NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

EXPERT SYSTEMS DEVELOPMENT
UTILIZING
HEURISTIC METHODS

by

John N. Lewis

June 1996

Thesis Advisor: Hemant Bhargava

Approved for public release; distribution is unlimited.

19961021 170

DTIC QUALITY INSPECTED 1

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE: Expert Systems Development Utilizing Heuristic Methods	5. FUNDING NUMBERS
6. AUTHOR(S) LT John N. Lewis, USNR	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis analyzes the diagnostic domain and isolates the heuristics employed by experts to arrive at diagnostic solutions. These heuristic methods are then generalized in order to arrive at a series of heuristic rules that can be applied to a wide range of diagnostic processes independent of there respective domain. To test the validity of the generalized heuristics, a prototype expert system was created targeting the heuristics employed by avionics repair technicians in repair of the APS-115 radar system on the P-3C Orion.

14. SUBJECT TERMS Expert System, Prototype, Knowledge Acquisition, Heuristic Development, Heuristic, P-3C Orion, Avionics Maintenance, Diagnostics, Troubleshooting			15. NUMBER OF PAGES 82 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

EXPERT SYSTEMS DEVELOPMENT UTILIZING HEURISTIC METHODS

John N. Lewis
Lieutenant, United States Navy Reserve
B.A., California State University, Sacramento, 1984

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL

June 1996

Author: _____
John N. Lewis

Approved by: _____
Hemant Bhargava, Thesis Advisor

_____
Donald R. Eaton, Associate Advisor

_____
Ruben T. Harris, Chairman, Department of Systems Management

ABSTRACT

This thesis analyzes the diagnostic domain and isolates the heuristics employed by experts to arrive at diagnostic solutions. These heuristic methods are then generalized in order to arrive at a series of heuristic rules that can be applied to a wide range of diagnostic processes independent of there respective domain. To test the validity of the generalized heuristics, a prototype expert system was created targeting the heuristics employed by avionics repair technicians in repair of the APS-115 radar system on the P-3C Orion.

TABLE OF CONTENTS

I. INTRODUCTION 1
- A. BACKGROUND 1
- B. OBJECTIVES 1
- C. THE RESEARCH QUESTIONS 1
- D. SCOPE 2
- E. METHODOLOGY 2
- F. THESIS ORGANIZATION 3

II. BACKGROUND 5
- A. THE AVIATION MAINTENANCE SYSTEM 5
  - 1. Levels of maintenance. 5
    - a. Organizational Level Maintenance 5
    - b. Intermediate Level Maintenance 6
    - c. Depot Level Maintenance 7
    - d. Other Technical Support 7
  - 2. Maintenance Scenario 8
  - 3. The A-799 Rate. 9
- B. EXPERT SYSTEM 9
- C. DOMAIN SELECTION 11
- D. THE APS 115 RADAR SYSTEM 12

III. EXPERT SYSTEM SOLUTIONS 13
- A. DIAGNOSTICS 13
- B. EXPERT SYSTEMS IMPLEMENTATIONS 13
  - 1. GTE COMPASS System 14
  - 2. MK-92 Maintenance Advisor System 15
  - 3. The Interactive Fault Diagnosis System 16
  - 4. Automotive Engine Troubleshooter 17
  - 5. MYCIN 17
- C. SEPARATING DIAGNOSTICS FROM THE DOMAIN. 18

IV. HEURISTIC DEVELOPMENT FOR DIAGNOSTIC SYSTEMS 21
- A. HEURISTIC DEFINITION 21
  - 1. Heuristic Example 21
  - 2. Benefits Of Generalization 22
  - 3. Heuristic Representation 23
- B. HEURISTIC GENERATION 23
  - 1. Abduction 24
    - a. Pseudo-code Representation 25
  - 2. Commonality 25
    - a. Pseudo-code Representation 26

3. Cross Commonality 27
a. Pseudo-code Representation 27
4. Complexity 28
a. Pseudo-code Representation 28
C. PROTOTYPE KNOWLEDGE ACQUISITION 29
1. Initial Steps 30
2. The Expert 31
D. IMPLEMENTATION 31
E. PROTOTYPE DESIGN 32

V. TEST AND VALIDATION 33
A. PROTOTYPE TESTING 33
1. Initial Prototype Improvements 33
2. Subsequent Improvements 34
3. Knowledge Acquisition Issues 35
B. HEURISTIC TESTING 35
1. Abduction 36
2. Commonality 36
3. Cross-Commonality 37
4. Complexity 37
C. FINAL TEST 38

VI. LESSONS LEARNED 39
A. KNOWLEDGE ACQUISITION 39
B. KNOWLEDGE REPRESENTATION 40
C. IMPLEMENTATION 40
D. CLIPS INTERFACE 41
E. CONCLUSION 41

APPENDIX A. PROTOTYPE SYSTEM CODE 43

APPENDIX B. APS 115 RADAR SYTEM 65

LIST OF REFERENCES 71

INITIAL DISTRIBUTION LIST 73

I. INTRODUCTION

A. BACKGROUND

The era of shrinking budgets has had enormous impact on the Department of Defense, a significant facet of which has been personnel losses in experienced senior military maintenance technicians as well as their civilian counterparts. Highly trained technicians have exited the service in droves. This has created a twofold problem within DOD; an eroding experience base coupled with the requirement to "do more with less." This problem is further exacerbated when older platforms are factored in, systems which require highly experienced technical support. With few acquisitions of modern weapons platforms planned, the gap in maintenance experience will increase. By applying expert systems to capture some of the technical expertise before it evaporates, the gap between the departing experience base and the remaining technicians can be narrowed.

B. OBJECTIVES

This thesis describes the design and prototyping of an avionics' maintenance expert system for the P-3C Orion aircraft. It addresses the initial development aspects of the life cycle of an expert system, emphasizing the use of heuristics to capture generalized diagnostic procedures in order to insure modularity and future growth. Knowledge acquisition, representation, and prototype implementation issues will also be addressed.

C. THE RESEARCH QUESTIONS

1. How can expert systems best be applied to avionics diagnostic processes?
2. How can heuristics be used to model diagnostic processes?

3. Can generalized heuristics be extended to encompass other diagnostic domains?

D. SCOPE

This thesis will develop diagnostic heuristic modules using an Expert System shell. The modules will then be applied to actual diagnostic processes via a prototype diagnostic advisor expert system. The prototype expert system will be used to evaluate failure indications in the P-3C APS 115 Radar, and will then make recommendations for further action in order to assist in accurate repair.

Emphasizing the design of heuristic modules to capture diagnostic processes will result in a modular approach useful for designing add-ons, thus aiding replication and expansion of the prototype to incorporate additional avionics systems.

E. METHODOLOGY

Development of the heuristic models will rely on observation of diagnostic processes, interviews with technicians, and the authors experience in the domain. Once identified, diagnostic processes will be written using predicate logic and then converted into expert system modules using the expert system shell developed by NASA.

Development of the Avionics Diagnostic Advisor Expert System prototype will follow the approach used by Chorafas (1990, pp. 21) which consists of four phases: problem definition, knowledge acquisition, programing, and test and maintenance.

The goal of the acquisition phase should be to develop a prototype and will build off of the experience gained in developing the heuristic models. Iteration of phases two

through four will further refine the prototype until a finalized version of the system is achieved. The scope of this thesis precludes completing a finished product, but will complete one iteration of prototype refinement.

F. THESIS ORGANIZATION

The thesis is organized as follows: Chapter II provides the reader with a general background and description of the Aviation Maintenance System, domain selection, and how the P-3C APS 115 Radar was selected for prototype development.

Chapter III describes how other expert systems solutions have been applied to the diagnostic problem. Several troubleshooting domains and their expert systems will be examined, focusing on similarities between the various systems. In addition, it introduces the concept of treating troubleshooting processes as generalized functions that can be applied to a wide variety of domains.

Chapter IV addresses the design of the heuristic models that will be used to capture diagnostic processes. This chapter will also encompass knowledge acquisition, implementation, and prototype design. In the prototype phase the diagnostic heuristics will be incorporated into the prototype.

Chapter V examines how the heuristic models will be tested in a troubleshooting environment. This is where the prototype will be applied to the P-3C APS 115 Radar failure indications. Knowledge gained will then be applied to refining the prototype and further testing.

Chapter VI discusses lessons learned during heuristic development and prototype design. Emphasis is place on insights gained about the developmental process in order to

foster follow on research.

Appendix A contains the CLIPS code for the heuristic modules and the prototype, while Appendix B is an overview of APS-115 radar system failure indications.

II. BACKGROUND

This chapter provides the reader with a general background and description of the Naval Aviation Maintenance Program, some typical avionics maintenance practices, an expert system description, and how an expert system can benefit. A brief description of the APS 115 Radar is also provided.

A. THE AVIATION MAINTENANCE SYSTEM

The Naval Aviation Maintenance Program (NAMP) is described in detail in OPNAVINST 4790.2E. This section is largely excerpted from OPNAVINST 4790.2E and provides a brief overview of the NAMP, paying particular attention to those details that bear directly on avionics maintenance.

1. Levels of maintenance

Maintenance for aircraft is divided into three levels. Organizational, Intermediate, and Depot level. The three levels correspond in general to the degree of repair that can be carried out, with Organizational being the simpler repairs done at the squadron level up to Depot, which performs the most complicate types of aircraft repairs.

a. Organizational Level Maintenance

Organizational level is maintenance is performed by members of the squadron that the aircraft is assigned to. Two types of maintenance are carried out; scheduled and unscheduled. Scheduled maintenance consists of inspections, phase maintenance, and life cycle support for equipment. Unscheduled maintenance takes place

when some equipment fails, for example a UHF radio fails to transmit and requires corrective action.

In the event of an equipment failure, a Maintenance Action Form (MAF) is generated. A MAF describes the nature of the malfunction and contains relevant equipment data to aid in repair. A technician will use the MAF as a starting point for corrective action. This is typically where the troubleshooting process begins, and any subsequent repair work on that equipment is annotated on the MAF. The majority of troubleshooting at the"O" level is relegated to isolating a fault to a particular component and then replacing the component, essentially parts-swapping good for bad. Components that cannot be repaired in the squadrons shop are sent up to the next level of maintenance, Aircraft Intermediate Maintenance Department (AIMD).

b. Intermediate Level Maintenance

Intermediate level maintenance (I-level) is where more detailed maintenance occurs, particularly in regards to electronic repairs. OPNAVINST 4790.2E states that all repair on microcircuits will take place at the I-level if the facility is suitably certified. A squadron may have technicians capable of this type of repair, but they are usually assigned to the AIMD on a semi-permanent basis. The technicians at I-level have a higher degree of training than those at O-level, as well as access to more sophisticated test equipment. The aviation maintenance philosophy dictates that when more difficult repairs are needed, they will go up the chain from organizational to intermediate and depot level if necessary. This implies that not every squadron can be supplied with a comprehensive test suite or have every technician extensively trained in all phases of

repair. Economics is an additional factor as supplying all levels of maintenance comprehensive test suites and personnel would be prohibitively expensive. If there are funds for only a few specialists it makes sense to have them centrally located where they can serve more customers, however this helps to insure troubleshooting at the O-level will likely remain on a part-swapping basis.

c. Depot Level Maintenance

The final level of aviation maintenance is Depot level. This is where the most difficult and technically challenging aircraft repairs are made. The personnel working at Depot level are typically highly experienced technicians, many of whom are retired from the military and continue to work on the aircraft that they maintained during their service careers. This is a significant experience base, one that is rapidly eroding due to budget cuts.

For naval aircraft there are five depot level maintenance facilities which serve the fleet which may not be co-located with the type of aircraft they serve. Aircraft are sent to the depot from their respective locations worldwide. This results in an inverted pyramid in which the most experienced technicians are not located near their customers. The concentration of experience at the depot level, those that would likely be called "experts", are the farthest away from the customers that need their services.

d. Other Technical Support

Additionally, there are organizations that can lend technical expertise to any of the three levels. Naval Aviation Engineering Support Units (NAESU) are funded by NAVAIR and can be utilized down to O-level when specialized assistance is required.

Manufacturers also support technical representatives (tech-reps), usually as part of a maintenance contract which can also be used at the O-level. Both NAESU and manufacturers tech-reps are typically highly experienced technicians and engineers, and may be either former miliary or civilian with commensurate experience levels. Use of these assets is also a function of budget restrictions, however.

