to the original symbol lists containing only 'GOOD and 'BAD.

Though the resulting model adequately diagnoses failures within the 400Hz Reference Function, there are several aspects of the representation that are undesirable. Foremost is the fact that the model does not represent the actual configuration of the 400Hz Reference Function. One of the attractive features of model-based diagnostics is that the underlying model represents the actual structure and function of the target device. In the DMINS, feedback is extremely prevalent and having to contrive constraint network representations to overcome propagation problems is unacceptable. In addition, the resulting representation prohibits the probing of certain valuable test points. For instance, the output of

transformer T3-B is accessible by technicians, but must be preset by our prototype because of the resulting circuit representation. Attempts to query the user concerning the 400Hz sine wave resulted in a system response of Multiple Failures. Therefore, our prototype presets the status of the 400Hz sine wave in the symptom class at the start of a consultation. Without access to the LISP code underlying IDEA, we were unable to determine the cause of this bug. During the development several bugs were uncovered and subsequently "fixed" by AI Squared, except for the one just mentioned.

6.2 Lack of a Hierarchical Diagnostic Strategy

Hierarchical diagnosis is a fundamental feature of most model-based systems. This results from the inherent human tendency to shift areas of focus when solving problems. Perhaps a technician begins with how a given device fits into its parent system then shifts his focus to a structural view of the device, and subsequently to an electrical representation. This characteristic allows a technician to apply knowledge only when absolutely necessary or when he has exhausted all knowledge at some level.

As previously mentioned BDS incorporates a hierarchical diagnostic strategy. Our prototype, however, was unable to incorporate such a strategy. IDEA does not currently provide an abstraction shifting mechanism. In other words, model components could only be described at one level of abstraction. For instance, the 400Hz Reference function can be envisioned at either a functional level or at a component level. However the current IDEA software only permits the behavioral description to encapsulate one of these views. Had an abstraction shifting mechanism been incorporated we could have more readily described each function at multiple levels of detail. Initially, the system could have diagnosed fault-

using a system level model incorporating appropriate behavioral descriptions. Once the system narrows the search space to a specific function within the IMU, it could shift to a more detailed representation of that particular function and the components composing it. This hierarchical strategy would inhibit the instantiation of detailed models until such level of detail was warranted. As a tradeoff we developed a top-level diagnostic model which is capable of isolating an IMU failure to one of the 11 major functions. Once a particular function is identified the technician must manually initiate another diagnostic consultation to begin a diagnosis of that function.

A modeling tool should allow a hierarchical diagnosis to start with the least complicated model possible and only introduce more detailed and complex models as necessary. As suggested, this strategy could allow multiple views of a device to be used in a systematic diagnosis of that device. This strategy also more closely mirrors the approach taken by experienced technicians.

6.3 Rule-based extension

Current rule-based expert systems handle specific faults rather well but do not offer wide coverage of all possible faults. Model-based systems offer more complete coverage but solve problems using the "first principles" approach every time.

Suppose a technician frequently encounters a specific problem which can be quickly diagnosed based on some heuristic. Further suppose, that the heuristic deals with how long the given device has been in operation or the type of environment it has been in. A model-based system would be unable to consider these factors. Instead it would resort to a first principles approach each and every time it encountered this problem. By system-

atically requesting device observations it would, it is hoped, correctly identify the failed component. The inefficiency lies in the system's inability to realize that the diagnosis just performed could have been been accomplished much easier using a heuristic. Since model-based techniques do not incorporate heuristics, AI Squared devised an alternative solution, referred to as the LEARN feature. Each model created with IDEA can have a file associated with it called a LEARN file. During each consultation the LEARN feature compares the current symptom information with past diagnoses contained in the appropriate LEARN file. If the current symptom information matches a past diagnosis exactly, the candidate list from the previous diagnosis is recalled. This provides a significant improvement in system performance, but still relies on a first principles approach. However, after a diagnostic scenario is captured in the LEARN file, subsequent encounters with that scenario will be substantially faster. This performance improvement is achieved by re-calling past candidate lists which eliminates the need to perform a complete model propagation.

Skinner's research keyed upon a very important aspect of implementing expert systems. A simple set of rules concerning how a particular device has malfunctioned in the past, which is accumulated over time, is only going to provide a superficial, or shallow, solution. Sooner or later a symptom will occur that hasn't been seen before and the system had better have a strategy to deal with it. Blending a rule-based system with a model-based system would be a step in attempting to solve this dilemma. In their next release AI Squared plans to add a rule-based system to IDEA in order to capture troubleshooting experience.

6.4 Component replacement in Hypothesis Discrimination

As discussed in chapter 3, in order to reduce the number of components possibly contributing to a fault, hypothesis discrimination requests further device observations . In testing IMUs, technicians often remove and replace suspected faulty circuit cards with other cards in attempting to exonerate that card. The problem we faced was the fact that replacement cards were often just as bad as the cards just removed. The IDEA software, however, demands that component tests fully implicate a component; otherwise it remains a candidate. That is, if the test passes, the module is no longer considered a candidate. If the test fails, the module is still considered a candidate. IDEA provides no probabilistic measure of the likelihood of that component's being bad given the number of times it is removed and replaced. Technicians have removed and replaced components dozens of times in the course of a single diagnosis before coming across a good one. IDEA was unable to handle this scenario since it only allows components to be tested once. After completing a given test the component is either flagged as being the malfunctioning component or will remain a candidate throughout the diagnosis. This appears to be more a failing of current AGMC troubleshooting procedures than of IDEA.

6.5 Summary

The problems we encountered were solved, though perhaps not always optimally. To some degree the problems can be attributed to the relatively new Intelligent Diagnostic Expert Assistant (IDEA) software. However, throughout the prototype development AI Squared was instrumental in providing updates to IDEA. This was especially true as bugs were identified. We are confident that our concerns, outlined in this chapter, will be

evaluated for possible future integration into IDEA.

VII. Conclusions and Recommendations

7.1 Summary

The Blended Diagnostic System (BDS) was developed as an investigation into the blending of a rule-based diagnostic system and a model-based system. The motivation for this thesis was a desire to improve upon the model-based reasoning techniques used in BDS. We used a constraint network representation to model the various subsystems in the Dual Miniature Inertial Navigation System (DMINS). This provided more complete and consistent fault coverage than did the methods employed by Skinner. The resulting prototype is capable of diagnosing problems within the 400Hz reference function and the frequency standard function.

In developing the prototype we gained an appreciation for the problems involved with implementing model-based techniques on complex problem domains. Particular problems arose in using a constraint network representation to handle circuit feedback. The nature of the problem domain, however, warrants that any subsequent implementation of model-based reasoning to inertial navigation equipment be able to handle feedback. Current literature on model-based reasoning is greatly influenced by the research efforts of Dr Randall Davis. In reviewing Davis' contributions (5, 9) to the field of model-based reasoning we discovered that AI Squared Inc., was developing and implementing a software tool based on his research. This tool, the Intelligent Diagnostic Expert Assistant (IDEA), was subsequently used to develop the models in our prototype. Our primary consideration in selecting IDEA, as our development tool, was its being an implementation of Davis' work. However, we were also intrigued by AI Squared's claim that IDEA was capable of

identifying and treating feedback. This, however, was not the case and made the subsequent development much more difficult. The DMINS IMU contains numerous multiple coupled feedback loops which were not easily modeled using IDEA. Even the relatively fundamental gain control feedback circuits were not easily modeled.

Perhaps the greatest hindrance in the development of the prototype was the actual lack of the LISP code underlying IDEA. Several problems arose which were subsequently traced to errors in that code.

Overall, the prototype development afforded us the opportunity to realize what features future model-based reasoning tools should include. These features are summarized below:

7.1.1 Hierarchical Diagnosis. One feature, characteristic of most current model based strategies, is a hierarchical diagnosis. Most troubleshooting scenarios are conducted by systematically breaking the target system into independent subsystems. Future model-based tools should incorporate the capability to model at various levels of this hierarchy.

7.1.2 High Resolution Graphics. IDEA uses a high resolution graphics interface in communicating with the user. This is extremely important in any expert system and was highly successful in our prototype.

7.1.3 Model Instantiation Speed. The speed with which the models can be instantiated must also be improved. Our prototype was developed on an IBM AT 80386, and took approximately 30-60 seconds to create the DMINS models. Though not excessive it is long considering that the models are relatively small. Had the models been much larger

the increased instantiation time would have been unacceptable. This would be especially true if the system ever enters a real maintenance environment. Perhaps the individual models could be compiled thereby making the instantiation faster.

7.1.4 Rule-Based Extension The assumption that a single tool or knowledge representation scheme will suffice is misleading. In developing our prototype we encountered numerous "heuristics" which our system could not accommodate. The incorporation of a rule-based system would allow a diagnosis to prune the problem space before resorting to a more detailed diagnosis using a model. AI Squared has stated that the next version of IDEA will incorporate some rule-based capabilities.

7.1.5 Feedback Future model-based tools should inherently identify and handle feedback. The constraint network representation we used made it extremely difficult to encode feedback, which was prevalent in the DMINS. AI Squared has stated that subsequent versions of IDEA will include this feature.

7.1.6 System Speed IDEA has an extensive user interface to help in system development. This interface, however, turned out to be an obstacle in our prototype development. There are approximately 70 different menu screens with a high degree of interaction. The speed of the development environment was extremely slow and cumbersome. Future model-based reasoning tools should exhibit a low response time in both development and delivery environments.

7.2 *Conclusion*

Our prototype adequately diagnoses IMU failures within the 400Hz reference function and the frequency standard function. It could be extended to handle all Mode B test failures, though the feedback problem, previously discussed, would hamper that development. However, the potential benefit of the resulting system does not appear to support the necessary effort and cost associated with that development.

Our experiences imply that a model-based system would only be suitable to the Mode B tests. Since those tests only contribute 4% to the average 70 hour IMU repair time, incorporation of such a system would be inappropriate.