2. Maintenance Scenario

An unscheduled maintenance action is initiated by filling out a MAF. For example, if an APS 115 radar fails in-flight and cant be fixed by the in-flight technician (IFT), the IFT will fill out a MAF listing the type of failure and description of the problem. After completion of the flight, that MAF and any others generated during the flight will be sent to maintenance control. The APS 115 MAF will be assigned to the avionics shop for and repair. An avionics technician will then troubleshoot the system using the previously mentioned manuals. (OPNAVINST 4790.2E)

Troubleshooting begins at the most simple level by checking a series of fault indicators and *go/no go* lights which are listed in a particular series in the troubleshooting diagram. The technician will initially attempt to duplicate the initial fault indications as written up in the MAF. The APS 115 radar has a built in test module which utilizes a series of rotary switches to select different components to test and a dial indicator to tell if that particular component is good or bad. If the indications on the test module are all positive, the technician will next use an ohm-meter to check the outputs at various test points in the system. Based on these indications, the fault may be tracked down to a particular component, for example a printed circuit card. If so, this card is removed and

replaced with a card that is known to be good.

The card will then be given to the squadron supply office to be forwarded to an Aviation Maintenance Screening Unit (AMSU). At the AMSU the card will be screened to determine where it will be sent for repair. The fault indications will be matched against the capabilities of the repair depots and the card will be forwarded to that facility which has the ability to effect the repair. This means the card may be sent across the country or across the street, depending on the location of the depot. Once at the depot, the card will be repaired if possible and returned to the supply system for reuse. If the fault indications cannot be replicated, and the card is otherwise determined to be working, it will also be returned to supply system. This opens up the possibility that a component may be returned to the system that still is not repaired correctly.

3. The A-799 Rate

An A-799 is a code added to a MAF for two possible reasons. Either a maintenance action cannot duplicate the failure or the cause of the failure cannot be determined. In both cases if the suspect component now functions correctly it is placed back in service and can be re- used for repair purposes again. The rate of A-799's are closely tracked, and a high rate may indicate a problem with the particular parts, with the aircraft, or with the maintenance actions involved.

B. EXPERT SYSTEM

Definition:

> An expert system permits the knowledge and experience of one or more experts to be captured and stored in a computer. This knowledge can then be used by anyone requiring it. The purpose of an expert system is not to replace the experts,

> but simply make their knowledge and experience more widely available. The expert system permits others to increase their productivity, improve the quality of their decisions, or simply to solve problems when an expert is not available. (Frenzel, 1987, pp.9-10)

This definition is particularly applicable to the realm of Naval Aviation maintenance. It can be derived from this definition that expert systems can also be used to capture knowledge when that knowledge base is shrinking, as is the case in aviation maintenance.

1. Expert Systems Benefits

Diagnostic expert systems can make maintenance actions, particularly those at the lowest levels, more effective. (Sprague and McNurlin, 1995, pp. 462-464) Over an eighteen month period from November 1993 until March 1995 there were over ten thousand avionics related A-799s recorded in the NALDA database. This represents a significant amount of wasted time and effort. Every MAF created must be cleared before the equipment can be returned to service. This may mean as little as one hour of work or as much as several weeks.

Additional benefits result from use of Expert Systems. Technicians gain insight into difficult diagnostic procedures that increases their knowledge base, effectively raising their training level without the expense of sending them to school. Use of Expert Systems will also free up the time that Intermediate and Depot level technicians are devoting to routine repairs, allowing them to concentrate their time on the more difficult problems. This will result in more effective utilization of existing weapons systems.

C. DOMAIN SELECTION

> "The selection of the appropriate domain for the project may have more to do with the success of the effort than any other decision." (Prerau, 1990, pp. 78-79)

The selection of P-3C avionics as the domain was based on two factors; suitability and experience. Most avionics equipment has Built-in-Test (BIT) capability, that is, a display or readout from the equipment that will report any discrepancies or failures and give the technician a starting point to effect repairs. While the output of all BIT equipment is not standardized, it does provide an workable starting point for integration into an expert system. Personal experience in the P-3C Orion community was the other deciding factor. The author has twelve months of prior experience as a P-3C squadron maintenance officer. Additionally, a P-3C facility is located within driving distance of NPS, making access to experts and technical references convenient. Use of the APS 115 Radar as the basis of the prototype Avionics Diagnostic Advisor was determined by searching the Naval Aviation Logistics Data Analysis database. Using frequency of A-799s as selection criteria, the search yielded the pilots horizontal situation indicator horizontal situation indicator (HSI)) as having the highest reported occurrence of A-799, the autopilot system second, and the radar system having the third highest with 155 A-799s in that period. Review of the associated tech manuals revealed that the troubleshooting procedures for the HSI and the autopilot would be more difficult to translate into an expert system relative to the APS 115 radar, therefore the radar system was chosen as the target system for a prototype.

D. THE APS 115 RADAR SYSTEM

The APS 115 radar system is the primary airborne surveillance device for the P-3C aircraft. The search radar system comprises two separate antennas mounted in the nose and tail of the aircraft, two radar receiver-transmitters, radar control panels, and a video data display. The system is used for detecting surface vessels, submarines operating with a snorkel, aircraft, and for weather avoidance. (NAVAIR 01-75PAC-1.1, pp. 12-267) A more complete system description along with APS 115 radar system failure modes is provided in Appendix B.

III. EXPERT SYSTEM SOLUTIONS

This chapter examines existing expert systems that address diagnostics in varied domains, including medicine, electronics, and automotive mechanics. Many expert systems are targeted at a narrow domain, however many of the diagnostic processes that each system attempts to capture have strong similarities that can be exploited for developing new applications.

A. DIAGNOSTICS

Diagnostics is defined in the Random House dictionary as the *process of identifying the cause or nature of a problem from the symptoms*. This process is used in a wide range of domains and is typically performed by human experts. Automating the diagnostic or troubleshooting process was the goal of many early expert systems. Capturing the mental procedures used by doctors or technicians in performing diagnoses resulted in systems that could supplement their work. It is important to point out that while an expert system can perform diagnostics accurately and consistently, they are by no means meant to replace the expert. (Prerau, 1990, pp. 5)

B. EXPERT SYSTEMS IMPLEMENTATIONS

Expert systems are not strictly limited to troubleshooting or diagnostics domains. Expert systems applications range from areas as diverse as the stock market, income tax, and personnel management to name a few. The focus of this thesis, however is on diagnostics, specifically in the realm of avionics.

Several different existing diagnostic expert systems were examined. The purpose was two-fold; first to gain insight into the methods and processes used to capture and represent expert knowledge, and second, to determine which systems could be integrated into the design of a prototype avionics expert system. The next section describes key points of some of these systems.

1. GTE COMPASS System

In the mid 1980's, GTE initiated an expert systems project which came to be called COMPASS, (Central Office Maintenance Printout Analysis and Suggestion System). The impetus for the system was GTE's upgrading of their network switches for long distance telephone traffic. The upgrade required the installation of electronic switches to replace the analog switches then in use in their long distance switching networks. The upgrade was scheduled over a period of several years, which would require the use of both types of switches concurrently. (Prerau, 1990)

As the new switches were installed, the maintenance force was trained to repair the new equipment, resulting in a shrinking pool of technicians that were skilled in repair of the old type of switches. Under the direction of David Prerau, GTE Laboratories initiated an expert systems project designed to capture the knowledge of those technicians who were most experienced with the old switches.

The system was designed and run on purpose built artificial intelligence symbolic processors utilizing LISP. Initial success with an analog switch maintenance expert system led to development and expansion of the COMPASS project to include expert systems for other electronics applications within GTE.

A significant parallel between GTE and DOD can be drawn. DOD is facing a similar diminishing of experienced technical experts while legacy systems continue to operate, providing the same impetus for use of expert systems within DOD.

2. MK-92 Maintenance Advisor System

This project was undertaken at Naval Postgraduate School and its development has provided the source for several theses. The Mk-92 Maintenance Advisor is an expert system designed to perform electronic troubleshooting on the fire control system of the FFG-7 class of frigate.

Motivation for this system development came from a high number of failures of the fire control system that were beyond the ability of the shipboard personnel to repair. This necessitated the use of civilian technicians to make repairs, frequently having to travel to where the particular ship happened to be in the world. The problems posed by a broken fire control system to a deployed unit are obvious, in addition to the lost resources allocated to funding travel for technicians. (Lewis, 1993)

The Mk-92 maintenance advisor was designed to be run on a personal computer of at least Intel 386 class, making use of an expert system software package called ADEPT. Expert knowledge for inclusion into the system was gained from the same civilian technicians who were most experienced in the repair of the fire control system. A working prototype was developed and has been successfully tested in the field.

Significant in the development of the expert system was the selection of ADEPT as the implementation software. ADEPT is one of the most popular expert system software packages, but its use imposes some limitations. ADEPT requires the use of

formal decision trees as documentation. All of the decision nodes must be logically connected in series. This limits the ability of ADEPT to make use of heuristics, in that frequently a heuristic repair technique will not fit into the flow of a decision tree. (Frenzel, 1990, pp. 61)

3. The Interactive Fault Diagnosis System

This is an expert system designed to perform troubleshooting on the Pratt and Whitney TF-30 engine. Undertaken by the Aeronautical Research Laboratory of the Australian Defense Science and Technology Organization, this system attempted to incorporate the heuristic processes used by some experienced civilian technicians that were better at repairs than their military counterparts. (Forsyth, et al, 1990)

The system was designed to run on a personal computer using the Neuron Data shell Nexpert Object software. The development methodology employed is similar to that used to develop the prototype avionics maintenance for this thesis. The system initially used the diagnostic decision trees from the TF-30 technical manuals entered into the expert system shell. Heuristic methods employed by the technicians were then evaluated for inclusion into the expert system.

A shortcoming in the system was the method used to include the heuristics into the prototype. One of the design criteria for the prototype stipulated that all steps in the repair must be included in a decision tree. This had the effect of merely improving the existing decision trees from the technical manuals. If a heuristic employed could not be included in the decision tree, it was discarded from the final iteration of the expert system, even if it was proven to work. This resulted in an expert system that was well

documented in terms of the decision tree, but was not as accurate as it was when all of the heuristics were used.

4. Automotive Engine Troubleshooter

This system was designed for troubleshooting diesel engines for the public transit system in New Delhi, India. This system used a similar development method to the TF-30 system, employing troubleshooting trees from technical manuals combined with expert knowledge to create the expert system. (Ghandi, et al, 1994)

The paper describing this system makes an important distinction between surface and deep knowledge when included in an expert system. Surface knowledge is that possessed by the expert and may encompass a wide variety of skills and experiences. Deep knowledge is that which is accurate and exhaustive and is rigorously documented. The authors point out that deep knowledge lends itself well to decision trees, however it is generally time consuming and wastes manpower and test equipment when employed. Surface knowledge can be characterized as heuristic methods, which the authors point out may be quicker at achieving a solution than deep knowledge. The difficult part of developing an expert system is how to combine the two types of knowledge.

5. MYCIN

This is one of the earliest implementations of expert systems technology. MYCIN is a medical diagnostic system that was designed to aid a doctor in diagnosing different types of bacterial infections and to recommend the type of drug and dosage needed to overcome the infection. (Frenzel, 1987, pp. 69)

MYCIN is significant in that it performs diagnosis in a manner similar to a

troubleshooting diagram. The system requires a physician to interact with, and the final decision as to diagnosis is left up to the doctor. One of the benefits of MYCIN is that it is intended to aid inexperienced in diagnosing diseases with which they may have had little practical exposure, which is also an intended byproduct of the Avionics Diagnostic Advisor. (Olson and Courtney, 1992, pp. 246)

C. SEPARATING DIAGNOSTICS FROM THE DOMAIN

For this thesis, the words diagnostics and troubleshooting are used synonymously. As was shown in the previous section, diagnostic expert systems have been applied to a wide range of domains. The key element in this thesis is separating and quantifying the thought process involved in reaching a diagnostic or troubleshooting conclusion from the domain in which it is made. If possible, these quantified processes can then be applied to troubleshooting scenarios outside the domain in which they originated.

Rasmussen and Jensen published and article for Ergonomics in 1974 in which they detailed their work in analyzing the mental processes used by electronics repair technicians. Their purpose was to examine the routines employed by several technicians and determine if the successful technicians had routines that were similar. They identified three distinct processes that could be generalized to any troubleshooting domain.

While not specifically referring to the technicians' methods as a "heuristic", the authors compared the problem solving process to reading a "topographic map" of a diagnostic problem. The "map" supplements the troubleshooting diagram in that it is more detailed and may encompass factors such as the users experience level and past exposure to similar problems. Clearly such elements are impractical to include in a

traditional troubleshooting diagram.

The generalized procedures that were derived from this work consisted of the way that the technicians would "read" their "map" to solve the problem at hand. Significant in this work was the authors conclusion that many diagnostic processes outside the realm of electronics utilize a similar approach, ie, the problem solver has their own "map" specific to the domain. What is unique is the similarity in method of reading the map in order to derive a solution. This lends credibility to the research question in chapter I concerning generalizing heuristics.

IV. HEURISTIC DEVELOPMENT FOR DIAGNOSTIC SYSTEMS

This chapter describes the process used to generate the heuristics that constitute the basis of the prototype expert system. The use of predicate logic as a tool for representing heuristics will also be examined. The purpose is to create generalized diagnostic rules that can be used in a working prototype as well as other diagnostic domains.

A. HEURISTIC DEFINITION

A heuristic is defined as a procedure to develop an alternative without necessarily optimizing; in expert systems, applying decision rules which are not necessarily the best, but seem to perform well. (Olson and Courtney, 1992, pp. 407) Simply stated, a heuristic is a rule that works, even if it does not initially make sense. People create and use their own heuristics constantly, usually without realizing it. To differentiate between a rule and a heuristic, the following example is provided.

1. HEURISTIC EXAMPLE

Traffic laws are rules in the strictest sense. Rules like speed limits offer no room for interpretation. The rule for the speed limit on highway 101 is 55 miles per hour will not be exceeded. Driving in excess of 55 mph will result in a ticket for speeding. However anyone who has driven on that freeway (or any others in California) knows that traffic frequently moves in excess of 55 mph. The reason is that most drivers have developed their own heuristic for driving on the freeway.

The freeway driving heuristic results from the drivers knowledge base,

observation of driving conditions, and past driving experiences. The drivers knowledge base says there a limited number of Highway Patrol officers. Observation of conditions show that almost nobody is obeying the speed limit plus they haven't seen a highway patrol officer on the freeway on this day. Past experience on this freeway says that usually drivers that get pulled over are way in excess of the speed limit, enough to make them stand out from the rest of the traffic which is also speeding. The resulting heuristic is "If I drive at 63 mph, I can blend into traffic and probably not get caught".

Use of this simple heuristic also demonstrates on of the faults of relying on them; they don't always work.

> "Heuristics can wipe out huge, unusable portions of the search tree leaving only those branches most likely to lead to a satisfactory goal. On the other hand, they are not always guaranteed to work". [Frenzel, 1987, pp. 61]

If applied frequently and over time, the freeway heuristic will result in a ticket.

Troubleshooting and diagnostics are common uses of heuristics. An experienced technician will often resolve a problem indication by making use of a heuristic that stems from the same type of framework that the freeway driving example does. The technician has a knowledge base specific to the domain he works in, can make observations of the status of the equipment, and has past experience in dealing with similar failure indications. The resulting heuristic may not make logical sense in terms of established procedures, like those found technical manuals, but that is irrelevant because it solves the problem.

2. BENEFITS OF GENERALIZATION

Expert systems frequently make use of heuristics. And, as was shown in Chapter

III, several systems have been designed specifically for diagnostics that use heuristics where rules or tree diagrams cannot accurately depict. The shortcoming of these system is that the heuristics developed are specific to the domain. This makes use of those heuristics as the basis for expert systems outside of that domain impractical.

Many aspects of diagnostics and troubleshooting are similar, irrespective of the domain. A mechanic, electronics technician, or a heart surgeon may use the same mental processes to solve a diagnostic problem. This thesis hypothesizes that there may be perhaps a dozen general troubleshooting or diagnostic heuristics that would may encompass the majority of diagnostic processes. If these processes can be captured, the result will be generalized diagnostic heuristics that can be incorporated into a wide range of expert systems.

3. HEURISTIC REPRESENTATION

Representing heuristics can be done using any one of several forms, plain English, pseudo-code, computer code, or predicate logic are just a few of the representation schemes. The above example obviously uses plain English, which has the advantage of being easy to understand. Unfortunately, if the goal is to translate the heuristic into computer code it is more difficult using English constructs. For this reason, pseudo-code will be used to represent the heuristics for this project.

B. HEURISTIC GENERATION

The heuristics generated for this thesis were the result of interview, experience, and observation of typical diagnostic processes used by technicians. Additionally, use is made of the work of Rasmussen and Jensen in describing the mental procedures used by

electronics technicians in troubleshooting. The processes examined are by no means limited to the electronics domain, as automotive technicians and medical personnel were also included in the generation process.

No quantified method of developing heuristics was found in the literature, but after a series of trial and error, a workable system for capturing heuristics was developed for the purposes of this thesis. Making use of observation and interview, the heuristic was first written out in plain English. An example of a troubleshooting scenario was then postulated to test the validity of the English representation. The heuristic was then transformed into pseudo-code, and from this point rewritten in CLIPS in modular format for inclusion in the prototype Expert System.

The research provided the basis for the modeling of four diagnostic processes. It must be noted that these four are not an exhaustive list. It is hypothesized that a thorough list of generalized diagnostic heuristics would be longer, ie, there is considerable room for expansion. The heuristic are identified as follows: Abduction, Commonality, Cross-commonality, and Complexity.

1. Abduction

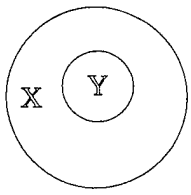
The Abduction heuristic is the simplest and probably the most frequently used diagnostic tool. This heuristic makes use of the technicians knowledge, experience, and observation of the systems' condition to hypothesize a corrective action. Past experience with a particular piece of equipment may allow the technician to jump to a corrective action that is not specifically within the "flow" of an established procedure or troubleshooting tree diagram.

Abduction compensates for those repair actions that do not at first make sense, except for the fact that they work. This condition is frequently encountered in troubleshooting, and may in fact be impossible or impractical to be included in a logic diagram or troubleshooting tree. That is precisely the advantage of heuristics however; its immaterial if the solution can be rigorously documented or diagramed, so long as it works.

a. Pseudo-code Representation

Statements:

1. component_of(Y, X);	Y is a component of X
2. req_to_operate[ok(Y), X];	X requires a working component Y to operate



Heuristic: If system X has failed, and Y is a component of X, and X requires Y to operate, then check component Y.

2. Commonality

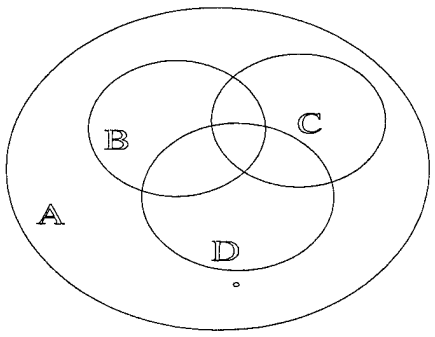
Consider a system made up of components, which requires all of those components to function correctly in order for the system to operate correctly. The components have a common intersection, for example a series of pumps with a common

power source or common cooling system. If the system as a whole has failed, then the first step to take would be to check the intersection of all the components, ie, the common point of failure. If unsuccessful, check the next largest intersection, iterating the process of checking points of intersection in descending order until the problem is successfully corrected. This process may be iterated until individual components are checked. Commonality allows for the elimination of superfluous steps, as the most likely point of failure is the point where the largest intersection occurs.

a. Pseudo-code Representation

Statements:

components_of[(B,C,D), A];	B, C, and D are components of A
req_to_operate[ok(B,C,D), A];	A requires B, C, and D to operate
common([B,C,D]);	B, C, and D have a common intersection



Heuristic: If system A has subcomponents B, C, and D, and these subcomponents have a common intersection, in the event of failed A, first check the intersection of B, C, and D.

3. Cross Commonality

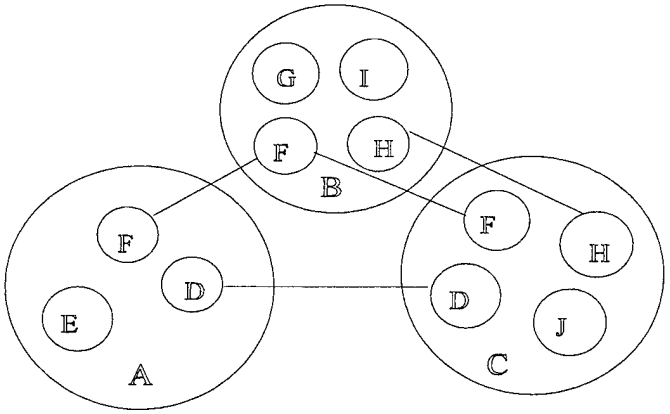
Cross-commonality is the condition where two or more systems with separate functions must operate in concert. A shipboard example of this would be a steam turbine, a reduction gear assembly, and a propeller drive shaft. These systems can operate independently of one another but must work together to produce a useful output; making a ship move through the water.

Each of these may have components in common, for example a lube oil system, an electrical power source for pumps and valves, and a heat exchanger may serve all three. If there is a failure indication of either the turbine, reduction gear, or propeller shaft while the other two continue to operate normally, the initial domain to check would be those components which are exclusively used by the failed system and excluded because the other two systems continue to function.

a. Pseudo-code Representation

Statements:

components_of[(D,E,F), A];	D, E, and F are components of A
components_of[(F,G,H,I), B];	F, G, H, and I are components of B
components_of[(D,F,H,J), C];	D, F, H, and J are components of C
req_to_operate[(D,E,F),A];	A requires D, E , and F to operate
req_to_operate[(F,G,H,I),B];	B requires F, G, H, and I to operate
req_to_operate[(D, F,H,J),C];	C requires D, F , H and J to operate



Heuristic: If systems A, B, and C require their respective subsystems to operate and system B has failed while A and C continue to operate, check those components of B which are not shared by A and C.

4. Complexity

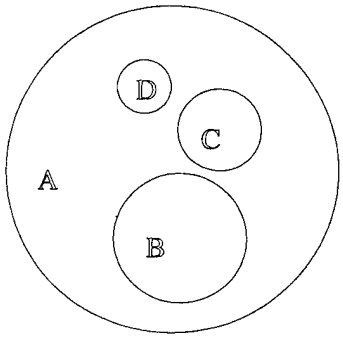
This heuristic captures the condition where a system is made up of components which have the characteristic of "relative complexity". What is meant by relative complexity is that one component will have a number of subcomponets, the next will have less, and so on in descending order. In the event of a system failure, the component with the highest likelihood of causing it is the one with the most subcomponents, therefore check first the one with the greatest degree of relative complexity.

a. Pseudo-code Representation

Statements:

components_of[(B,C,D),A]; B, C, and D are components of A

req_to_operate[(B,C,D),A];	A requires B, C, and D to operate
relative_complexity(B>C>D);	B is more complex than C which is more complex than D



Heuristic: System A has subcomponents B, C, and D, and these subcomponents are in descending order of relative complexity. A requires B, C and D to operate, so if system A has failed, first check the subcomponent with highest degree of complexity, B.

C. PROTOTYPE KNOWLEDGE ACQUISITION

Knowledge acquisition used a hybrid approach, which took advantage of the method proposed by Chorafas. The major difference in acquisition methodology occurred in timing, and this was due to the heuristic design phase. The heuristic design phase involved some exposure to the avionics technicians and the troubleshooting processes during model development, therefore some of the domain knowledge acquisition occurred here. This isn't in strict accordance to Chorafas' development methodology, as some of the domain knowledge is exposed earlier than required, however this had little impact on prototype development. Domain selection criteria and target system selection for prototype development are as discussed in Chapter II.

1. Initial Steps

The first step in creating the prototype Avionics Diagnostic Advisor (ADA) was to examine the Crew Station Manuals (CSM) NAVAIR 01 PAC 12-5. These are the technical manuals for the P-3C onboard equipment, and are maintained onboard the aircraft. Included in the CSM are basic troubleshooting and repair procedures. These procedures are simplified based on the assumption that they will be carried out by the operators of that particular equipment either in-flight or during preflight. Troubleshooting procedures are generally in the form of tree diagrams, and these diagrams formed the starting point for expert system knowledge.

Where applicable, the individual troubleshooting steps were "classified" as making use of one of the heuristic models. Most of the tree-diagram type of procedures fit closely with abduction. When the experts knowledge was added, there were similarities with the other heuristic models, however there are some that are difficult to classify using just those four models. This lends credence to the hypothesis that several additional models can be created for a more thorough set of heuristic tools.

Additional troubleshooting information was gathered from the technical manuals used in the avionics work center. These manuals are similar to the CSM's, but contain more detailed information as they are used in a "shop" environment vice onboard the aircraft. For the radar system, the NAVAIR 01 PAC 2-8.1 was used. Once sufficient information was obtained, the first iteration of the prototype was begun. The first iteration did no more than automate the existing repair literature. This may at first seem redundant, however making use of existing documented procedures for the purposes of prototyping

is advocated by both Chorafas and Prerau. This allows for more rapid development of the prototype in order to establish a baseline. The next step is to identify an expert or experts and incorporate their knowledge into the prototype.