Our review of the Blended Diagnostic System (BDS) revealed that its primary inputs, required to initiate a diagnostic consultation, were test equipment error codes. These are only a fraction of the criteria used by AGMC technicians in diagnosing IMU failures. Current efforts by Capt Florian, in extending the rule-based portion of BDS, are aimed at including Mode B test codes and Mode A performance parameters. In supplying a DMINS diagnostic system to the Aerospace Guidance and Metrology Center these efforts appear to be more appropriate than continuing to implement a model-based reasoning system using IDEA.

7.3 *Recommendations*

Maintenance of military equipment has been, and will continue to be, an expensive endeavor for the US government. Increased equipment complexity and rapid personnel turnover are two factors contributing to this problem. Advances in artificial intelligence,

in recent years, have begun to offer possible solutions to this problem in the form of rule-based expert systems and model-based reasoning. However, limitations of rule-based expert systems tend to make them difficult to develop rapidly, efficiently and -most importantly- correctly. This has motivated researchers to investigate capturing "deeper" knowledge, in the form of models, and reasoning about those models as the basis of a diagnostic system. As a result, future implementations of model-based diagnostics to military systems seems inevitable. To facilitate successful use of this technology it is imperative that we pursue the development of a model-based reasoning tool. Particular attention should be focused on implementing techniques to identify and manage feedback. Work by de Kleer and Williams (7) appears to provide some promising techniques in handling feedback. General Electric's corporate research and development center has implemented some of the techniques presented by de Kleer and Williams and was successful in developing a system to diagnose a major portion of a servo drive control system(6). This system contains analog circuitry, multiple coupled feedback loops, and on the order of a hundred base-level components. Future efforts should, perhaps, use that research as a basis for solving the feedback problems previously discussed in this thesis.

Appendix A. *MODE B TESTS*

TEST No.	SUBTEST No.	TEST FUNCTION
B1		**IMU Power Up and Temperature Tests**
	.1	IMU Thermal switch
	.2	IMU Overload
	.3	IMU Clock
	.4	AZ Gimbal Motor
	.5	IMU Airflow
B2		**DC Power Supply Tests**
	.1	DC Pwr Supply - +28v
	.2	DC Pwr Supply - +6v
	.3	DC Pwr Supply - -6v
	.4	DC Pwr Supply - +12v
	.5	DC Pwr Supply - -12v
	.6	DC Pwr Supply - +24v
	.7	DC Pwr Supply - -24v
	.8	DC Pwr Supply - +48v
	.9	DC Pwr Supply - +60v
B3		**AC Power Supply Tests**
	.1	AC Pwr Supply 115v REF
	.2	AC Pwr Supply 9.6KHz
	.3	AC Pwr Supply 6.72KHz
	.4	AC Pwr Supply 4.8KHz 0
	.5	AC Pwr Supply 4.8KHz 90
	.6	AC Pwr Supply 4.8KHz 90
	.7	AC Pwr Supply 640Hz 0
	.8	AC Pwr Supply 400Hz Case Rotation
	.9	AC Pwr Supply 80Hz 0
	.10	AC Pwr Supply 80Hz 0 Triangle
	.11	AC Pwr Supply 80Hz 120 deg
	.12	AC Pwr Supply 80Hz 0 deg
	.13	AC Pwr Supply 64Hz Clock

TEST No.	SUBTEST No.	TEST FUNCTION
B4		**Gyro Temperature Alarm Tests**
	.1	XY Gyro Hot Test
	.2	XY Gyro Cold Test
	.3	YZ Gyro Hot Test
	.4	YZ Gyro Cold Test
B5		**Thermoelectric Control Tests**
	.1	Heating
	.2	Cooling
B6		**Bite Status Checks**
	.1	XY Speed Control
	.2	YZ Speed Control
	.3	400Hz
	.4	Over Rate Enable
	.5	Servo Disable
	.6	Free Run
	.7	AZ Cage
B7		**Bite Operation Check**
	.1	400Hz (Servo Disable)
	.2	4.8Hz (Servo Disable)
	.3	400Hz Bite Test
	.4	AZ Overrate (Servo Disable)
	.5	Free Run Test
	.6	Free Run Reset
B8		**Cage Discrete Test**
	.1	AZ Gimbal Motor
	.2	Discrete Test
B9		**Resolver Presence**
	.1	Roll 2X
	.2	Roll 36X
	.3	Pitch 2X
	.4	Pitch 36X
	.5	Azimuth 1X
	.6	Azimuth 36X
B10		**Attitude Readout Tests**
	.1	Attitude Readout - Part 1
	.2	Attitude Readout - Part 2

TEST No.	SUBTEST No.	TEST FUNCTION
B11		**Gimbal Freedom Tests**
	.1	AZ Test C.W. - Pt 1
	.1	AZ Test C.W. - Pt 2
	.1	AZ Test C.W. - Pt 3
	.2	AZ Test C.C.W. - Pt 1
	.2	AZ Test C.C.W. - Pt 2
	.2	AZ Test C.C.W. - Pt 2A
	.3	Roll Test 0 deg
	.4	Roll Test +
	.5	Roll Test -
	.6	Pitch Test 0 deg
	.7	Pitch Test +50 deg
	.8	Pitch Test -50 deg
B12		**Gyro Torque Presence**
	.1	+ Gyro Torque
	.2	- Gyro Torque
B13		**Case Rotation Tests**
	.1	Case Rotation Test - Pt 1
	.2	Case Rotation Test - Pt 2
B14		**Gyro Start/Run**
	.0	Gyro Start / Run (Start)
	.1	Gyro Start / Run (Run)
	.2	XY Gyro Speed
	.3	YZ Gyro Speed
B15		**Stabilization Test**
	.1	Excess Angle
B16		**Speed Control Tests**
	.1	XY Gyro Speed Control
	.2	YZ Gyro Speed Control
B17		**Bit Edge Hold Test**
	.1	AZ Bit Edge Hold
	.2	Roll Bit Edge Hold
	.3	Pitch Bit Edge Hold
B18		**Gyro Torque Test**
	.1	X Gyro Torque (SX)
	.2	Y Gyro Torque (SY)
	.3	Z Gyro Torque (SZ)
B19		**Velocity Meter Reversal**
	.1	VM Reversal (X)
	.2	VM Reversal (Y) Pt 2

TEST No.	SUBTEST No.	TEST FUNCTION
B20		**Velocity Meter Stability Test**
	.1	VM Stability (YVM)
	.2	VM Stability (XVM)
B21		**Gyro Bottom**
	.1	AZ Excess Angle - Pt 1
	.1	AZ Excess Angle - Pt 2
	.1	AZ Excess Angle - Pt 2A
	.1	AZ Excess Angle - Pt 3
	.1	AZ Excess Angle - Pt 3A
	.1	AZ Excess Angle - Pt 4
	.2	Roll Excess Angle - Pt 1
	.2	Roll Excess Angle - Pt 2
	.2	Roll Excess Angle - Pt 2A
	.2	Roll Excess Angle - Pt 3
	.2	Roll Excess Angle - Pt 3A
	.2	Roll Excess Angle - Pt 4
	.3	Pitch Excess Angle - Pt 1
	.3	Pitch Excess Angle - Pt 2
	.3	Pitch Excess Angle - Pt 2A
	.3	Pitch Excess Angle - Pt 3
	.3	Pitch Excess Angle - Pt 3A
	.3	Pitch Excess Angle - Pt 4
B22		**Gyro Temperature Control Tests**
	.1	Gyro Temp Cont, α = 0 Pt 1
	.1	Gyro Temp Cont, α = 0 Pt 2
	.1	Gyro Temp Cont, α = 0 Pt 3
	.2	Gyro Temp Cont, α = 70 Pt 1
	.2	Gyro Temp Cont, α = 70 Pt 2
	.2	Gyro Temp Cont, α = 70 Pt 3
	.3	Gyro Temp Cont, α = 140 Pt 1
	.3	Gyro Temp Cont, α = 140 Pt 2
	.3	Gyro Temp Cont, α = 140 Pt 3
	.4	Gyro Temp Cont, α = 220 Pt 1
	.4	Gyro Temp Cont, α = 220 Pt 2
	.4	Gyro Temp Cont, α = 220 Pt 3
	.5	Gyro Temp Cont, α = 290 Pt 1
	.5	Gyro Temp Cont, α = 290 Pt 2
	.5	Gyro Temp Cont, α = 290 Pt 3
B23		**Periodic Temperature Tests**
	.1	Periodic Temp Test Pt 1
	.1	Periodic Temp Test Pt 2
	.1	Periodic Temp Test Pt 3

Appendix B. *DMINS Behaviors*

B.1 NAVIGATION CONTROL CONSOLE

```
Behavior Definition: NCC
 TERMINAL
    Name                    POS28V-OUT
    Direction               OUTPUT
    Comment


  TERMINAL
    Name                    SWITCH-ON
    Direction               INPUT
    Comment


  TERMINAL
    Name                    400HZ-115V
    Direction               OUTPUT
    Comment


  SIMULATE
    Terminal                POS28V-OUT
    From Terminals          SWITCH-ON
    Perform                 if @switch-on = 'ON
                             then set @pos28v-out to 'GOOD;
    Comment


  SIMULATE
    Terminal                400HZ-115V
    From Terminals          SWITCH-ON
    Perform                 if @SWITCH-ON = 'ON
                             then set @400hZ-115V to 'GOOD;
    Comment
```

B.2 4.8KHz Power Supply

Behavior Definition: 4-8KHZ-PWR

TERMINAL	
Name	POS24V-IN
Direction	INPUT
Comment	+24volt input from the power cube. (+24volt supply 2)
TERMINAL	
Name	NEG24V-IN
Direction	INPUT
Comment	-24volt input from the power cube.
TERMINAL	
Name	4-8KHZ-IN
Direction	INPUT
Comment	(P1-X) 4.8KHz input from frequency standard
TERMINAL	
Name	4-8KHZ-REF
Direction	OUTPUT
Comment	(P1- H) 4.8KHz reference ,clean version of freq std 4.8KHz
TERMINAL	
Name	POS12V-IN
Direction	INPUT
Comment	+12v from the power cube.
TERMINAL	
Name	48-100V-IN
Direction	INPUT
Comment	(P1-C) 4.8KHz 100vac feedback from transformer and output at P1-H
TERMINAL	
Name	4-8KHZ-270
Direction	OUTPUT
Comment	(P1-J) 4.8KHz at 270deg phase shift
TERMINAL	
Name	4-8KHZ-90
Direction	OUTPUT

Comment	4.8KHz at 90deg phase shift
TERMINAL	
Name	4-8KHZTEST
Direction	OUTPUT
Comment	

```
SIMULATE
  Terminal              4-8KHZ-REF
  From Terminals        48-100V-IN, POS12V-IN, 4-8KHZ-IN, NEG24V-IN,
                        POS24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @POS24V-IN @NEG24V-IN @48-100V-IN
                        @4-8KHZ-IN @4-8KHZ-REF:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                        'BAD    ??    ??    ??    ??  'BAD
                          ??  'BAD    ??    ??    ??  'BAD
                          ??    ??  'BAD    ??    ??  'BAD
                          ??    ??    ??  'BAD    ??  'BAD
                          ??    ??    ??    ??  'BAD  'BAD;
  Comment               (P1-H)  4.8KHz ref dependent on pwr the 4.8KHz
                        from freq std and fdbk

SIMULATE
  Terminal              4-8KHZ-270
  From Terminals        48-100V-IN, POS12V-IN, NEG24V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS24V-IN @NEG24V-IN @POS12V-IN @48-100V-IN
                        @4-8KHZ-270:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                        'BAD    ??    ??    ??  'BAD
                          ??  'BAD    ??    ??  'BAD
                          ??    ??  'BAD    ??  'BAD
                          ??    ??    ??  'BAD  'BAD;
  Comment               (P1-J) dependent on pwr and feedback to P1-C,
                        4.8KHz 100vac

SIMULATE
  Terminal              4-8KHZ-90
  From Terminals        48-100V-IN, POS12V-IN, NEG24V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS24V-IN @NEG24V-IN @POS12V-IN @48-100V-IN
                        @4-8KHZ-90:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                        'BAD    ??    ??    ??  'BAD
                          ??  'BAD    ??    ??  'BAD
                          ??    ??  'BAD    ??  'BAD
                          ??    ??    ??  'BAD  'BAD;
  Comment               (P1-F) dependent on power and feedback to P1-C
```

```
SIMULATE
  Terminal              4-8KHZTEST
  From Terminals        48-100V-IN, POS12V-IN, NEG24V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @48-100V-IN @POS12V-IN @NEG24V-IN @POS24V-IN
                         @4-8KHZTEST:
                        'GOOD 'GOOD 'GOOD 'GOOL 'GOOD
                        'BAD    ??    ??    ??  'BAD
                          ??  'BAD    ??    ??  'BAD
                          ??    ??  'BAD    ??  'BAD
                          ??    ??    ??  'BAD  'BAD;
  Comment               this was added to allow for the creation of TEST
                        B3.6 NOGO
```

```
INFER
  Terminal              48-100V-IN
  From Terminals        4-8KHZ-90, 4-8KHZ-270, POS12V-IN, NEG24V-IN,
                        POS24V-IN
  Perform               TRUTH TABLE
                        @POS24V-IN @POS12V-IN @NEG24V-IN @4-8KHZ-270
                        @4-8KHZ-90 @48-100V-IN:
                          ??    ??    ??  'GOOD   ??  'GOOD
                          ??    ??    ??    ??  'GOOD 'GOOD
                        'GOOD 'GOOD 'GOOD 'BAD    ??  'BAD
                        'GOOD 'GOOD 'GOOD   ??  'BAD  'BAD;
  Comment

INFER
  Terminal              4-8KHZ-IN
  From Terminals        48-100V-IN, POS12V-IN, 4-8KHZ-REF, NEG24V-IN,
                        POS24V-IN
  Perform               TRUTH TABLE
                        @POS24V-IN @POS12V-IN @NEG24V-IN @4-8KHZ-REF
                        @48-100V-IN  @4-8KHZ-IN:
                          ??    ??    ??  'GOOD   ??  'GOOD
                        'GOOD 'GOOD 'GOOD 'BAD  'GOOD 'BAD;
  Comment
```

B.3 Frequency Standard

```
Behavior Definition: FREQ-STD
  TERMINAL
    Name                POS24V-IN
    Direction           INPUT
    Comment             This is the postive 24 volt input from +24v No.2/
                        in the Power Cube

  TERMINAL
    Name                NEG6V-IN
    Direction           INPUT
    Comment             This is the -6v input from the Power Cube.

  TERMINAL
    Name                POS6V-IN
    Direction           INPUT
    Comment             This is the +6v input from the Power Cube

  TERMINAL
    Name                80HZ-120DG
    Direction           OUTPUT
    Comment             (P1-E) to triangle generator.

  TERMINAL
    Name                80HZ-REF
    Direction           OUTPUT
    Comment             (P1-D) to triang generator pwr: NCC TEST pt @ J3-K

  TERMINAL
    Name                4-8KHZ-REF
    Direction           OUTPUT
    Comment             4.8KHz reference to 4.8KHz pwr supply.  (P1-W)

  TERMINAL
    Name                POS12V-IN
    Direction           INPUT
    Comment

  TERMINAL
    Name                6720HZ
    Direction           OUTPUT
    Comment             (P1-U) to gyro speed control function:
```

TERMINAL

Name	640HZ-90
Direction	OUTPUT
Comment	(P1-L) to gyro speed control REFERENCE SIGNAL

TERMINAL

Name	640HZ-210
Direction	OUTPUT
Comment	(P1-Z) to gyro speed control function. REFERENCE SIGNAL

TERMINAL

Name	640HZ-120
Direction	OUTPUT
Comment	(P1-F) to gyro speed control function REFERENCE SIGNAL

TERMINAL

Name	640HZ-REF
Direction	OUTPUT
Comment	(P1-S) to gyro speed control function. REFERENCE SIGNAL NCC TEST @ J3-N

TERMINAL

Name	9-6KHZ
Direction	OUTPUT
Comment	9.6KHz ref to synchro signal buffer amp for 9.6KHz triangle wave.