2. The Expert

Identification of an expert in this environment was relatively easy. Patrol Squadron (VP) 91 is a reserve P-3C squadron which has a core of full-time personnel for administrative purposes. Due to the nature of a reserve squadron the experience level of many of the avionics technicians is quite high. Some of the in-flight technicians that aided in development of the prototype have as much as sixteen continuous years of P-3C experience. Additionally, VP-91 has a NAESU representative working in the same hangar who has twenty four years experience with P-3C avionics. The problem now became who out of this group to use? In order to simplify the variety of inputs available from such a large group, it was decided to use the two most experienced personnel; the Avionics Shop supervisor, and the NAESU representative.

D. IMPLEMENTATION

Once the steps in the troubleshooting procedure were converted to English, those that conformed to the models were re-written using pseudo-code similar to the models. The procedures that were unique were re-written in a simple If/Then format, similar to pseudo-code representations. This aided in converting unique procedures to CLIPS and provides a source or additional models. After only a few repetitions of this process it became easier and quicker to write the individual steps in the troubleshooting procedure directly into the CLIPS prototype. The result was a series of diagnostic procedures that

were in modular format. The modules were then combined into a working structure for the prototype system which roughly followed the guideline of the original troubleshooting diagram from the CSM.

E. PROTOTYPE DESIGN

Several diagnostic expert systems have already been fully implemented in CLIPS. To aid in creating a working prototype expediently, parts of existing CLIPS expert systems were rewritten and incorporated into the prototype. Singular use was made of an automotive troubleshooting example program which is supplied with the CLIPS software package.

Modules of troubleshooting procedures from the CSM and other technical manuals were added to the existing program, resulting in the first iteration of the prototype. This version was demonstrated to organizational level technicians to insure that all of the previously documented procedures were fully captured .Once this was completed, the knowledge of the expert was added incrementally and tested by having both the O level technicians and the NAESU representative run the program as changes were incorporated. Results of the test were positive and will be covered in detail in Chapter V.

V. TEST AND VALIDATION

The majority of implementation issues were addressed in the previous chapter, this chapter will focus on the test and validation process. The test and validation phase proceeded on two fronts: first to determine if the prototype system actually worked, and second, to determine if the heuristics were valid.

A. PROTOTYPE TESTING

The initial prototype consisted of the APS-115 crew station manual (NAVAIR 01-75PAC-12-5, 1995, section 4, pp. 13-16) troubleshooting tree written in CLIPS. The development employed was to demonstrate this version to the users, use their commentary and suggestions to update the prototype, and demonstrate again. As these changes were incorporated, the prototype expert system began to more closely resemble the methods used by the experts. Many positive suggestions were gathered, unfortunately time constraints allowed for only two iterations of the improvement process.

1. Initial Prototype Improvements

One of the first improvements suggested by the users was to rearrange the order of appearance of some of the troubleshooting steps in the CSM to correspond more closely with the order that an experienced technician uses them. The CSM presents decision nodes for repair after a test node. Typically a go/no-go light indication begins a specific test series. If the component fails the go/no-go indication a more detailed series of test s are undertaken on that component to isolate the fault.

The decision nodes are presented in ascending order of relative complexity,

starting with the simplest items to repair. Most of the test series start with tests of the mechanical or electrical portions of the component that is being tested. Typical of this process is test module 12, the test for 360 degree scan of the antenna. Test module 12 is initiated by checking the video display for a 360 degree scan pattern, which tells the operator that both antennas are sweeping correctly. If their is a failure indication at this point, the test series proceeds by checking the mechanical tilt and stabilization components, followed by the electrical servo motors, and finally with the electronic portion of the radar position programmer.

Experience with this failure indication and series of tests leads the avionics technician to cut off the electro-mechanical portion of the decision tree and initially check the test points (TP) 1A11A1 TP 1 or TP 7. While they acknowledge that the electro-mechanical failures of the radar component are easier to fix, ie, take less technical expertise, they are encountered so infrequently that they are skipped.

2. Subsequent Improvements

Once the order of appearance of decision nodes had been rearranged, the prototype system was again tested by the end users. At this juncture more of the experts knowledge was incorporate into the system.

Using the knowledge of the NAESU representative and the avionics shops senior technician, some of the decision nodes were modified to include more comprehensive testing. An example of this is decision node 14.42, where the R/T crystal are tested to determine if the crystals are causing an R/T failure indication. The procedure from the CSM uses a simple go/no-go light indication to check each of the three R/T crystals. This

test may reveal if a crystal is failed, but will not reveal if one is marginal or near failure. A marginal crystal can cause an R/T failure, yet still test satisfactorily if the CSM procedure is followed. This step was changed in order to incorporate testing each individual crystal with an oscilloscope.

Time allowed for only the inclusion of a few of the improved test procedures into the prototype, there are several remaining that would improve the accuracy of the system that could not be included. Once again the improved test procedures were incorporated into the prototype and demonstrated to the end users with satisfactory results.

3. Knowledge Acquisition Issues

Some of the classic knowledge acquisition issues detailed in the references (Prerau, 1990) were also encountered in prototype development. Issues such as resistance to new methods, fear of replacement of the expert by a computer, and reluctance to reveal knowledge of procedures that were not in strict compliance with regulations were typical. In general, after working in the avionics shop and familiarizing the personnel with the system, the problems became less of an issue. Knowledge acquisition issues will be covered in more detail in the conclusion.

B. HEURISTIC TESTING

Once the prototype was working it remained to test the system to determine if the heuristics designed in Chapter IV were applicable. In general, all of the four heuristics were used in parts of the troubleshooting procedures, although frequently it was

impossible to draw a strong distinction as to where one heuristic ended and another began. Many steps within the troubleshooting procedure used more than on of the heuristics simultaneously. Additionally, some parts of the troubleshooting process could not be categorized by any of the heuristics, which adds credence to the theory that there are more heuristics to be developed.

1. Abduction

This was the simplest heuristic and the most frequently applied. Given an indication, perform an action. The action could be the termination of a repair procedure, for example decision node 12.1, which state if presented with an indication of +5VDC on the fault isolation switch, replace printed circuit card 1A6. Or the action could be to continue with another test and continue the troubleshooting process. Almost all of the decision nodes make use of abduction.

2. Commonality

The most obvious use of the commonality heuristic occurs in decision node 14 series. Staring with a R/T failure indication, subcomponents of the R/T portion of the radar are examined for failure individually. The final series of tests involves checking the electronic components of the R/T, culminating in checking the crystals. At this point all of the exclusive electronic components have been eliminated, al that remains is the point of common intersection, where the video signal is processed by the crystals. In fact, checking the crystals is one of the first procedures that was given a higher precedence in the order in which subcomponents of the radar system are evaluated.

3. Cross-Commonality

The APS-115 radar system actually consists of two separate radar systems, including individual antennas and R/T's, up to the point where the signal is combined from the forward and aft antennas and presented at the operators screen as processed video. An example of the application of this heuristic is the swapping of the forward and aft R/T sets.

Decision node 14.26 culminates with the swapping of the forward and aft R/T sets to determine if one has failed. The operator will see a failure indication at his console, but not know if components within the forward or aft system are at fault. If the R/T sets are substituted, the technician can determine if the fault lies somewhere outside of either R/T. This allows for the exclusion of the R/T set as a primary cause of failure, meaning that the problem is located in a component that nether forward or aft system are sharing.

4. Complexity

The use of this heuristic is evident when the order of the decision nodes is shuffled. The CSM presents the troubleshooting steps in ascending order of complexity based on the fact that mechanical or electrical repairs are the easiest to perform. One of the first changes made to the prototype was placing the electronic portion of a test series at the beginning of the series vice the end. The more experienced technicians knew that the most likely failure would take place in the most complicated portion of system or subsystem, the electronic portion.

C. FINAL TEST

After two iterations of the update process, the prototype was again demonstrated to the end users. More suggestion for improvement were made, unfortunately they could not be incorporated due to time constraints.

The final version of the prototype was well received once the user was trained in its operation. The CLIPS interface was not particularly user friendly, appearing similar to a DOS application, however the information contained in the prototype was judged to be accurate and useful by the users. The final version of the prototype had some anomalies that required the designer to be present while the program was being run, further progress is still undergoing to clear up these anomalies.

VI. LESSONS LEARNED

This chapter presents some insights gained through the experience of developing the Avionics Diagnostic Advisor system as well as recommendations for further development.

A. KNOWLEDGE ACQUISITION

As discussed in Chapter V, knowledge acquisition issues closely followed those revealed in the expert systems literature. Specifically, at the beginning of the knowledge acquisition session, there was a great deal of skepticism and reluctance on the part of the experts.

The more experienced avionics technicians were initially skeptical of an expert system for the APS-115 radar. The consensus opinion was that after prolonged exposure to the APS-115 system any technician would reach a sufficient level of competence required of any repairs. Additionally, the APS-115 had been operating for a long period in the fleet, any particular anomalies that affected it were already well known.

After it was explained that it was the time which elapses as a new avionics technician gains experience that was being attacked, the use of an expert system gained more acceptance. In the course of demonstrating the prototype, one of the junior technicians was exposed to a previously unknown procedure for repair. This sparked a discussion with the shop supervisor in which the junior technician gained insight into troubleshooting the APS-115. A clear point was made as to the advantages of expert

systems, particularly in regards to training.

Naval Aviation maintenance follows strict guidelines as to what type of repairs can be undertaken at a particular level. There was some reluctance on the part of the two experts to impart some of the procedures they employed due to this. Exposure to repair functions carried out at the intermediate level was the basis of some of these procedures. There was a tacit feeling on the part of the experts that repercussions might entail if these repair actions were documented in an expert system.

B. KNOWLEDGE REPRESENTATION

Initially, predicate logic was attempted for knowledge representation, however this proved to be more difficult than initially thought, particularly with regards to the heuristics from Chapter IV. Abduction and Complexity were relatively easy, but commonality and cross-commonality were hard to work out. Therefore a decision was made to utilize pseudo-code to quantify the heuristics.

Representation in pseudo-code aided the design phase when the prototype was written. The heuristics when written in pseudo-code are similar to the way knowledge is represented in CLIPS. CLIPS uses a right-hand-side for indications, left-hand-side for action format to represent rules. This corresponded closely to the way the heuristics were actually employed.

C. IMPLEMENTATION

Implementing the knowledge in CLIPS was time consuming, although after experience in using CLIPS was gained it was fairly straightforward. There were uncovered a number of sample programs, from the examples in the software package and

from the CLIPS bulletin board which dealt with troubleshooting and diagnostics. Design was merely a matter of choosing an existing program and modifying it until the output matched the troubleshooting tree from the CSM. From that point it was simple to add expert knowledge by editing the existing code. The final version of the prototype made extensive use of the Automotive Troubleshooting Sample program to create the ADA prototype.

D. CLIPS INTERFACE

The CLIPS version used for this project was 6.0 for Windows. This version, while Windows compatible, does not use many Windows features. Basically the 6.0 version is the same as the DOS version, it presents the same interfaces screens, only they are in a DOS window vice a DOS screen. The editing interface for the designer is also similar to the DOS version, with the addition of limited mouse compatibility for editing.

The user interface for the final version is also in a DOS window. The only advantage over the DOS version is that a few of the commands, such as run or reset can be accessed via drop-down windows rather than typed directly on the screen. Users had a hard time making the system run correctly, so much so that the author had to guide them through a few sessions before the end user could run it on their own. Adding a better interface to the CLIPS code is an obvious avenue to pursue.

E. CONCLUSION

Expert systems application to diagnostics is not new, as shown in Chapter III there are many existing systems and many more likely to be developed. Capturing and representing heuristics in an expert system has also been applied in many instances.

Research for this project revealed that while work in generalizing heuristic methods across different domains has been explored (Rasmussen and Jensen, 1974) none have as of yet been applied to an expert system.

Follow on research to this project should focus on finding more generalized heuristics to add to those already developed. This will make further development of the prototype easier and provide an avenue to expand into the rest of the avionics realm.