TERMINAL

Name	64HZ-REF
Direction	OUTPUT
Comment	master timing signal sent to NCC

TERMINAL

Name	4-8KHZ-TST
Direction	OUTPUT
Comment	this was added to allow for the creation of TEST B3.5

```
SIMULATE
  Terminal              80HZ-120DG
  From Terminals        POS12V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS12v-IN @NEG6V-IN @POS24V-IN  @80HZ-120DG:
                        'good      'good      'good      'good
                        'bad         ??         ??       'bad
                         ??        'bad         ??       'bad
                         ??         ??       'bad       'bad;
  Comment               the 80hz/120 degree signal requires both the -6v
                        and +24v.

SIMULATE
  Terminal              80HZ-REF
  From Terminals        POS12V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @NEG6V-IN   @POS24V-IN   @POS12V-IN   @80HZ-REF:
                        'good         'good       'good      'good
                        'bad            ??         ??       'bad
                         ??           'bad         ??       'bad
                         ??             ??       'bad       'bad;
  Comment               the 80hz reference signal requires +6v, +12v and
                        +24v for operation

SIMULATE
  Terminal              4-8KHZ-REF
  From Terminals        POS12V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG6V-IN @POS24V-IN @4-8KHZ-REF:
                        'GOOD 'GOOD 'GOOD  'good
                        'BAD    ??    ??   'bad
                          ??  'BAD    ??   'bad
                          ??    ??  'BAD   'bad;

SIMULATE
  Terminal              9-6KHZ
  From Terminals        POS12V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG6V-IN @POS24V-IN @9-6KHZ:
                        'GOOD 'GOOD 'GOOD  'good
                        'BAD    ??    ??   'bad
                          ??  'BAD    ??   'bad
                          ??    ??  'BAD   'bad;
  Comment
```

```
SIMULATE
  Terminal              640HZ-REF
  From Terminals        POS12V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG6V-IN @POS24V-IN @640HZ-REF:
                        'GOOD 'GOOD 'GOOD 'good
                        'BAD    ??    ??  'bad
                          ??  'BAD    ??  'bad
                          ??    ??  'BAD  'bad;
  Comment

SIMULATE
  Terminal              640HZ-90
  From Terminals        POS12V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG6V-IN @POS24V-IN @640HZ-90:
                        'GOOD 'GOOD 'GOOD 'good
                        'BAD    ??    ??  'bad
                          ??  'BAD    ??  'bad
                          ??    ??  'BAD  'bad;
  Comment

SIMULATE
  Terminal              640HZ-210
  From Terminals        POS12V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG6V-IN @POS24V-IN @640HZ-210:
                        'GOOD 'GOOD 'GOOD 'good
                        'BAD    ??    ??  'bad
                          ??  'BAD    ??  'bad
                          ??    ??  'BAD  'bad;
  Comment

SIMULATE
  Terminal              640HZ-120
  From Terminals        POS12V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG6V-IN @POS24V-IN @640HZ-120:
                        'GOOD 'GOOD 'GOOD 'good
                        'BAD    ??    ??  'bad
                          ??  'BAD    ??  'bad
                          ??    ??  'BAD  'bad;
  Comment
```

```
SIMULATE
  Terminal              6720HZ
  From Terminals        POS12V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG6V-IN @POS24V-IN @6720HZ:
                        'GOOD 'GOOD 'GOOD 'good
                        'BAD    ??    ??  'bad
                          ??  'BAD    ??  'bad
                          ??    ??  'BAD  'bad;
  Comment

SIMULATE
  Terminal              64HZ-REF
  From Terminals        POS12V-IN, POS6V-IN, NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @POS24V-IN @NEG6V-IN @POS6V-IN
                        @64HZ-REF:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                        'BAD    ??    ??    ??  'BAD
                          ??  'BAD    ??    ??  'BAD
                          ??    ??  'BAD    ??  'BAD
                          ??    ??    ??  'BAD  'BAD;
  Comment

SIMULATE
  Terminal              4-8KHZ-TST
  From Terminals        @POS12V-IN @NEG6V-IN @POS24V-IN @4-8KHZ-TST
  Perform               'GOOD 'GOOD 'GOOD 'GOOD
                        'BAD    ??    ??  'BAD
                          ??  'BAD    ??  'BAD
                          ??    ??  'BAD  'BAD;
  Comment
```

```
INFER
  Terminal              POS24V-IN
  From Terminals        9-6KHZ, POS12V-IN, 4-8KHZ-REF, 80HZ-REF,
                        80HZ-120DG, NEG6V-IN
  Perform               TRUTH TABLE
                        @4-8KHZ-REF @80HZ-REF @80HZ-120DG @9-6KHZ
                        @POS12V-IN @NEG6V-IN    @POS24V-IN:
                        'GOOD   ??    ??    ??    ??    ??  'GOOD
                          ??  'GOOD   ??    ??    ??    ??  'GOOD
                          ??    ??  'GOOD   ??    ??    ??  'GOOD
                          ??    ??    ??  'GOOD   ??    ??  'GOOD
                        'BAD    ??    ??    ??  'GOOD 'GOOD 'BAD
                          ??  'BAD    ??    ??  'GOOD 'GOOD 'BAD
                          ??    ??  'BAD    ??  'GOOD 'GOOD 'BAD
                          ??    ??    ??  'BAD  'GOOD 'GOOD 'BAD,
  Comment               INFER using only outputs in FREQ-STD function. 2nd
                        INFER for Gyro SPD

INFER
  Terminal              POS24V-IN
  From Terminals        640HZ-REF, 640HZ-120, 640HZ-210, 640HZ-90, 6720HZ,
                        POS12V-IN, NEG6V-IN
  Perform               TRUTH TABLE
                        @640HZ-210 @640HZ-90 @640HZ-120 @640HZ-REF
                        @6720HZ @POS12V-IN @NEG6V-IN @POS24V-IN:
                        'GOOD   ??    ??    ??    ??    ??    ??  'GOOD
                          ??  'GOOD   ??    ??    ??    ??    ??  'GOOD
                          ??    ??  'GOOD   ??    ??    ??    ??  'GOOD
                          ??    ??    ??  'GOOD   ??    ??    ??  'GOOD
                          ??    ??    ??    ??  'GOOD   ??    ??  'GOOD
                        'BAD    ??    ??    ??    ??  'GOOD 'GOOD 'BAD
                          ??  'BAD    ??    ??    ??  'GOOD 'GOOD 'BAD
                          ??    ??  'BAD    ??    ??  'GOOD 'GOOD 'BAD
                          ??    ??    ??  'BAD    ??  'GOOD 'GOOD 'BAD
                          ??    ??    ??    ??  'BAD  'GOOD 'GOOD 'BAD;
  Comment               INFER using outputs to Gyro Speed control
```

```
INFER
  Terminal              NEG6V-IN
  From Terminals        9-6KHZ, POS12V-IN, 4-8KHZ-REF, 80HZ-REF,
                        80HZ-120DG, POS24V-IN
  Perform               TRUTH TABLE
                        @9-6KHZ @4-8KHZ-REF @80HZ-REF 80HZ-120DG
                        @POS12V-IN @POS24V-IN    @NEG6V-IN:
                        'GOOD   ??    ??    ??    ??    ??  'GOOD
                          ??  'GOOD   ??    ??    ??    ??  'GOOD
                          ??    ??  'GOOD   ??    ??    ??  'GOOD
                          ??    ??    ??  'GOOD   ??    ??  'GOOD
                        'BAD    ??    ??    ??  'GOOD 'GOOD 'BAD
                          ??  'BAD    ??    ??  'GOOD 'GOOD 'BAD
                          ??    ??  'BAD    ??  'GOOD 'GOOD 'BAD
                          ??    ??    ??  'BAD  'GOOD 'GOOD 'BAD;
  Comment

INFER
  Terminal              NEG6V-IN
  From Terminals        640HZ-REF, 640HZ-120, 640HZ-210, 640HZ-90, 6720HZ,
                        POS12V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @640HZ-210 @640HZ-90 @640HZ-120 @640HZ-REF
                        @6720HZ @POS24V-IN @POS12V-IN  @NEG6V-IN:
                        'GOOD   ??    ??    ??    ??    ??    ??  'GOOD
                          ??  'GOOD   ??    ??    ??    ??    ??  'GOOD
                          ??    ??  'GOOD   ??    ??    ??    ??  'GOOD
                          ??    ??    ??  'GOOD   ??    ??    ??  'GOOD
                          ??    ??    ??    ??  'GOOD   ??    ??  'GOOD
                        'BAD    ??    ??    ??    ??  'GOOD 'GOOD 'BAD
                          ??  'BAD    ??    ??    ??  'GOOD 'GOOD 'BAD
                          ??    ??  'BAD    ??    ??  'GOOD 'GOOD 'BAD
                          ??    ??    ??  'BAD    ??  'GOOD 'GOOD 'BAD
                          ??    ??    ??    ??  'BAD  'GOOD 'GOOD 'BAD;
  Comment
```

```
INFER
  Terminal              POS12V-IN
  From Terminals        9-6KHZ, 4-8KHZ-REF, 80HZ-REF, 80HZ-120DG,
                        NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @4-8KHZ-REF @80HZ-REF @80HZ-120DG @9-6KHZ
                         @POS24V-IN @NEG6V-IN @POS12V-IN:
                        'GOOD   ??    ??    ??    ??    ??   'GOOD
                          ??  'GOOD   ??    ??    ??    ??   'GOOD
                          ??    ??  'GOOD   ??    ??    ??   'GOOD
                          ??    ??    ??  'GOOD   ??    ??   'GOOD
                        'BAD    ??    ??    ??  'GOOD 'GOOD 'BAD
                          ??  'BAD    ??    ??  'GOOD 'GOOD 'BAD
                          ??    ??  'BAD    ??  'GOOD 'GOOD 'BAD
                          ??    ??    ??  'BAD  'GOOD 'GOOD 'BAD;
  Comment

INFER
  Terminal              POS12V-IN
  From Terminals        640HZ-REF, 640HZ-120, 640HZ-210, 640HZ-90, 6720HZ,
                        NEG6V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @640HZ-210 @640HZ-90 @640HZ-120 @640HZ-REF
                        @6720HZ @POS24V-IN @NEG6V-IN  @POS12V-IN:
                        'GOOD   ??    ??    ??    ??    ??    ??   'GOOD
                          ??  'GOOD   ??    ??    ??    ??    ??   'GOOD
                          ??    ??  'GOOD   ??    ??    ??    ??   'GOOD
                          ??    ??    ??  'GOOD   ??    ??    ??   'GOOD
                          ??    ??    ??    ??  'GOOD   ??    ??   'GOOD
                        'BAD    ??    ??    ??    ??  'GOOD 'GOOD 'BAD
                          ??  'BAD    ??    ??    ??  'GOOD 'GOOD 'BAD
                          ??    ??  'BAD    ??    ??  'GOOD 'GOOD 'BAD
                          ??    ??    ??  'BAD    ??  'GOOD 'GOOD 'BAD
                          ??    ??    ??    ??  'BAD  'GOOD 'GOOD 'BAD;
  Comment
```

B.4 Power Cube

```
Behavior Definition: PWR-CUBE
  TERMINAL
    Name                POS28V-IN
    Direction           INPUT
    Comment             +28v from the NCC.

  TERMINAL
    Name                NEG-24V
    Direction           OUTPUT
    Comment             -24v Test point @ J3-Z

  TERMINAL
    Name                POS24V-NO2
    Direction           OUTPUT
    Comment             +24v Test point @ J3-V

  TERMINAL
    Name                NEG-6V
    Direction           OUTPUT
    Comment             -6v Test point @ J3-E

  TERMINAL
    Name                POS-6V
    Direction           OUTPUT
    Comment             +6v Test at NCC test panel with volt meter select
                        switch

  TERMINAL
    Name                POS-48V
    Direction           OUTPUT
    Comment             +48v Test point J3-F

  TERMINAL
    Name                POS24V-NO1
    Direction           OUTPUT
    Comment

  TERMINAL
    Name                IMU-PWROFF
    Direction           OUTPUT
    Comment             indicates power not applied to IMU. Currently no
                        propagation here!
```

```
TERMINAL
  Name                POS-12V
  Direction           OUTPUT
  Comment             +12v Test point @ J3-W

TERMINAL
  Name                NEG-12V
  Direction           OUTPUT
  Comment             -12v, Test pt @ J3-J

SIMULATE
  Terminal            NEG-24V
  From Terminals      POS28V-IN
  Perform             SET @NEG-24V TO @POS28V-IN;
  Comment

SIMULATE
  Terminal            NEG-6V
  From Terminals      POS28V-IN
  Perform             SET @NEG-6V TO @POS28V-IN;
  Comment

SIMULATE
  Terminal            POS-6V
  From Terminals      POS28V-IN
  Perform             SET @POS-6V TO @POS28V-IN;
  Comment

SIMULATE
  Terminal            POS-48V
  From Terminals      POS28V-IN
  Perform             SET @POS-48V TO @POS28V-IN;
  Comment

SIMULATE
  Terminal            POS24V-NO2
  From Terminals      POS28V-IN
  Perform             SET @POS24V-NO2 TO @POS28V-IN;
  Comment
```

```
SIMULATE
  Terminal              POS24V-NO1
  From Terminals        POS28V-IN
  Perform               SET @POS24V-NO1 TO @POS28V-IN;
  Comment

SIMULATE
  Terminal              NEG-12V
  From Terminals        POS28V-IN
  Perform               SET @NEG-12V TO @POS28V-IN;
  Comment

SIMULATE
  Terminal              POS-12V
  From Terminals        POS28V-IN
  Perform               SET @POS-12V TO @POS28V-IN;
  Comment
```

```
INFER
  Terminal              POS28V-IN
  From Terminals        NEG-12V, POS-12V, POS24V-NO1, POS-48V, POS-6V,
                        NEG-6V, POS24V-NO2, NEG-24V
  Perform               IF  @neg-12v     = 'good OR
                            @pos-12v     = 'good OR
                            @pos24v-no1  = 'good OR
                            @pos-48v     = 'good OR
                            @pos24v-no2  = 'good OR
                            @pos-6v      = 'good OR
                            @neg-6v      = 'good OR
                            @neg-24v     = 'good
                         THEN SET @pos28v-in TO 'good;
  Comment
```

B.5 400Hz Power Supply No.2

```
Behavior Definition: 400HZ-PWR-N02
  TERMINAL
    Name                POS24V-IN
    Direction           INPUT
    Comment             +24volt input from the power cube. (+24volt suppl
                        2) IMUIC jumper:J69

  TERMINAL
    Name                NEG24V-IN
    Direction           INPUT
    Comment             -24volt input from the power cube. IMUIC jumper:
                        J/2

  TERMINAL
    Name                400HZ-2VAC
    Direction           OUTPUT
    Comment             (P1-Y) 400Hz-2.2vac output used for platform
                        control 'adequate->OSC-1

  TERMINAL
    Name                POS6V-IN
    Direction           INPUT
    Comment             +6volt from the power cube. Measurable at NCC test
                        panel. pp31 0-Level

  TERMINAL
    Name                400-26VAC
    Direction           INPUT
    Comment             (P1-d)  400Hz-26vac input from the NCC 115vac400Hz
                        pwrdistr & T3-A

  TERMINAL
    Name                400HZ-120
    Direction           INPUT
    Comment             (P1-L)  400Hz 26vac at 120deg from P1-H of
                        400Hz-PWR-N01 and plug xA8.

  TERMINAL
    Name                POS12V-IN
    Direction           INPUT
    Comment             +12v input from the pwr cube. IMUIC jumper:J70
```

```
TERMINAL
  Name                  NEG12V-IN
  Direction             INPUT
  Comment               -12v input from the pwr cube. IMUIC jumper:J59

TERMINAL
  Name                  FREE-RUN
  Direction             OUTPUT
  Comment               discrete indicating source of the 400Hz
                        (high=ships 400Hz ref) pp119

TERMINAL
  Name                  400HZ-DISC
  Direction             OUTPUT
  Comment               400Hz discrete. T:+0.25vdc (pwr removed)  F:+5vdc
                        (ship 400Hz ref OK)

TERMINAL
  Name                  400HZ-0DEG
  Direction             OUTPUT
  Comment               400Hz-26vac at 0 deg shift. 400Hz ref for platform
                        torquing circuits

SIMULATE
  Terminal              400HZ-2VAC
  From Terminals        NEG12V-IN, POS12V-IN, 400-26VAC, POS6V-IN,
                        NEG24V-IN, POS24V-IN
  Perform               TRUTH TABLE
                        @NEG12V-IN @POS12V-IN @POS24V-IN NEG24V-IN
                        @POS6V-IN @400-26VAC   400HZ-2VAC:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                        'BAD    ??    ??    ??    ??    ??  'BAD
                          ??  'BAD    ??    ??    ??    ??  'BAD
                          ??    ??  'BAD    ??    ??    ??  'BAD
                          ??    ??    ??  'BAD    ??    ??  'BAD
                          ??    ??    ??    ??  'BAD    ??  'BAD
                          ??    ??    ??    ??    ??  'BAD  'ADEQUATE;
```

```
  Comment              400Hz-2.2vac output is good if all pwr is good and
                       26vac-400hz ref good


SIMULATE
  Terminal             FREE-RUN
  From Terminals       NEG12V-IN, POS12V-IN, 400-26VAC, POS6V-IN,
                       POS24V-IN
  Perform              TRUTH TABLE
                       @NEG12V-IN @POS12V-IN @POS6V-IN @POS24V-IN
                       @400-26VAC @free-run:
                       'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'HIGH
                       'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'LOW;
  Comment              free-run is low when the 400hz carrier is not in
                       synch w/ ships 400Hz




SIMULATE
  Terminal             400HZ-0DEG
  From Terminals       NEG12V-IN, POS12V-IN, 400HZ-120, NEG24V-IN,
                       POS24V-IN
  Perform              TRUTH TABLE
                       @NEG12V-IN @POS12V-IN @NEG24V-IN @POS24V-IN
                       @400HZ-120  @400hz-0deg:
                       'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                       'BAD    ??    ??    ??    ??  'BAD
                         ??  'BAD    ??    ??    ??  'BAD
                         ??    ??  'BAD    ??    ??  'BAD
                         ??    ??    ??  'BAD    ??  'BAD
                         ??    ??    ??    ??  'BAD  'BAD;
  Comment              used as 400Hz ref for platform torquing circuits.
                       range >20vRMS <31vRMS

SIMULATE
  Terminal             400HZ-DISC
  From Terminals       NEG12V-IN, POS12V-IN, 400-26VAC, POS6V-IN
  Perform              TRUTH TABLE
                       @NEG12V-IN @POS12V-IN @POS6V-IN @400-26VAC
                       @400HZ-DISC:
                       'GOOD 'GOOD 'GOOD 'GOOD 'high
                       'GOOD 'GOOD 'GOOD 'BAD  'low;
  Comment              low indicates P1-C >31vRMS or <20vRMS. causes
```

```
                    gimbal torquer mtrs disab

INFER
  Terminal          400HZ-120
  From Terminals    400HZ-0DEG, NEG12V-IN, POS12V-IN, NEG24V-IN,
                    POS24V-IN
  Perform           TRUTH TABLE
                    @POS12V-IN @NEG12V-IN @POS24V-IN @NEG24V-IN
                    @400HZ-0DEG   @400HZ-120:
                    'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD
                      ??    ??    ??    ??  'GOOD 'GOOD;
  Comment

INFER
  Terminal          400-26VAC
  From Terminals    NEG12V-IN, POS12V-IN, POS6V-IN, 400HZ-2VAC,
                    NEG24V-IN, POS24V-IN
  Perform           TRUTH TABLE
                    @NEG12V-IN @POS12V-IN @POS6V-IN @NEG24V-IN
                    @POS24V-IN @400HZ-2VAC @400-26VAC:
                    'GOOD 'GOOD 'GOOD 'GOOD 'GOOD  'BAD 'BAD
                      ??    ??    ??    ??    ??   'GOOD 'GOOD;
  Comment
```

B.6 400Hz Power Supply No.1

```
Behavior Definition: 400HZ-PWR-N01
  TERMINAL
    Name                POS24V-IN
    Direction           INPUT
    Comment             +24volt input from the power cube. (+24volt supply
                        2) IMUIC jumper: J69

  TERMINAL
    Name                NEG24V-IN
    Direction           INPUT
    Comment             -24vdc input from the power cube. IMUIC jumper:
                        J72

  TERMINAL
    Name                400HZ-14V
    Direction           INPUT
    Comment             (P1-d) 400Hz 14VAC output signal to platform
                        torquing function

  TERMINAL
    Name                NEG12V-IN
    Direction           INPUT
    Comment             -12vdc from the power cube. IMUIC jumper: J59

  TERMINAL
    Name                POS12V-IN
    Direction           INPUT
    Comment             +12vdc input from the power cube. IMUIC jumper:
                        J70

  TERMINAL
    Name                400HZ-2VAC
    Direction           INPUT
    Comment             (P1-J) 400hz-2.2vac input from 400hz pwr supply
                        No.2 @ P1-Y

  TERMINAL
    Name                400HZ-26V
    Direction           OUTPUT
    Comment             (P1-H) 400Hz 26vac at 120 degree phase shift.
```

```
TERMINAL
  Name                  400-26V-IN
  Direction             INPUT
  Comment               (P1-a)  400Hz 26vac @120deg shift from P1-d to C4
                        to T3-B BACK to P1-a

TERMINAL
  Name                  DUMMY
  Direction             OUTPUT
  Comment


SIMULATE
  Terminal              400HZ-26V
  From Terminals        400-26V-IN, 400HZ-2VAC, 400HZ-14V
  Perform               SET @400HZ-26V TO @400HZ-2VAC;
  Comment

SIMULATE
  Terminal              DUMMY
  From Terminals        400HZ-2VAC
  Perform               set @DUMMY to @400HZ-2VAC;
  Comment


INFER
  Terminal              400HZ-2VAC
  From Terminals        DUMMY, 400-26V-IN, 400HZ-26V, 400HZ-14V
  Perform               if @DUMMY = @400HZ-26V
                         then set @400HZ-2VAC to @400HZ-26V;
  Comment

IMPLAUSIBLE
  Terminals             400-26V-IN, 400HZ-14V
  Predicate             if (@400-26v-in = 'TO-LOW and @400HZ-14V = 'GOOD)
                        or (@400-26v-in = 'TO-HIGH and @400HZ-14V = 'GOOD)
                         then implausible symptom;
  Comment
```

B.7 *Triangle Generator and Case Rotation Power Supply*

Behavior Definition: TRIANG-GEN-PWR

TERMINAL

Name	NEG24V-IN
Direction	INPUT
Comment	-24 volt power from the power cube.

TERMINAL

Name	POS24V-IN
Direction	INPUT
Comment	This is +24volt (2) from the power cube.

TERMINAL

Name	80HZREF-IN
Direction	INPUT
Comment	this is a 80Hz reference signal (P1-U) from the frequency std card

TERMINAL

Name	80-REF-SQR
Direction	OUTPUT
Comment	(P1-c) 80Hz reference square wave output signal.

TERMINAL

Name	80-120-SQR
Direction	OUTPUT
Comment	(P1-R) 80Hz square wave at 120 degrees phase shift.

TERMINAL

Name	80-120-IN
Direction	INPUT
Comment	80Hz (120 degrees phase shift) input signal (P1-A)

TERMINAL

Name	80-REF-TRI
Direction	OUTPUT
Comment	(P1-K) 80Hz reference triangle wave output signal.