APPENDIX A. PROTOTYPE SYSTEM CODE

```
;;;======================================================
;;;  AVIONICS DIAGNOSTIC ADVISOR EXPERT SYSTEM V 1.0
;;;
;;;          APS 115 MODULE
;;;
;;;   This expert system diagnoses some simple
;;;   problems with the APS 115 RADAR set in the
;;;   P-3C UIII. Taken from APS 115 CSM
;;;
;;;        CLIPS Version 6.0 Example
;;;
;;;   To execute, merely load, reset and run.
;;;
;;;======================================================


;;*****************************************************
;;*             DEFFUNCTIONS                  *
;;* This defines the yes or no and the three-way response*
;;*****************************************************


(deffunction ask-question (?question $?allowed-values)
  (printout t ?question)
  (bind ?answer (read))
  (if (lexemep ?answer)
     then (bind ?answer (lowcase ?answer)))
  (while (not (member ?answer ?allowed-values)) do
    (printout t ?question)
    (bind ?answer (read))
    (if (lexemep ?answer)
       then (bind ?answer (lowcase ?answer))))
  ?answer)

(deffunction yes-or-no-p (?question)
  (bind ?response (ask-question ?question yes no y n))
  (if (or (eq ?response yes) (eq ?response y))
     then TRUE
```

```
    else FALSE))

;;;*****************************
;;;
;;;* APS 115 RADAR STATE RULES *
;;;* Radar either SAT or UNSAT *
;;;*****************************
;;;

(defrule normal-radar-state-conclusions ""
  (declare (salience 10))
  (working-state radar normal)
  =>
  (assert (repair "No repair needed."))
  (assert (all/checks-state radar normal))
  (assert (lamp/comp/fan-state all sat))
  (assert (initialize-state radar initialized)))


(defrule unsatisfactory-radar-state-conclusions ""
  (declare (salience 10))
  (working-state radar unsatisfactory)
  =>
  (assert (lamp/comp/fan-state all sat))
  (assert (initialize-state radar initialized)))

;;;*********************************
;;;
;;;*       QUERY RULES           *
;;;* *******************************
;;;

;;;      Decision Node 1-10

(defrule determine-radar-state ""
  (not (working-state radar ?))
  (not (repair ?))
  =>
  (if (yes-or-no-p "Does radar initialize (yes/no)? ")
     then
     (if (yes-or-no-p "Are all steps completed satisfactorily (yes/no)? ")
        then (assert (working-state radar normal))
     . . else (assert (working-state radar unsatisfactory)))
     else
     (assert (working-state radar does-not-initialize))))

;;;      Decision Node 2 and 3
```

```
(defrule determine-lamp/comp/fan-state ""
  (working-state radar does-not-start)
  (not (initialize-state radar ?))
  (not (repair ?))
  =>
  (if (yes-or-no-p "Do the lamps,compressor, and fans operate (yes/no)? ")
     then
     (assert (lamp/comp/fan-state radar check sat))
     (assert (all/checks-state radar unsat-all/checks))
     else
     (assert (lamp/comp/fan-state radar does-not-check-sat))
     (assert (all/checks-state radar does-not-check))))


;;;      Decision node 12

(defrule determine-360-display ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Is the 360 scan present (yes/no)? ")
        then
        (assert (360-display-state radar sat))
        (assert (all/checks-state does-not-check));;; go to DN 13
        else
     (assert (repair "Verify ch 4 of SDD operates properly."))))

...      **********************************************
;;;
;;;      Decision node 13, branches to DN 12 subgroups
...      **********************************************
;;;

(defrule determine-app-fail-lamp ""
  (working-state radar unsatisfactory)
  (not (app-fail-lamp-state ?))
  (not (repair ?))
  =>
  (if (yes-or-no-p "Does APP fail lamp on radar control go off (yes/no)? ")
     then
     (assert (app-fail-lamp-state sat))
     (assert (all/checks-state bad))
        else
        (assert (app-fail-lamp-state unsat))))
```

```
;;;      DN 13.1


(defrule determine-hv-off-state ""
  (working-state radar unsatisfactory)
  (not (hv-off-state radar ?))
  (not (repair ?))
  =>
  (if (yes-or-no-p "Turn HV off on RSC, check +5V dc posit on APP fault isolate switch
(yes/no)? ")
     then
     (assert (hv-off-state sat))
     (assert (all/checks-state does-not-check))
         else
         (assert (repair "replace Module 1A6 and/or 1A7.")))) ;;; First terminal


(defrule detemine-15vdc-app-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Is +/-15V dc indicated on APP fault iso sw(yes/no)? ")
     then
        (assert (15vdc-app-state sat))
        (assert (all/checks-state does-not-check))
          else
          (assert (repair "Replace module 1A3 and/or 1AX."))))

(defrule detemine-synchro-posit-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check SYNCHRONIZER posit on APP fault iso sw(yes/no)? ")
     then
        (assert (synchro-posit-state sat))
        (assert (all/checks-state does-not-check))
          else
          (assert (repair "Replace sync module 1A8 and/or bite module 1A11."))))


(defrule detemine-fwd-az-servo-amp-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
```

```
  =>
  (if (yes-or-no-p "Check FWD AZ SERVO AMP posit on APP fault isolation
switch(yes/no)? ")
      then
        (assert (fwd-az-servo-amp-state sat))
        (assert (all/checks-state does-not-check))
         else
         (assert (repair "Replace module 1A4 and/or bite module 1A11."))))

(defrule detemine-aft-az-servo-amp-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check AFT AZ SERVO AMP posit on APP fault isolation
switch(yes/no)? ")
      then
        (assert (aft-az-servo-amp-state sat))
        (assert (all/checks-state does-not-check))
         else
         (assert (repair "Replace module 1A5 and/or bite module 1A11."))))

(defrule detemine-fwd-az-gear-assy-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check FWD AZ GEAR ASSY posit on APP fault isolation
switch(yes/no)? ")
      then
        (assert (fwd-az-gear-assy-state sat))
        (assert (all/checks-state does-not-check))
         else
         (assert (repair "Replace module 3A1 and/or bite module 1A11."))))


(defrule detemine-aft-az-gear-assy-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check AFT AZ GEAR ASSY posit on APP fault isolation
switch(yes/no)? ")
      then
        (assert (aft-az-gear-assy-state sat))
        (assert (all/checks-state does-not-check))
```

```
        else
        (assert (repair "Replace module 8a1 and/or bite module 1A11.")))))


;;;     Decision Node 12.20

(defrule detemine-az-scan-prgrmr1-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check AFT AZ SCAN PROGRAMMER #1 posit on APP fault
isolation switch(yes/no)? ")
      then
        (assert (az-scan-prgrmr-state sat))
        (assert (all/checks-state does-not-check))
         else
         (assert (repair "Replace AZ Scan programmer 1A9."))))


(defrule detemine-az-scan-prgrmr2-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check AFT AZ SCAN PROGRAMMER #2 posit on APP fault
isolation switch(yes/no)? ")
      then
        (assert (az-scan-prgrmr2-state sat))
        (assert (all/checks-state does-not-check))
         else
         (assert (repair "Replace AZ Scan programmer 1A9."))))

;;;     Decision Node 12.24


(defrule detemine-fwd-tilt-servo-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check FWD TILT SERVO AMP posit on APP fault isolation
switch(yes/no)? ")
      then
        (assert (fwd-tilt-servo-state sat))
        (assert (all/checks-state does-not-check))
```

```
        else
        (assert (repair "Replace FWD TILT SERVO AMP 1A1.")))

(defrule detemine-aft-tilt-servo-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check AFT TILT SERVO AMP posit on APP fault isolation
switch(yes/no)? ")
      then
        (assert (aft-tilt-servo-state sat))
        (assert (all/checks-state does-not-check))
         else
         (assert (repair "Replace FWD TILT SERVO AMP 1A2.")))

(defrule detemine-tilt-function-gen-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check TILT FUNTION GEN posit on APP fault isolation
switch(yes/no)? ")
      then
        (assert (tilt-function-gen-state sat))
        (assert (all/checks-state does-not-check))
         else
         (assert (repair "Replace Tilt function generator 1A13.")))

;;;      Decision Node 12.30

(defrule detemine-fwd-tilt-gear-assy-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check FWD TILT GEAR ASSY posit on APP fault isolation
switch(yes/no)? ")
      then
        (assert (fwd-tilt-gear-assy-state sat))
        (assert (all/checks-state does-not-check))
         else
         (assert (repair "Check 1A11A1TP1, if yes, replace Bite Module 1a11, if no
repace FWD TILT GEAR ASSY 3A2.")))
```

```
(defrule detemine-aft-tilt-gear-assy-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check AFT TILT GEAR ASSY posit on APP fault isolation
switch(yes/no)? ")
      then
        (assert (aft-tilt-gear-assy-state sat))
        (assert (all/checks-state does-not-check))
         else
         (assert (repair "Replace BITE Module 1A11."))))

(defrule detemine-1a11a1tp11-state ""
  (working-state radar unsatisfactory)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check 1A11A1TP1 (yes/no)? ")
     then
        (assert (1a11a1tp1-state sat))
        (assert (all/checks-state does-not-check))
        (assert (repair "Replace BITE Module 1A11."))
         else
         (assert (repair "Replace AFT TILT GEAR ASSY 8A2."))))



;;;       Decision Node 14


(defrule determine-rt-fail-lamp-state ""
  (or (and (working-state radar unsatisfactory)
           (all/checks-state bad))
     (symptom radar low-output))
  (not (repair ?))
  =>
  (if (yes-or-no-p "Do the RT FAIL lamps on radar control go off (yes/no)? ")
        then
        (assert (rt-fail-lamp-state sat))
      ' ' (assert (all/checks-state neg))  ;;; jump to DN 16
        else
        (assert (rt-fail-lamp-state unsat))
        (assert (all/checks-state broke))))
```

```
;;;     Starts Decision Node 16


(defrule determine-dummy-state ""
  (working-state radar unsatisfactory)
  (all/checks-state neg)
  (not (repair ?))
  =>
  (if (yes-or-no-p "***Depress LOAD switch two times***
Do the DUMMY and ANTENNA indicators come on alternately
on both FWD and AFT radar control (yes/no)? ")
        then
        (assert (dummy-state sat))
        (assert (all/checks-state neg)) ;;; goes to DN17
        else
        (assert (repair "Replace waveguide switch on applicable antenna
and/or replace applicable radar control."))))


;;;     DN 17


(defrule determine-stdby-indic-state ""
  (working-state radar unsatisfactory)
  (all/checks-state neg)
  (not (repair ?))
  =>
  (if (yes-or-no-p "If radar power has been applied for 3 min
STBY indicator comes on fwd and aft radar control (yes/no)? ")
        then
        (assert (stdby-indic-state sat))
        (assert (all/checks-state neg)) ;;; goes to DN 19
        else
        (assert (repair "Replace applicable radar control and/or
replace primary mode`logic module 1A10."))))


;;;     DN 19


(defrule determine-hv-on-state ""
  (working-state radar unsatisfactory)
  (all/checks-state neg)
```

```
  (not (repair ?))
  =>
  (if (yes-or-no-p "***Depress HV switch***
Does HV ON indicator come on fwd and aft radar control(yes/no)? ")
        then
        (assert (hv-on-state sat))
        (assert (all/checks-state neg)) ;;; goes 20
        else
        (assert (repair "Replace primary mode logic module of APP 1A12
and replace applicable radar control."))))


;;;     DN 20


(defrule determine-app-fail-2-state ""
  (working-state radar unsatisfactory)
  (all/checks-state neg)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Does APP FAIL on fwd and aft radar control go off(yes/no)? ")
        then
        (assert (app-fail-2-state sat))
        (assert (all/checks-state neg)) ;;; goes 21
        else
        (assert (app-fail-lamp-state unsat)))) ;;; should loop to 12.1


;;;     DN 21


(defrule determine-rt-fail-2-state ""
  (working-state radar unsatisfactory)
  (all/checks-state neg)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Does RT fail lamp on fwd and aft radar control go off(yes/no)? ")
        then
        (assert (rt-fail-2-state sat))
        (assert (all/checks-state neg)) ;;; goes 22
        else
        (assert (rt-fail-lamp-state unsat))
        (assert (all/checks-state broke)))) ;;; loop to 14.2
```

```
;;;      DN 23


(defrule determine-agile-short-state ""
  (working-state radar unsatisfactory)
  (all/checks-state neg)
  (not (repair ?))
  =>
  (if (yes-or-no-p "***Depress FREQ and PULSE switches fwd and aft radar control***
Do AGILE & SHORT indicators come on(yes/no)? ")
        then
        (assert (agile-short-state sat))
        (assert (all/checks-state neg)) ;;; goes 24
        else
        (assert (repair "Replace applicable radar control and primary
logic module 1A10."))))


;;;      DN 24


(defrule determine-rt-fail-3-state ""
  (working-state radar unsatisfactory)
  (all/checks-state neg)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Does RT fail lamp on fwd and aft radar control go off(yes/no)? ")
        then
        (assert (rt-fail-3-state sat))
        (assert (all/checks-state neg)) ;;; goes 26
        else
        (assert (rt-fail-lamp-state unsat))
        (assert (all/checks-state broke)))) ;;; loop to 14.2

;;;      DN 26

(defrule determine-240-sector-state ""
  (working-state radar unsatisfactory)
  (all/checks-state neg)
  (not (repair ?))
  =>
  (if (yes-or-no-p "***Depress HV switch on fwd and aft radar control***
```

```
***Place PWR/OFF switch in OFF position***
240 degree sector sweep should be on display
and APP fail lamp on fwd and aft radar control goes off(yes/no)? ")
        then
        (assert (repair "***Depress HV switch on aft radar control***
***Place PWR/OFF switch in OFF position***
>>ALL CHECKS COMPLETE<<"))        ;;; End of TS guide
        else
        (assert (repair "Replace slave mode logic module 1A12."))))


;;;     DN 14.2

(defrule determine-wg-press-state ""
  (working-state radar unsatisfactory)
  (all/checks-state broke)
  (not (repair ?))
  =>
  (if (yes-or-no-p "**NOTE following switch posits refer to R/T fault isolation switch***
Check W/G PRESS switch position (yes/no)? ")
        then
        (assert (wg-press-state sat))
        (assert (all/checks-state busted))
          else
          (assert (repair "Replace compressor 2A7/7A10 and/or bite mod 2A10/7A10"))))


(defrule determine-20vdc-posit-state ""
  (working-state radar unsatisfactory)
  (all/checks-state busted)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check both posits of +/-20V DC (yes/no)? ")
        then
        (assert (20vdc-posit-state sat))
        (assert (all/checks-state busted))
          else
          (assert (repair "Replace +/-20V DC module 2A11/7A11 and/or bite module
2A10/7A10"))))

(defrule determine-thy-trig-state ""
  (working-state radar unsatisfactory)
  (all/checks-state busted)
```

```
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check THY TRIG switch position (yes/no)? ")
        then
        (assert (thy-trig-state sat))
        (assert (all/checks-state icky))
        else
        (assert (thy-trig-state unsat))
        (assert (all/checks-state busted))))

(defrule determine-radar-trgr-sync-state ""
  (working-state radar unsatisfactory)
  (all/checks-state busted)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Replace XMTR ASSY, check radar trigger sync (yes/no)? ")
        then
        (assert (radar-trgr-sync-state sat))
        (assert (all/checks-state busted))
        else
        (assert (repair "Check cable to unit 2/7 J4"))))


(defrule determine-tp3-state ""
  (working-state radar unsatisfactory)
  (all/checks-state busted)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Check 2A8/7A8 at test point 3 (yes/no)? ")
        then
        (assert (repair "Replace XMTR ASSY 2A1/7A1"))
        else
        (assert (repair "Replace Isolation Amplifier module 2A8/7A8"))))



(defrule determine-fil-trig-state ""
  (working-state radar unsatisfactory)
   (all/checks-state icky)
     (not (repair ?))
  =>
  (if (yes-or-no-p "Check FIL TRIG switch position (yes/no)? ")
     then
```

```
      (assert (fil-trig-state sat))
      (assert (all/checks-state afu))
      else
      (assert (all/checks-state busted)))) ;;; need this to loop to 14.9

(defrule determine-thy/mod-state ""
  (working-state radar unsatisfactory)
   (all/checks-state afu)
    (not (repair ?))
  =>
  (if (yes-or-no-p "Check THY/MOD switch position (yes/no)? ")
     then
     (assert (thy/mod-state sat))
     (assert (all/checks-state wfu))
     else
     (assert (repair "Replace bite module 2A10/7A10 and/or XMTR ASSY 2A1/7A1."))))


;;; Decision Node 14.18 branches again

(defrule determine-mag/mod-state ""
  (working-state radar unsatisfactory)
   (all/checks-state wfu)
    (not (repair ?))
  =>
  (if (yes-or-no-p "Check MAG/MOD switch position (yes/no)? ")
     then
     (assert (mag/mod-state sat))
     (assert (all/checks-state tfu))
     else
     (assert (mag/mod-state unsat))
        (assert (all/checks-state silly))));;; branch down to silly


(defrule determine-afc-man-state ""
  (working-state radar unsatisfactory)
   (all/checks-state silly)
    (not (repair ?))
  =>
  (if (yes-or-no-p "***Turn video test switch video test on radar control fwd & aft***
***Place display unit to 'A' scan***
Are minimum 12 video pulses(LP) or 6 pulses(SP) visible
in both AFC & MAN switch posits (yes/no)? ")
```

```
      then
      (assert (afc-man-state sat))
      (assert (all/checks-state stupid))
      else
      (assert (repair "***Check 2A8/7A8, test point 7 for 5.75V dc +/- .75***
If yes, replace XMTR ASSY 2A1/7A1.
If no, replace Isolation AMPL module 2A8/7A8."))))

;;;       DN 14.24

(defrule determine-j1-state ""
  (working-state radar unsatisfactory)
  (all/checks-state stupid)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Does 2A6/7A6 at J1 indicate -.6 minimum (yes/no)? ")
        then
        (assert (j1-state sat))
        (assert (all/checks-state stupid))
        else
        (assert (repair "Replace 2A6/7A6 CR 1."))))

;;;       DN 14.26


(defrule determine-2/7a10-tp1-state ""
  (working-state radar unsatisfactory)
  (all/checks-state stupid)
  (not (repair ?))
  =>
  (if (yes-or-no-p "Does 2A10/7A10 at test point 1
indicate 2.5/0.5 micro sec pulse, 1 +/- .4 volts (yes/no)? ")
        then
        (assert (repair "Replace Bite Module 2A10/7A10."))
        else
        (assert (repair "Replace Receiver Transmitter."))))


;;;       DN 14.30 branches again

(defrule determine-freq-agile-state ""
 (working-state radar unsatisfactory)
   (all/checks-state tfu)
```

```
    (not (repair ?))
  =>
  (if (yes-or-no-p "***Press FREQ switch on radar control panel***
***to place transmitter in Agile Mode***
Check AGILE switch position (yes/no)? ")
      then
      (assert (freq-agile-state sat))
         (assert (all/checks-state tfu))  ;;; go to 14.36
      else
         (assert (freq-agile-state unsat))
         (assert (all/checks-state goofy)))) ;;; go to 14.31

;;;       14.31


(defrule determine-agile2-state ""
 (working-state radar unsatisfactory)
   (all/checks-state goofy)
    (not (repair ?))
  =>
  (if (yes-or-no-p "Does AGILE indicator on radar control come on (yes/no)? ")
     then
     (assert (agile2-state sat))
        (assert (all/checks-state goofy))        ;;; go to 14.33
     else
        (assert (repair "Replace Primary Logic Module 1A12"))))

;;;       DN 14.33


(defrule determine-2/7a10a2-state ""
 (working-state radar unsatisfactory)
   (all/checks-state goofy)
    (not (repair ?))
  =>
  (if (yes-or-no-p "Check 2/7A10A2 TP5 for 1/2 wave 71 Hz, 50V minimum (yes/no)? ")
     then
     (assert (repair "Replace AGILE module 2A12/7A12."))
     else
        (assert (repair "Replace Transmitter Assembly
including Magnetron2A1/2A7"))))

;;;       DN 14.37
```

```
(defrule determine-freq2-state ""
 (working-state radar unsatisfactory)
   (all/checks-state tfu)
     (not (repair ?))
  =>
  (if (yes-or-no-p "***Press FREQ sw on radar control, places xmitter in fixed mode***
Check STC switch position (yes/no)? ")
      then
      (assert (freq2-state sat))
         (assert (all/checks-state tfu))  ;;; go to 14.40
      else
         (assert (repair "Replace STC module 2A9/7A9."))))) ;;;

;;;       DN 14.40


(defrule determine-lo-rcvr-state ""
 (working-state radar unsatisfactory)
   (all/checks-state tfu)
     (not (repair ?))
  =>
  (if (yes-or-no-p "***Verify applicable radar control AFC/MAN switch***
***is in AFC posit***
Check LO RCVR XTAL 1 and 2 switch position (yes/no)? ")
      then
      (assert (lo-rcvr-state sat))
         (assert (all/checks-state waybad))      ;;; go to 14.46
      else
         (assert (lo-rcvr-state unsat))
         (assert (all/checks-state negat))))      ;;; go to 14.41

;;;       DN 14.41



(defrule determine-fault-iso-state ""
 (working-state radar unsatisfactory)
   (all/checks-state negat)
     (not (repair ?))
  =>
  (if (yes-or-no-p "Is fault isolation meter pointer sweeping (yes/no)? ")
```

```
      then
      (assert (fault-iso-state sat))
      (assert (all/checks-state waybad))          ;;; go to 14.46
      else
         (assert (fault-iso-state unsat))
         (assert (all/checks-state negat))))      ;;; go to 14.43


;;;      DN 14.43


(defrule determine-lo-rcvr2-iso-state ""
 (working-state radar unsatisfactory)
   (all/checks-state negat)
    (not (repair ?))
  =>
  (if (yes-or-no-p "***Substitute RCVR XTALS with AFC XTALS***
Check LO RCVR XTAL switch positions 1 and 2 (yes/no)? ")
     then
     (assert (repair "***Perform receiver tuning procedure prior to reinstalling crystals***
Replace appropriate RCVR XTAL.
***Note-Reinstall AFCS XTALS***"))
         else
         (assert (repair "***Perform receiver tuning procedure prior to reinstalling crystals***
Replace appropriate RCVR XTAL.
***Note-Reinstall AFCS XTALS***"))))

;;;      DN 14.46


(defrule determine-lo-rcvr-3-state ""
 (working-state radar unsatisfactory)
   (all/checks-state waybad)
    (not (repair ?))
  =>
  (if (yes-or-no-p "Check LO RCVR switch positions (yes/no)? ")
     then
     (assert (lo-rcvr-3-state sat))
        (assert (all/checks-state still bad))    ;;; go to 14.57
     else
        (assert (lo-rcvr-3-state unsat))
        (assert (all/checks-state fubar))))      ;;; go to 14.47
```

```
;;;      DN 14.47


(defrule determine-fault-iso2-state ""
 (working-state radar unsatisfactory)
   (all/checks-state fubar)
    (not (repair ?))
  =>
  (if (yes-or-no-p "Is fault isolation meter pointer sweeping (yes/no)? ")
     then
     (assert (fault-iso2-state sat))
        (assert (all/checks-state tfu))  ;;; go to 14.57
     else
        (assert (fault-iso2-state unsat))
        (assert (all/checks-state snafu))))      ;;; go to 14.50


;;;      DN 14.50



(defrule determine-video-pulse-state ""
 (working-state radar unsatisfactory)
   (all/checks-state snafu)
    (not (repair ?))
  =>
  (if (yes-or-no-p "***Place display in A scan mode***
***Place VIDEO TEST/OFF switch to VIDEO TEST on fwd & aft radar control***
A minimum of 12 long video or 6 short video pulses are visible (yes/no)? ")
     then
     (assert (video-pulse-state sat))
        (assert (all/checks-state snafu))        ;;; go to 14.53
     else
        (assert (repair "Replace RCVR/TRANS and/or Bite module 2A10/7A10."))))


;;;      DN 14.53

(defrule determine-2a5/7a5-state ""
 (working-state radar unsatisfactory)
   (all/checks-state snafu)
    (not (repair ?))
```

```
  =>
  (if (yes-or-no-p "Check 2A5/7A5 at test point 1 for 7.5 +/-2.5V pulse (yes/no)? ")
     then
     (assert (repair "Replace STO XTAL 2A5/7A5 CR 1 & 2 and/or
replace SELF TEST OSC module 2A5/7A5."))
         else
         (assert (repair "Replace SYNC module 1A8."))))

;;;      DN 14.57


(defrule determine-afc-xtal-state ""
 (working-state radar unsatisfactory)
   (all/checks-state still bad)
    (not (repair ?))
  =>
  (if (yes-or-no-p "Check AFC XTAL 1 and 2 switch positions (yes/no)? ")
     then
     (assert (afc-xtal-state sat))
         (assert (all/checks-state tfu))  ;;; go to 14.59
     else
         (assert (repair "Replace applicable AFC XTAL 2A3/7A3 and CR2."))))


;;;      DN 14.59


(defrule determine-last-state ""
 (working-state radar unsatisfactory)
   (all/checks-state tfu)
    (not (repair ?))
  =>
  (if (yes-or-no-p "Check AFC switch positions (yes/no)? ")
     then
     (assert (repair "Replace bite module 2A10/7A10."))
         else
         (assert (repair "***Place AFC MAN/AUTO switch to MANUAL***
***Perform Manual Tune***
If yes, replace RCVR/TRANSMITTER
If no, replace SSLO module 2A4/7A4."))))


(defrule no-repairs ""
```

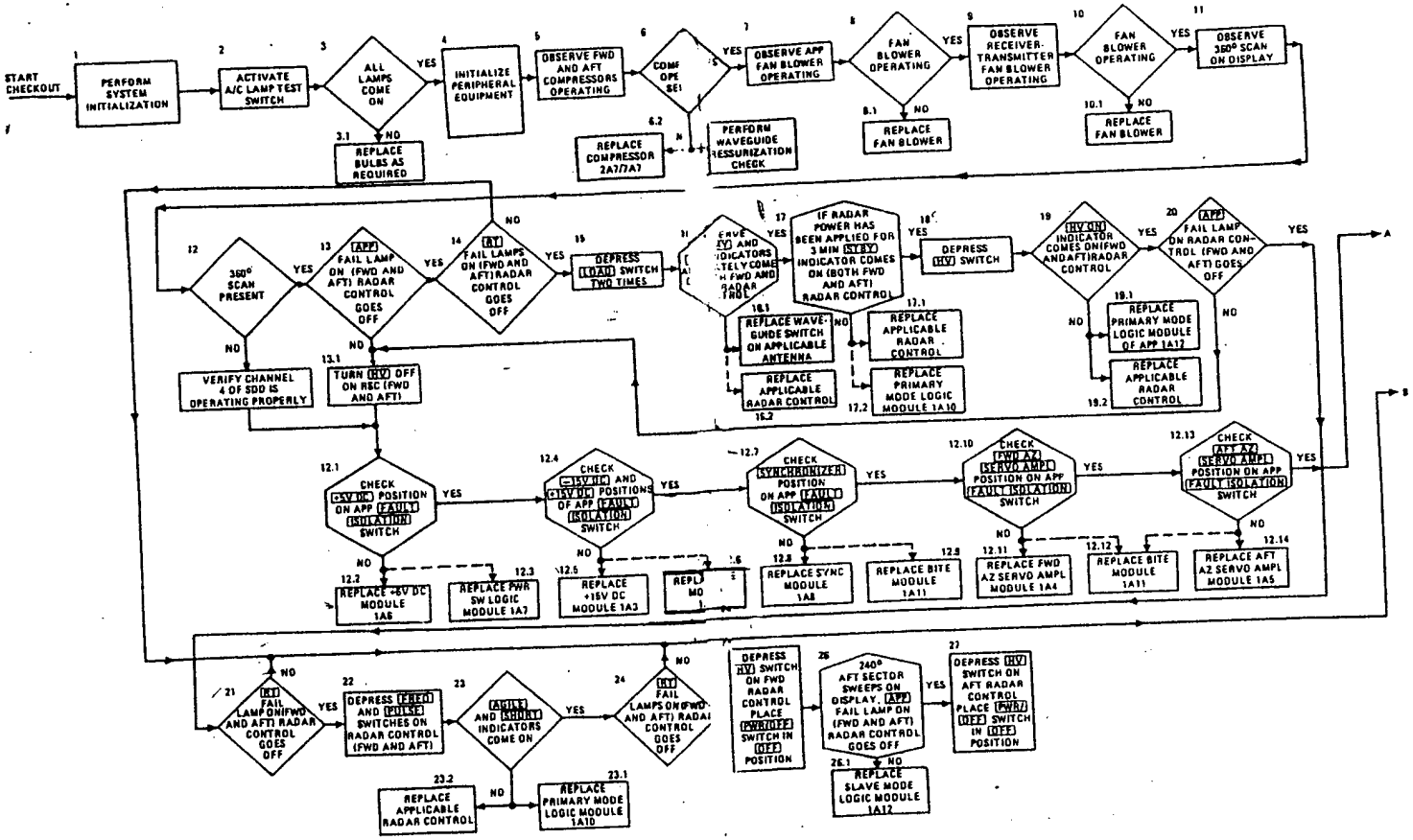
```
  (declare (salience -10))
  (not (repair ?))
  =>
  (assert (repair "Check power supply, attempt to re-initialize.")))

;;;**************************
;;;* STARTUP AND REPAIR RULES *
;;;**************************

(defrule system-banner ""
  (declare (salience 10))
  =>
  (printout t crlf crlf)
  (printout t "The Avionics Diagnostic Advisor Expert System v1.0")
  (printout t crlf crlf))

(defrule print-repair ""
  (declare (salience 10))
  (repair ?item)
  =>
  (printout t crlf crlf)
  (printout t "Suggested Repair:")
  (printout t crlf crlf)
  (format t " %s%n%n%n" ?item))
```