TERMINAL

Name	400HZ-0DEG

```
  Direction          INPUT
  Comment            400Hz-26vac at 120deg phase shift.

TERMINAL
  Name               POS12V-IN
  Direction          INPUT
  Comment            +12v input signal from the power cube

TERMINAL
  Name               POS28V-IN
  Direction          INPUT
  Comment            +28v input from the NCC's 115vac 400Hz power
                     distribution

TERMINAL
  Name               400HZ-SQR
  Direction          OUTPUT
  Comment            40v p-p 400Hz square wave referenced to P1-V.
                     (P1-Z, P1-Y)

SIMULATE
  Terminal           80-REF-SQR
  From Terminals     POS28V-IN, 80HZREF-IN
  Perform            IF @POS28V-IN = 'GOOD and @80HZREF-IN = 'GOOD
                       then SET @80-ref-sqr to  'GOOD;
                     IF @POS28V-IN = 'BAD or @80HZREF-IN = 'BAD
                       THEN SET @80-ref-sqr to 'bad;
  Comment            the 80Hz reference square wave only requires the
                     80Hz ref and +28v

SIMULATE
  Terminal           80-120-SQR
  From Terminals     POS28V-IN, 80-120-IN
  Perform            TRUTH TABLE
                     @POS28V-IN @80-120-IN  @80-120-SQR:
                     'GOOD 'GOOD 'good
                     'BAD    ??  'bad
                       ??  'BAD  'bad;
  Comment            80Hz/120Deg/Square wave requires the 80Hz/120deg
                     input and +28v power.
```

```
SIMULATE
   Terminal              80-REF-TRI
   From Terminals        POS28V-IN, POS12V-IN, 80HZREF-IN, NEG24V-IN
   Perform               TRUTH TABLE
                         @POS28V-IN @POS12V-IN @NEG24V-IN @80HZREF-IN
                         @80-ref-tri:
                         'GOOD 'GOOD 'GOOD 'GOOD  'good
                         'BAD    ??    ??    ??   'bad
                           ??  'BAD    ??    ??   'bad
                           ??    ??  'BAD    ??   'bad
                           ??    ??    ??  'BAD   'bad;
   Comment               80Hz reference triangle wave only requires the
                         80Hz ref input and pwr.

 SIMULATE
   Terminal              400HZ-SQR
   From Terminals        POS28V-IN, POS12V-IN, 400HZ-0DEG, POS24V-IN,
                         NEG24V-IN
   Perform               TRUTH TABLE
                         @POS24V-IN @NEG24V-IN @POS12V-IN @POS28V-IN
                         @400HZ-0DEG @400HZ-SQR:
                         'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                         'BAD    ??    ??    ??    ??  'BAD
                           ??  'BAD    ??    ??    ??  'BAD
                           ??    ??  'BAD    ??    ??  'BAD
                           ??    ??    ??  'BAD    ??  'BAD
                           ??    ??    ??    ??  'BAD  'BAD   ;
   Comment               400Hz-square wave dependent only on pwr and the
                         26vac-400Hz input P1-a
```

```
INFER
  Terminal              400HZ-0DEG
  From Terminals        400HZ-SQR, POS28V-IN, POS12V-IN, POS24V-IN,
                        NEG24V-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG24V-IN @POS28V-IN @POS24V-IN
                        @400HZ-SQR   @400HZ-0DEG:
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD 'BAD
                          ??    ??    ??    ??  'GOOD 'GOOD;
  Comment

INFER
  Terminal              80HZREF-IN
  From Terminals        80-REF-TRI, 80-REF-SQR
  Perform               if @80-REF-TRI = 'GOOD
                         then set @80HZREF-IN to 'GOOD;
                        if @80-REF-SQR = 'GOOD
                         then set @80HZREF-IN to 'GOOD;
  Comment

INFER
  Terminal              80-120-IN
  From Terminals        80-120-SQR
  Perform               if @80-120-SQR = 'GOOD
                        then set @80-120-IN to 'GOOD;
  Comment
```

```
Behavior Definition: 640HZ-POWER
  TERMINAL
    Name                NEG6V-IN
    Direction           INPUT
    Comment

  TERMINAL
    Name                POS48V-IN
    Direction           INPUT
    Comment

  TERMINAL
    Name                RUN-CTRL
    Direction           INPUT
    Comment             Gyro Run control input, a DC input at P1-H, P1-J

  TERMINAL
    Name                NEG12V-IN
    Direction           INPUT
    Comment             -12v power input

  TERMINAL
    Name                640HZ-REF
    Direction           INPUT
    Comment             640 Hz reference input signal at P1-Y

  TERMINAL
    Name                640HZ-120
    Direction           INPUT
    Comment             640Hz input signal, with 120 degree phase shift at
                        P1-Z

  TERMINAL
    Name                640HZ-90
    Direction           INPUT
    Comment             640Hz input signal with 90 degree phase shift at
                        P1-F

  TERMINAL
    Name                640HZ-210
    Direction           INPUT
    Comment             640Hz input signal with 210 degree phase shift at
```

```
                    P1-C

TERMINAL
  Name              OUT-LM-90
  Direction         OUTPUT
  Comment           P1-L/M output using 640KHz at 90deg from P1-F

TERMINAL
  Name              OUT-ST-0
  Direction         OUTPUT
  Comment           P1-S/T output using 640KHz at 0deg input from P1-Y

TERMINAL
  Name              OUT-CD-120
  Direction         OUTPUT
  Comment           P1-c/d output using 640KHZ at 120deg input from
                    P1-Z

TERMINAL
  Name              OUT-AB-210
  Direction         OUTPUT
  Comment           P1-A/B output using 640KHz at 210 degrees from
                    P1-C

SIMULATE
  Terminal          OUT-ST-0
  From Terminals    640HZ-REF, NEG12V-IN, RUN-CTRL, POS48V-IN,
                    NEG6V-IN
  Perform           TRUTH TABLE
                    @POS48V-IN @NEG6V-IN @NEG12V-IN @640HZ-REF
                    @RUN-CTRL @OUT-ST-0:
                    'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'good
                    'BAD    ??    ??    ??    ??  'bad
                      ??  'BAD    ??    ??    ??  'bad
                      ??    ??  'BAD    ??    ??  'bad
                      ??    ??    ??  'BAD    ??  'bad
                      ??    ??    ??    ??  'BAD  'bad;
  Comment           along with OUT-ST-120 provides a 640Hz square wave
                    for gyro spin motor
```

```
SIMULATE
  Terminal              OUT-CD-120
  From Terminals        640HZ-120, NEG12V-IN, RUN-CTRL, POS48V-IN,
                        NEG6V-IN
  Perform               TRUTH TABLE
                        @POS48V-IN @NEG6V-IN @NEG12V-IN @640hZ-120
                        @RUN-CTRL @OUT-CD-120:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'good
                        'BAD    ??    ??    ??    ??  'bad
                          ??  'BAD    ??    ??    ??  'bad
                          ??    ??  'BAD    ??    ??  'bad
                          ??    ??    ??  'BAD    ??  'bad
                          ??     ??   ??    ??  'BAD  'bad;
  Comment along w/ OUT-ST-0 provides a 640KHZ square wave. 2 voltages 120shift
SIMULATE
  Terminal              OUT-LM-90
  From Terminals        640HZ-90, NEG12V-IN, RUN-CTRL, POS48V-IN, NEG6V-IN
  Perform               TRUTH TABLE
                        @POS48V-IN @NEG12V-IN @NEG6V-IN @640HZ-90
                        @RUN-CTRL @OUT-LM-90:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'good
                        'BAD    ??    ??    ??    ??  'bad
                          ??  'BAD    ??    ??    ??  'bad
                          ??    ??  'BAD    ??    ??  'bad
                          ??    ??    ??  'BAD    ??  'bad
                          ??    ??    ??    ??  'BAD 'bad;
  Comment               along w/ OUT-AB-210 provides a 640HZ square wave
                        for gyro spin motor

SIMULATE
  Terminal              OUT-AB-210
  From Terminals        640HZ-210, NEG12V-IN, RUN-CTRL, POS48V-IN,
                        NEG6V-IN
  Perform               TRUTH TABLE
                        @POS48V-IN @NEG12V-IN @NEG6V-IN @640HZ-210
                        @RUN-CTRL @OUT-AB-210:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'good
                        'BAD    ??    ??    ??    ??  'bad
                          ??  'BAD    ??    ??    ??  'bad
                          ??    ??  'BAD    ??    ??  'bad
                          ??    ??    ??  'BAD    ??  'bad
                          ??    ??    ??    ??  'BAD  'bad;
```

Comment	along w/ OUT-LM-90 provides a 640Hz square wave for gyro spin motor

B.9 DC Amplifier

Behavior Definition: DC-AMPLIFIER

TERMINAL

Name	GYRO-ERROR
Direction	INPUT
Comment	(P1-Z)Gyro Speed Error. Pulse width modulated 6720Hz: 50%duty cycle OK

TERMINAL

Name	GYRO-RUN
Direction	INPUT
Comment	(P1-E) gyro run command. A high indicates gyro-run

TERMINAL

Name	GYRO-START
Direction	INPUT
Comment	(P1-M) a high indicates the initial 12-15 seconds of gyro start phase.

TERMINAL

Name	BRIDGE-RET
Direction	INPUT
Comment	Bridge return from 640Hz pwr analogous to the pwr supplied to gyro mtr

TERMINAL

Name	POS48V-IN
Direction	INPUT
Comment	(P1-b) +48v from the power cube

TERMINAL

Name	28V/42V-IN
Direction	INPUT
Comment	1st 15sec of gyro start this is +42v, it then drops to +28v. pp23 T.O.