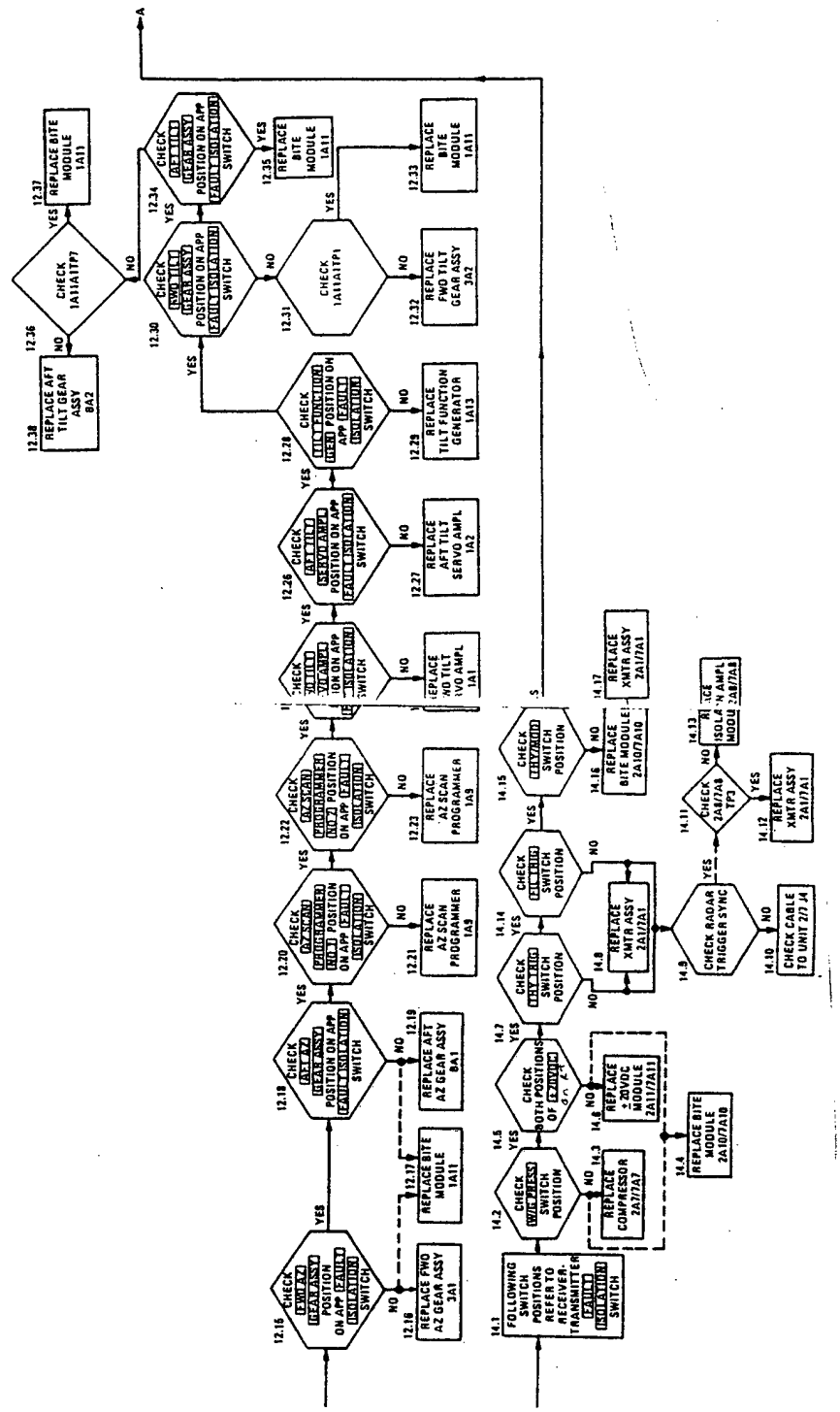
APPENDIX B. APS 115 RADAR SYTEM

The APS 115 radar system is the principal airborne surveillance device for observing and detecting surface vessels, submarines operating with a snorkel, aircraft, and other objects of military significance. The search radar system comprises: (1) two separate, selective, long and short pulse type radar receiver-transmitters, (2) two antennas: one in the aircraft nose radome and one in the tail radome providing 360 degree azimuth coverage, and (3) radar and antenna control panels. Radar search scan and data pick-up is performed by each radar set independently.

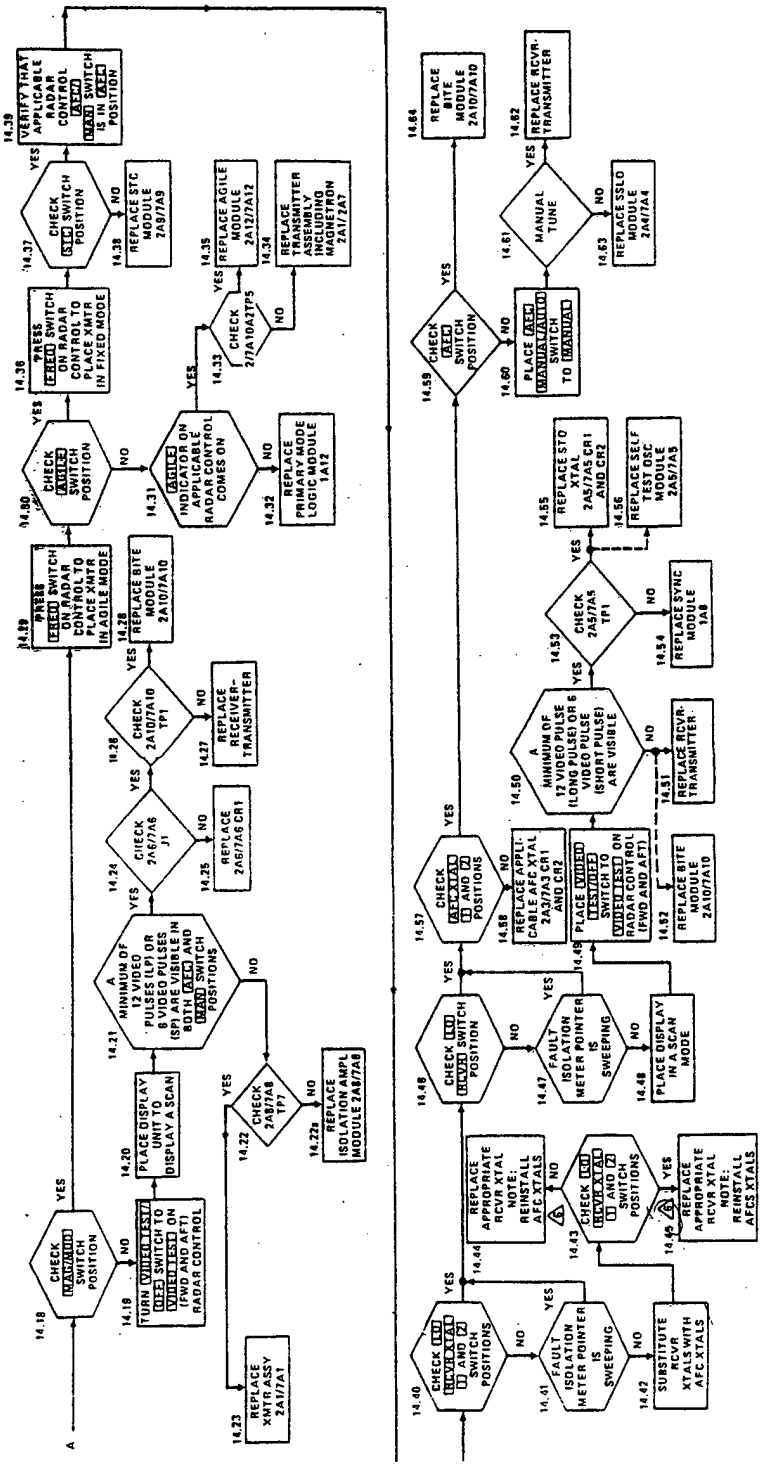
Video data from both radars is combined in the antenna position programmer (APP), routed through the RIU to the SDD dislay. Search radar data can be viewed only at the non-acoustic operator station on the SDD. The display presentation is true-north stabilized, with a computer-generated symbol depicting aircraft true course. Radar operating controls are located at the non-acoustic operator station. (NAVAIR 01-75PAC-1.1)

Fault indications are displayed at the operators station, at the antenna position programmer panel, and at the receiver-transmitter control panel (forward and aft). Detailed fault isolation trees are provided in the crew station manual (NAVAIR 01-75PAC-12-5). A copy of the troubleshooting diagram is provided on the following pages.