TERMINAL

Name	POS28V-IN
Direction	INPUT
Comment	(P1-c/d) +28v from the NCC 115vac 400Hz pwr distribution

```
TERMINAL
  Name                POS12V-IN
  Direction           INPUT
  Comment             (P1-R) from the power cube

TERMINAL
  Name                NEG12V-IN
  Direction           INPUT
  Comment             (P1-T/U) from the power cube: IMUIC Test pt:J3-J

TERMINAL
  Name                POS6V-IN
  Direction           INPUT
  Comment             +6v from power cube . Test w/ volt meter on NCC

TERMINAL
  Name                GYRO-SPEED
  Direction           OUTPUT
  Comment             (P1-H) discrete indicating correct gyro speed.
                      low: duty cycle 15-85%

TERMINAL
  Name                RUN-CTRL
  Direction           OUTPUT
  Comment             (P1-D) D.C. voltage used to develop AC voltage for
                      the gyro spin motor

SIMULATE
  Terminal            GYRO-SPEED
  From Terminals      POS6V-IN, POS12V-IN, GYRO-ERROR
  Perform             TRUTH TABLE
                      @POS6V-IN @POS12V-IN @GYRO-ERROR  @GYRO-SPEED:
                      'GOOD      'GOOD       'PROPER     'low
                      'GOOD      'GOOD       'IMPROPER   'high;
  Comment             gyro speed is HIGH whenever gyro speed is NOT
                      14,400RPMs
```

```
SIMULATE
  Terminal              RUN-CTRL
  From Terminals        NEG12V-IN, POS12V-IN, POS28V-IN, 28V/42V-IN,
                        POS48V-IN, BRIDGE-RET, GYRO-START, GYRO-RUN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG12V-IN @POS28V-IN @28V/42V-IN
                        @POS48V-IN @GYRO-RUN @GYRO-START @BRIDGE-RET
                        @run-ctrl:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'HIGH 'LOW 'GOOD
                        '28v
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'LOW  'HIGH 'GOOD
                        '42V;
  Comment               omitted wire from C-GAT-2 to Q-SW-1.
```

B.10 Displacement Gyroscope

```
Behavior Definition: GYRO
  TERMINAL
    Name                  4-8KHZ-IN
    Direction             INPUT
    Comment               pickoff excitation fed to the gyro spherical
                          bearing through FUZZbutton

  TERMINAL
    Name                  GYRO-OUT
    Direction             OUTPUT
    Comment               general gyro output, unclear what this signal
                          looks like!

  TERMINAL
    Name                  80-REF-IN
    Direction             INPUT
    Comment               in conjunction with 80-120-in provides case
                          rotation motor power.SQUARE

  TERMINAL
    Name                  80-120-IN
    Direction             INPUT
    Comment               along with 80-ref-in provides 80Hz square wave for
                          case rotation motor

  TERMINAL
    Name                  POS12V-IN
    Direction             INPUT
    Comment

  TERMINAL
    Name                  NEG12V-IN
    Direction             INPUT
    Comment

  TERMINAL
    Name                  MTR-PWR-A
    Direction             INPUT
    Comment               Spin motor power from 640Hz power supply

  TERMINAL
    Name                  MTR-PWR-B
```

```
  Direction             INPUT
  Comment               Spin motor power from the 640Hz power supply

SIMULATE
  Terminal              GYRO-OUT
  From Terminals        NEG12V-IN, POS12V-IN, 80-120-IN, 80-REF-IN,
                        4-8KHZ-IN
  Perform               TRUTH TABLE
                        @NEG12V-IN @POS12V-IN @80-120-IN @80-REF-IN
                         @4-8KHZ-IN @GYRO-OUT:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                        'BAD    ??    ??    ??    ??  'BAD
                          ??  'BAD    ??    ??    ??  'BAD
                          ??    ??  'BAD    ??    ??  'BAD
                          ??    ??    ??  'BAD    ??  'BAD
                          ??    ??    ??    ??  'BAD  'BAD;
  Comment

INFER
  Terminal              4-8KHZ-IN
  From Terminals        NEG12V-IN, POS12V-IN, 80-120-IN, 80-REF-IN,
                        GYRO-OUT
  Perform               TRUTH TABLE
                        @NEG12V-IN @POS12V-IN @80-120-IN @80-REF-IN
                        @GYRO-OUT @4-8KHZ-IN:
                          ??    ??    ??    ??  'GOOD 'GOOD
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD;
  Comment

INFER
  Terminal              80-REF-IN
  From Terminals        NEG12V-IN, POS12V-IN, 80-120-IN, GYRO-OUT,
                        4-8KHZ-IN
  Perform               TRUTH TABLE
                        @NEG12V-IN @POS12V-IN @80-120-IN @4-8KHZ-IN
                        @GYRO-OUT @80-REF-IN:
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD
                          ??    ??    ??    ??  'GOOD 'GOOD;
  Comment
```

```
INFER
  Terminal              80-120-IN
  From Terminals        NEG12V-IN, POS12V-IN, 80-REF-IN, GYRO-OUT,
                        4-8KHZ-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @NEG12V-IN @80-REF-IN @4-8KHZ-IN
                        @GYRO-OUT @80-120-IN:
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD
                          ??    ??    ??    ??  'GOOD 'GOOD;
  Comment

INFER
  Terminal              POS12V-IN
  From Terminals        NEG12V-IN, 80-120-IN, 80-REF-IN, GYRO-OUT,
                        4-8KHZ-IN
  Perform               TRUTH TABLE
                        @NEG12V-IN @80-120-IN @80-REF-IN @4-8KHZ-IN
                        @GYRO-OUT @POS12V-IN:
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD
                          ??    ??    ??    ??  'GOOD 'GOOD;
  Comment

INFER
  Terminal              NEG12V-IN
  From Terminals        POS12V-IN, 80-120-IN, 80-REF-IN, GYRO-OUT,
                        4-8KHZ-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @80-REF-IN @80-120-IN @4-8KHZ-1N
                        @GYRO-OUT @NEG12V-IN:
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD
                          ??    ??    ??    ??  'GOOD 'GOOD;
  Comment
```

B.11 Dependency behavior

```
Behavior Definition: DEPENDENCY
  TERMINAL
    Name                    OUT
    Direction               OUTPUT
    Comment

  TERMINAL
    Name                    DEP-OUT
    Direction               OUTPUT
    Comment

  TERMINAL
    Name                    IN
    Direction               INPUT
    Comment

  SIMULATE
    Terminal                OUT
    From Terminals          IN
    Perform                 set @OUT to @IN;
    Comment

  SIMULATE
    Terminal                DEP-OUT
    From Terminals          PRESET
    Perform                 set @DEP-OUT to 'GOOD;
    Comment

  INFER
    Terminal                IN
    From Terminals          OUT
    Perform                 set @IN  to  @OUT;
    Comment
```

B.12 Transformer

```
Behavior Definition: TRANSFORMER
  TERMINAL
    Name                T-IN
    Direction           INPUT
    Comment

  TERMINAL
    Name                T-OUT
    Direction           OUTPUT
    Comment

  SIMULATE
    Terminal            T-OUT
    From Terminals      T-IN
    Perform             SET @T-OUT TO @T-IN;
    Comment

  INFER
    Terminal            T-IN
    From Terminals      T-OUT
    Perform             set @T-IN to @T-OUT;
    Comment
```

B.13 Synchro Buffer Amplifier

Behavior Definition: SYNCH-BUFF-AMP

TERMINAL	
Name	POS12V-IN
Direction	INPUT
Comment	
TERMINAL	
Name	NEG12V-IN
Direction	INPUT
Comment	
TERMINAL	
Name	POS24V-IN
Direction	INPUT
Comment	
TERMINAL	
Name	NEG24V-IN
Direction	INPUT
Comment	
TERMINAL	
Name	9-6-SQR-IN
Direction	INPUT
Comment	(P1-U)
TERMINAL	
Name	9-6KHZ-TRI
Direction	OUTPUT
Comment	(P1-T)

```
SIMULATE
  Terminal              9-6KHZ-TRI
  From Terminals        9-6-SQR-IN, NEG24V-IN, POS24V-IN, NEG12V-IN,
                        POS12V-IN
  Perform               TRUTH TABLE
                        @NEG24V-IN @POS24V-IN @NEG12V-IN @POS12V-IN
                        @9-6-SQR-IN @9-6KHZ-TRI:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                        'BAD    ??    ??    ??    ??  'BAD
                          ??  'BAD    ??    ??    ??  'BAD
                          ??    ??  'BAD    ??    ??  'BAD
                          ??    ??    ??  'BAD    ??  'BAD
                          ??    ??    ??    ??  'BAD  'BAD;
  Comment

INFER
  Terminal              9-6-SQR-IN
  From Terminals        9-6KHZ-TRI, NEG24V-IN, POS24V-IN, NEG12V-IN,
                        POS12V-IN
  Perform               TRUTH TABLE
                        @9-6KHZ-TRI @NEG24V-IN @POS24V-IN
                         @NEG12V-IN @POS12V-IN  @9-6-SQR-IN:
                        'GOOD   ??    ??    ??    ??  'GOOD
                        'BAD  'GOOD 'GOOD 'GOOD 'GOOD 'BAD;
  Comment
```

B.14 Velocity Meter

```
Behavior Definition: VELOCITY-METER
  TERMINAL
    Name                4-8KHZ-IN
    Direction           INPUT
    Comment             100vac 4.8Khz signal used for velocity meter servo
                        excitation

  TERMINAL
    Name                POS28V-IN
    Direction           INPUT
    Comment             +28v input from the NCC 155vac 400Hz power
                        distribution

  TERMINAL
    Name                POS24V-IN
    Direction           INPUT
    Comment             +24v No.1 input via SLIP RING

  TERMINAL
    Name                POS12V-IN
    Direction           INPUT
    Comment             +12v input from the power cube via a SLIP RING

  TERMINAL
    Name                NEG12V-IN
    Direction           INPUT
    Comment             -12v input from the power cube via a SLIP RING

  TERMINAL
    Name                VM-OUT
    Direction           OUTPUT
    Comment             Velocity Meter output
```

```
SIMULATE
  Terminal              VM-OUT
  From Terminals        NEG12V-IN, POS12V-IN, POS24V-IN, POS28V-IN,
                        4-8KHZ-IN
  Perform               TRUTH TABLE
                        @NEG12V-IN @POS12V-IN @POS24V-IN @POS28V-IN
                        @4-8KHZ-IN @VM-OUT:
                        'GOOD 'GOOD 'GOOD 'GOOD 'GOOD 'GOOD
                        'BAD    ??    ??    ??    ??  'BAD
                          ??  'BAD    ??    ??    ??  'BAD
                          ??    ??  'BAD    ??    ??  'BAD
                          ??    ??    ??  'BAD    ??  'BAD
                          ??    ??    ??    ??  'BAD  'BAD;
  Comment

INFER
  Terminal              4-8KHZ-IN
  From Terminals        VM-OUT, NEG12V-IN, POS12V-IN, POS24V-IN, POS28V-IN
  Perform               TRUTH TABLE
                        @NEG12V-IN @POS12V-IN @POS24V-IN @POS28V-IN
                         @VM-OUT @4-8KHZ-IN:
                          ??    ??    ??    ??  'GOOD 'GOOD
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD;
  Comment

INFER
  Terminal              POS28V-IN
  From Terminals        VM-OUT, NEG12V-IN, POS12V-IN, POS24V-IN, 4-8KHZ-IN
  Perform               TRUTH TABLE
                        @NEG12V-IN @POS12V-IN @POS24V-IN @4-8KHZ-IN
                        @VM-OUT @POS28V-IN:
                          ??    ??    ??    ??  'GOOD 'GOOD
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD;
  Comment

INFER
  Terminal              POS24V-IN
  From Terminals        VM-OUT, NEG12V-IN, POS12V-IN, POS28V-IN, 4-8KHZ-IN
  Perform               TRUTH TABLE
                        @NEG12V-IN @POS12V-IN @POS28V-IN @4-8KHZ-IN
                        @VM-OUT @POS24V-IN:
                          ??    ??    ??    ??  'GOOD 'GOOD
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD 'BAD;
```

```
  Comment

INFER
  Terminal              POS12V-IN
  From Terminals        VM-OUT, NEG12V-IN, POS24V-IN, POS28V-IN, 4-8KHZ-IN
  Perform               TRUTH TABLE
                        @NEG12V-IN @POS24V-IN @POS28V-IN @4-8KHZ-IN
                        @Y-VM-OUT @POS12V-IN:
                          ??    ??    ??    ??  'GOOD 'GOOD
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD;
  Comment

INFER
  Terminal              NEG12V-IN
  From Terminals        VM-OUT, POS12V-IN, POS24V-IN, POS28V-IN, 4-8KHZ-IN
  Perform               TRUTH TABLE
                        @POS12V-IN @POS24V-IN @POS28V-IN @4-8KHZ-IN
                        @VM-OUT @NEG12V-IN:
                          ??    ??    ??    ??  'GOOD 'GOOD
                        'GOOD 'GOOD 'GOOD 'GOOD 'BAD  'BAD;
  Comment
```

B.15 Slip Ring

```
Behavior Definition: SLIP-RING
  TERMINAL
    Name                  SR-IN
    Direction             INPUT
    Comment

  TERMINAL
    Name                  SR-OUT
    Direction             OUTPUT
    Comment

  SIMULATE
    Terminal              SR-OUT
    From Terminals        SR-IN
    Perform               set @SR-OUT to @SR-IN;
    Comment

  INFER
    Terminal              SR-IN
    From Terminals        SR-OUT
    Perform               set @SR-IN to @SR-OUT;
    Comment
```

B.16 Resistor

```
Behavior Definition: RESISTOR
  TERMINAL
    Name                  R-IN
    Direction             INPUT
    Comment

  TERMINAL
    Name                  R-OUT
    Direction             OUTPUT
    Comment

  SIMULATE
    Terminal              R-OUT
    From Terminals        R-IN
    Perform               set @R-OUT to @R-IN;
    Comment

  INFER
    Terminal              R-IN
    From Terminals        R-OUT
    Perform               set @R-IN to @R-OUT;
    Comment
```

Bibliography

1. Abelson, Harold and Sussman, Gerald Jay. *Structure and Interpretation of Computer Programs*. McGraw-Hill Book Company, New York, 1985.
2. Aikins, J.S. Prototypical knowledge for expert systems. *Artificial Intelligence*, 20:163–210, June 1983.
3. Blasdel, Arthur N., Senior Research and Development Engineer. Automated Fault handling of a satellite electrical power subsystem using a model-based expert system.*Personnel Correspondence*, Jan 1989.
4. Chandrasekaran B. An approach to medical diagnosis based on conceptual structures. *Journal of Optimization Theory and Application*, 5:134–142,20–23, August 1979.
5. Davis, Randall. "Diagnosis Via Causal Reasoning: Paths of Interaction and the Locality Principle,". *Artificial Intelligence in Maintenance*, 102-122. Park Ridge NJ: Noyes Publications, 1985.
6. Tong, David, et.al. "Diagnosing an Analog Feedback System using Model-Based Reasoning.". *IEEE*, CH2715-1/89, 290-295.
7. DeKleer,J. , and Williams, B.C., "Diagnosing Mulitple Faults,". *Artificial Intelligence in Maintenance*, 32(1987) 97-130.
8. Davis, Randall. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, April 1984.
9. Davis, Randall and Hamscher, Walter. Model-based reasoning:troubleshooting *Exploring Artificial Intelligence: Survey Talks from the National Conference on Artificial Intelligence*. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.
10. de Kleer, J. and Williams, B.C., Diagnosing multiple faults, *Artificial Intelligence* 28:127-162, 1986.
11. Department of the Navy. *Organizational level technical manual for inertial navigation system AN/WSN-1(V)2*. NAVSHIPS 0967-529-6010. November, 1973.
12. Department of the Navy. *Depot level technical manual for inertial navigation system AN/WSN-1(V)2*. NAVSHIPS 0967-529-7010. November, 1973.
13. Merriam-Webster dictionaries. *Websters Ninth New Collegiate Dictionary*. Merriam-Webster Inc., Springfield, Massachusetts, U.S.A., 1983.
14. Dyson, Esther. Repairman in a box, commentary. *FORBES*, page 306, January 1989.
15. Hamscher, W.C. *Model-Based Troubleshooting of Digital Systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1988.
16. Hanson, Eric. Class lecture in EENG749, Advanced Topics in Artificial Intelligence, Air Force Institute of Technology (AU), Wright-Patterson AFB OH., May 1989.
17. Harmon, Paul and King, David. *Artificial Intelligence in Business: EXPERT SYSTEMS*. John Wiley and Sons, INC., New York, 1985.

18. Heiser, J.E. Progress report: a computerized psychopharmacology advisor. *Proceedings of the Eleventh Colloquium International Neuro-Psychopharmacologicum, Vienna Austria.*, pages 210–237, 1978.

19. Intelligent diagnostic expert assistant (IDEA) *Development Reference manual*, AI Squared Inc., 1988.

20. Newquist III, Harvey P. Expert systems invade mainstream. *AI Expert*, 4:71–73, April 1989.

21. Merritt, Bonnie. Anatomy of a diagnostic system. *AI Expert*, 2:52–63, May 1987.

22. Papenhaussen, David W. and Hadley, Brent L., "CEPS - an artificial intelligence approach to avionics maintenance." *Proceedings of the Air Force Workshop on Artificial Intelligence Applications for Integrated Diagnostics*, pages 208–217, 1987 (AFWAL-TR-87-1101)

23. Pipitone, Frank. The FIS electronics troubleshooting system. *COMPUTER*, 19:68–76, July 1986.

24. Rasmussen, Steven J. System Engineer for DMINS Testing. *Personal interviews.* AGMC, Newark AFB OH., April 1988.

25. Shortliffe, E.H. An artificial intelligence program to advise physicians regarding antimicrobial therapy. *Computers and Biomedical research*, 6:544–560, May 1973.

26. Skinner, James M., Capt USAF. A diagnostic system blending deep and shallow reasoning., Master's Thesis, ENG, WPAFB, December 1988.

27. Waterman, Donald A. *A Guide to Expert Systems*, Addison-Wesley, Reading, Massachusetts, 1985.

Vita

Raymond E. Yost was [redacted] in 1982 received an Associates Degree in Aeronautical Engineering from Daniel Webster College (formerly the New England Aeronautical Institute), Nashua New Hampshire. In 1984, he received a Bachelor of Science in Mathematics, with a minor in Computer Science, from the State University of New York at Cortland. That same year, he was commisioned in the USAF and was assigned to the Air Force Operational Test Center (AFOTEC), Edwards AFB, CA. as a software engineer on the B-1 bomber. In 1988 he was selected to enter the Air Force Institute of Technology (AFIT) to pursue graduate work in Computer Systems while specializing in artificial intelligence.

[redacted]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/89D-19	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology (AU) Wright-Patterson AFB, Ohio 45433-6533	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Application of MODEL_BASED REASONING to Diagnosis of Faults in Inertial Navigation Equipment

12. PERSONAL AUTHOR(S)
Raymond E. Yost, A.S., B.S., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1989 December	135

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Expert Systems, Model-based diagnosis, Model-based reasoning
12	5		
12	9		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: Lt Col Charles Bisbee

(see reverse)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Lt Col Charles Bisbee	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.*

Abstract

In 1988 Skinner produced the Blended Diagnostic System (BDS) which was an attempt to provide the Aerospace Guidance and Metrology Center (AGMC) with an expert system capable of diagnosing faults in the Dual Miniature Inertial Navigation System (DMINS). Skinner proposed the blending of a traditional rule-based system with a model-based system. The techniques used to perform the model-based reasoning in BDS are however primitive compared to other techniques currently available. This thesis describes the development of a model-based diagnostic system using techniques pioneered by Randall Davis, and substantially more sophisticated than those used in BDS. A diagnostic prototype for the DMINS was developed which provides a more thorough and consistent diagnosis than does Skinner's model-based system.

The models in our prototype were created using the Intelligent Diagnostic Expert Assistant (IDEA) software developed by AI Squared Inc., of North Chelmsford MA. IDEA is based on extensive research by Davis at the Massachusetts Institute of Technology (MIT).