START CHECKOUT
PERFORM SYSTEM INITIALIZATION
ACTIVATE A/C LAMP TEST SWITCH
ALL LAMPS COME ON
REPLACE BULBS AS REQUIRED
INITIALIZE PERIPHERAL EQUIPMENT
OBSERVE FWD AND AFT COMPRESSORS OPERATING
REPLACE COMPRESSOR 2A7/7A7
PERFORM WAVEGUIDE PRESSURIZATION CHECK
OBSERVE APP FAN BLOWER OPERATING
FAN BLOWER OPERATING
REPLACE FAN BLOWER
OBSERVE RECEIVER-TRANSMITTER FAN BLOWER OPERATING
FAN BLOWER OPERATING
REPLACE FAN BLOWER
OBSERVE 360° SCAN ON DISPLAY
360° SCAN PRESENT
VERIFY CHANNEL 4 OF SDD IS OPERATING PROPERLY
APP FAIL LAMP ON (FWD AND AFT) RADAR CONTROL GOES OFF
RT FAIL LAMPS ON (FWD AND AFT) RADAR CONTROL GOES OFF
TURN HV OFF ON RSC (FWD AND AFT)
DEPRESS LOAD SWITCH TWO TIMES
IF RADAR POWER HAS BEEN APPLIED FOR 3 MIN STBY INDICATOR COMES ON (BOTH FWD AND AFT) RADAR CONTROL
REPLACE WAVE-GUIDE SWITCH ON APPLICABLE ANTENNA
REPLACE APPLICABLE RADAR CONTROL
REPLACE APPLICABLE RADAR CONTROL
REPLACE PRIMARY MODE LOGIC MODULE 1A10
DEPRESS HV SWITCH
HV ON INDICATOR COMES ON (FWD AND AFT) RADAR CONTROL
APP FAIL LAMP ON RADAR CONTROL (FWD AND AFT) GOES OFF
REPLACE PRIMARY MODE LOGIC MODULE OF APP 1A12
REPLACE APPLICABLE RADAR CONTROL
CHECK +5V DC POSITION ON APP FAULT ISOLATION SWITCH
REPLACE +5V DC MODULE 1A6
REPLACE PWR SW LOGIC MODULE 1A7
CHECK -15V DC AND +15V DC POSITIONS OF APP FAULT ISOLATION SWITCH
REPLACE +15V DC MODULE 1A3
CHECK SYNCHRONIZER POSITION ON APP FAULT ISOLATION SWITCH
REPLACE SYNC MODULE 1A8
REPLACE BITE MODULE 1A11
CHECK FWD AZ SERVO AMPL POSITION ON APP FAULT ISOLATION SWITCH
REPLACE FWD AZ SERVO AMPL MODULE 1A4
REPLACE BITE MODULE 1A11
CHECK AFT AZ SERVO AMPL POSITION ON APP FAULT ISOLATION SWITCH
REPLACE AFT AZ SERVO AMPL MODULE 1A5
RT FAIL LAMP ON (FWD AND AFT) RADAR CONTROL GOES OFF
DEPRESS FREQ AND PULSE SWITCHES ON RADAR CONTROL (FWD AND AFT)
AGILE AND SHORT INDICATORS COME ON
REPLACE APPLICABLE RADAR CONTROL
REPLACE PRIMARY MODE LOGIC MODULE 1A10
RT FAIL LAMPS ON (FWD AND AFT) RADAR CONTROL GOES OFF
DEPRESS HV SWITCH ON FWD RADAR CONTROL PLACE PWR/OFF SWITCH IN OFF POSITION
240° AFT SECTOR SWEEPS ON DISPLAY. APP FAIL LAMP ON (FWD AND AFT) RADAR CONTROL GOES OFF
REPLACE SLAVE MODE LOGIC MODULE 1A12
DEPRESS HV SWITCH ON AFT RADAR CONTROL PLACE PWR/OFF SWITCH IN OFF POSITION
YES
NO
A
B



12.15 CHECK FWD AZ GEAR ASSY POSITION ON APP FAULT ISOLATION SWITCH
12.16 REPLACE FWD AZ GEAR ASSY 3A1
12.17 REPLACE BITE MODULE 1A11
12.18 CHECK AFT AZ GEAR ASSY POSITION ON APP FAULT ISOLATION SWITCH
12.19 REPLACE AFT AZ GEAR ASSY 8A1
12.20 CHECK AZ SCAN PROGRAMMER NO 1 POSITION ON APP FAULT ISOLATION SWITCH
12.21 REPLACE AZ SCAN PROGRAMMER 1A9
12.22 CHECK AZ SCAN PROGRAMMER NO 2 POSITION ON APP FAULT ISOLATION SWITCH
12.23 REPLACE AZ SCAN PROGRAMMER 1A9
12.26 CHECK AFT TILT SERVO AMPL POSITION ON APP FAULT ISOLATION SWITCH
12.27 REPLACE AFT TILT SERVO AMPL 1A2
12.28 CHECK TILT FUNCTION GEN POSITION ON APP FAULT ISOLATION SWITCH
12.29 REPLACE TILT FUNCTION GENERATOR 1A13
12.30 CHECK FWD TILT GEAR ASSY POSITION ON APP FAULT ISOLATION SWITCH
12.31 CHECK 1A11A1TP1
12.32 REPLACE FWD TILT GEAR ASSY 3A2
12.33 REPLACE BITE MODULE 1A11
12.34 CHECK AFT TILT GEAR ASSY POSITION ON APP FAULT ISOLATION SWITCH
12.35 REPLACE BITE MODULE 1A11
12.36 CHECK 1A11A1TP7
12.37 REPLACE BITE MODULE 1A11
12.38 REPLACE AFT TILT GEAR ASSY 8A2
14.1 FOLLOWING SWITCH POSITIONS REFER TO RECEIVER-TRANSMITTER FAULT ISOLATION SWITCH
14.2 CHECK W/G PRESS SWITCH POSITION
14.3 REPLACE COMPRESSOR 2A7/7A7
14.4 REPLACE BITE MODULE 2A10/7A10
14.5 CHECK BOTH POSITIONS OF ±20VDC
14.6 REPLACE ±20VDC MODULE 2A11/7A11
14.7 CHECK TILT FWD SWITCH POSITION
14.8 REPLACE XMTR ASSY 2A1/7A1
14.9 CHECK RADAR TRIGGER SYNC
14.10 CHECK CABLE TO UNIT 2/7 J4
14.11 CHECK 2A8/7A8 TP3
14.12 REPLACE XMTR ASSY 2A1/7A1
14.13 REPLACE ISOLATION AMPL MODULE 2A8/7A8
14.14 CHECK TILT FWD SWITCH POSITION
14.15 CHECK TILT FWD SWITCH POSITION
14.16 REPLACE BITE MODULE 2A10/7A10
14.17 REPLACE XMTR ASSY 2A1/7A1
YES
NO
A


14.18 CHECK MAG/AUDIO SWITCH POSITION
YES
NO
14.19 TURN VIDEO TEST/OFF SWITCH TO VIDEO TEST ON (FWD AND AFT) RADAR CONTROL
14.20 PLACE DISPLAY UNIT TO DISPLAY A SCAN
14.21 A MINIMUM OF 12 VIDEO PULSES (LP) OR 8 VIDEO PULSES (SP) ARE VISIBLE IN BOTH AFC AND MAN SWITCH POSITIONS
14.22 CHECK 2A8/7A8 TP7
14.22a REPLACE ISOLATION AMPL MODULE 2A8/7A8
14.23 REPLACE XMTR ASSY 2A11/7A1
14.24 CHECK 2A6/7A6 J1
14.25 REPLACE 2A6/7A6 CR1
14.26 CHECK 2A10/7A10 TP1
14.27 REPLACE RECEIVER-TRANSMITTER
14.28 REPLACE BITE MODULE 2A10/7A10
14.29 PRESS FIXED SWITCH ON RADAR CONTROL TO PLACE XMTR IN AGILE MODE
14.30 CHECK AGILE SWITCH POSITION
14.31 AGILE INDICATOR ON APPLICABLE RADAR CONTROL COMES ON
14.32 REPLACE PRIMARY MODE LOGIC MODULE 1A12
14.33 CHECK 2/7A10A1TP5
14.34 REPLACE TRANSMITTER ASSEMBLY INCLUDING MAGNETRON 2A11/2A7
14.35 REPLACE AGILE MODULE 2A12/7A12
14.36 PRESS FIXED SWITCH ON RADAR CONTROL TO PLACE XMTR IN FIXED MODE
14.37 CHECK AGILE SWITCH POSITION
14.38 REPLACE STC MODULE 2A9/7A9
14.39 VERIFY THAT APPLICABLE RADAR CONTROL AFC/MAN SWITCH IS IN AFC POSITION
14.40 CHECK RCVR XTAL 1 AND 2 SWITCH POSITIONS
14.41 FAULT ISOLATION METER POINTER IS SWEEPING
14.42 SUBSTITUTE RCVR XTALS WITH AFC XTALS
14.43 CHECK RCVR XTAL 1 AND 2 SWITCH POSITIONS
14.44 REPLACE APPROPRIATE RCVR XTAL NOTE: REINSTALL AFC XTALS
14.45 REPLACE APPROPRIATE RCVR XTAL NOTE: REINSTALL AFC'S XTALS
14.46 CHECK RCVR XTAL 1 AND 2 SWITCH POSITIONS
14.47 FAULT ISOLATION METER POINTER IS SWEEPING
14.48 PLACE DISPLAY IN A SCAN MODE
14.49 PLACE VIDEO TEST/OFF SWITCH TO VIDEO TEST ON RADAR CONTROL (FWD AND AFT)
14.50 A MINIMUM OF 12 VIDEO PULSE (LONG PULSE) OR 6 VIDEO PULSE (SHORT PULSE) ARE VISIBLE
14.51 REPLACE RCVR-TRANSMITTER
14.52 REPLACE BITE MODULE 2A10/7A10
14.53 CHECK 2A5/7A5 TP1
14.54 REPLACE SYNC MODULE 1A8
14.55 REPLACE STO XTAL 2A5/7A5 CR1 AND CR2
14.56 REPLACE SELF TEST OSC MODULE 2A5/7A5
14.57 CHECK AFC XTAL 1 AND 2 POSITIONS
14.58 REPLACE APPLI-CABLE AFC XTAL 2A3/7A3 CR1 AND CR2
14.59 CHECK AFC SWITCH POSITION
14.60 PLACE AFC/MANUAL TUNE SWITCH TO MANUAL
14.61 MANUAL TUNE
14.62 REPLACE RCVR-TRANSMITTER
14.63 REPLACE SSLO MODULE 2A4/7A4
14.64 REPLACE BITE MODULE 2A10/7A10
A

LIST OF REFERENCES

Chorafas, D.N., *Knowledge Engineering*, Van Nostrand Reinhold, New York, New York, 1990.

Forsysth, G. F., Larkin, M. D., and Wallace, G. A., "Verification of Heuristic Diagnostic Knowledge by Comparison with a Causal/Qualitative Model," *ACM*, 1990.

Frenzel, L. E., *Crash Course in Artificial Intelligence and Expert Systems*, Howard W. Sams & Co., Indianapolis, Indiana, 1987.

Ghandi, O. P., Wadhwa, S., Chugh, V., and Vaze, S. C., "Development and Implementation of an Expert System for Automotive Troubleshooting," *Engineering Systems and Analysis*, Volume 8: Part B, ASME 1994.

Lewis, C. D., *Development of a Maintenance Advisor Expert System for the Mk 92 Mod 2 Fire Control System,* Masters Thesis, Naval Postgraduate School, Monterey, California, September 1993.

NAVAIR 01-75PAC-1.1, *P-3C NATOPS Manual*, U. S. Navy, August 1994.

NAVAIR 01-PAC-12-5, *P-3C Crewstation Manual*, U. S. Navy, February 1995.

Olson, D. L., and Courtney, J. F., Decision Support Models and Expert Systems, MacMilan Publishing Company, New York, New York, 1992.

OPNAV 4790.2E, Naval Aviation Maintenance Program, U. S. Navy, July 1992.

Prerau, D. S., Developing and Managing Expert Systems: proven Techniques for Business and Industry, Addison-Wesley Publishing Company Inc., Menlo Park, California, 1990.

Rasmussen, J., and Jensen, A., "Mental Procedures in Real-Life Tasks: A Case Study of Electronic Trouble Shooting," *Ergonomics*, v. 17, no. 3, 1974.

INITIAL DISTRIBUTION LIST

		No. copies
1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Library, Code 13 Naval Postgraduate School Monterey, California 93943-5101	2
3.	Professor Hemant Bhargava, Code SM/Bh Naval Postgraduate School Monterey, California 93943-5101	2
4.	Donald R. Eaton, (RADM, USN, ret.), Code SM/Et Naval Postgraduate School Monterey, California 93943-5101	2
5.	Lt. John N. Lewis COMPATWING TEN 356 N. Charles Porter Ave. NAS Whidbey Island, Oak Harbor, Washington 98278	1