15000 COUNTY ROAD 5 BURNINGWELL, MN 55007 USA
TEL 512 435 7887 FAX 512 435 7887 TLM 512000000

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file - Votre référence

Our file - Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**Model Based Reasoning
Applied to the Diagnosis of
Spacecraft Electrical Power System Faults**

By

Peter J. Adamovits, BAsC

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for a degree of
Master of Engineering

**Department of Mechanical and Aerospace Engineering
Ottawa-Carleton Institute
for Mechanical and Aerospace Engineering**

Carleton University

3 July 1992

© copyright

1992, Peter J. Adamovits

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Votre titre / Votre référence

Ouvrage / Autre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-79782-7

Canada

The undersigned hereby recommends to the Faculty of Graduate
Studies and Research acceptance of the thesis,

**Model Based Reasoning Applied to the Diagnosis of
Spacecraft Electrical Power System Faults**

Submitted by Peter Joseph Adamovits, B.A.Sc.P.Eng, in partial fulfilment of the
requirements for the degree of Master of Engineering

B. Pagurek
Thesis Supervisor
Department of Systems and Computer Engineering

R. Bell
Chairman
Department of Mechanical and Aerospace Engineering

**Department of Mechanical and Aerospace Engineering
Ottawa-Carleton Institute
for Mechanical and Aerospace Engineering**

July 1992

Abstract

Model-Based Artificial Intelligence software approaches to diagnosis require encoding a reasonable facsimile of the problem domain. In the work presented, the model is encoded as classical engineering simulation. (continuous, linear, discrete event) of the domain, a spacecraft electrical power subsystem. The fault detection and portions of the diagnostic reasoning system thus become comparators between the expected behaviour (that of the model) and the application domain under analysis. The diagnostic problem is partitioned into 6 discrete steps that include: fault detection, diagnosis and hypothesis generation, hypothesis space pruning, validation of the hypothesis, test case selection and verification of the diagnosis. The system performs a simplified form of learning by injecting diagnosed faults into the model to keep it consistent with the domain reducing further unnecessary processing in the future. We demonstrate that largely shallow reasoning systems performing a comparative analysis of the data is sufficient for a complex application.

Acknowledgements

I would like to acknowledge several people and organizations who provided guidance and support over the years leading to the development of this and related works. Of key importance to the development of any *real* AI system is a representative application domain. The Space Based Radar (SBR) Electrical Power System (EPS) was identified as an appropriate domain on which to demonstrate the diagnostic system approach of this work. The development of this system using SBR as an application became possible only through the cooperation of Craig Maskell (SBR EPS Project Manager) of the Canadian Department of National Defense and Louis Piche of CAL Corporation. Without their support only another *toy* problem domain would have been possible. While the SBR Program provided the EPS design details, considerable experiential knowledge in spacecraft operations was provided through numerous informal discussions with David Andeen of the Department of Communications and by Ed Fuchs and Barry Turner of Telesat Canada over a several year period. At Carleton University, I would like to thank Dr. Kind of Aeronautical and Mechanical Engineering for provided me with guidance at a time when the world appeared to be quite chaotic and Dr. B. Pagurek of Systems and Computer Engineering for being a patient supervisor over the several years of part time study that the work took to reach fruition. Finally, I would like to thank the management of the Canadian Space Agency who provided me with sufficient time and resources to finally complete the work of this thesis.

Table of Contents

Abstract	iii
Acknowledgements	iv
List of Illustrations	x
List of Appendices	xi
Glossary of Terms	xii
1.0 Introduction	1
2.0 Spacecraft and the Space Domain	8
2.1 Mission Requirements and Orbit Mechanics	8
2.2 Operational Requirement for Electrical Power System	10
2.3 Electrical Power System Overview	12
2.4 Failure Mode and Complexity Analysis of the EPS Domain	16
2.4.1 Multiple Fault Complexity Analysis	19
3.0 The Diagnostic Problem	26
3.1 Fault Detection	35
3.2 Diagnosis and Hypothesis Generation	39
3.3 Hypothesis Space Pruning	43

3.4 Hypothesis Validation	44
3.5 Test Case Selection	46
3.6 Hypothesis Verification	47
4.0 Artificial Intelligence Based Diagnosis	48
4.1 Rule Based Systems	49
4.1.1 Multiple Fault Diagnostic System Rule Bases	52
4.2 Model Based Reasoning Systems	55
4.2.1 Model Based Reasoning in the Multiple Fault Diagnostic System	58
4.3 Qualitative Reasoning Systems	61
4.3.1 Qualitative Reasoning in the Multiple Fault Diagnostic System	62
4.4 Control of the Reasoning Process	63
4.4.1 Control Mechanisms in the Multiple Fault Diagnostic System	66
5.0 Multiple Fault Diagnostic System Design	67
5.1 Data Acquisition System	76
5.2 Fault Detection System	78
5.2.1 Qualitative Fault Detection	79
5.2.2 Quantitative Fault Detection	82
5.2.3 A Priori Knowledge Base	85
5.3 Diagnosis and Hypothesis Generation System	86
5.4 Hypothesis Space Pruning System	90
5.5 Hypothesis Validation System	92

5.6 Test Case Selection System	93
5.7 Hypothesis Verification System	95
5.8 Reasoning Control Mechanism	97
5.9 The Electrical Power System Model	99
 6.0 Testing the Multiple Fault Diagnostic System	 103
 7.0 Analysis and Discussion of the Test Results	 109
7.1 Limitations of the Application Domain	114
7.2 Notes on the Implementation Environment	115
7.3 Notes on the Implementation	118
 8.0 Conclusions	 121
 References	 125

List of Tables

Table 2-1 Single Fault Complexity Analysis of the EPS Components and Faults.	18
Table 2-2 Single Fault Complexity Analysis of a Tile.	19
Table 2-3a Fault Masking due to Intermediate Conditioner Faults	22
Table 2-3b Fault Masking due to Transmit Receive Conditioner Faults	22
Table 5-1 Qualitative Fault Detection	80
Table 5-2 Improvement due to Focus of Attention	98
Table 6-1 Single Fault Test Summary	107
Table A-1 Operational States of Power Distribution Components	139
Table A-2 Operational States of Energy Storage Unit Components	140
Table A-3 Operational States for the Solar Panel and Primary Power Conditioner	140
Table A-4 Auxiliary Conditioner State/Parameter Relationships	141
Table A-4 ESU State/Parameter Relationships ...	142
Table A-5 Cell Parameter Relationships	143
Table A-6 Cell State/Parameter Relationships	144
Table A-7 Cell State/Parameter Relationships: Shorted Cell Model	145
Table A-8 Cell Pack Parameters	145
Table A-9 Intermediate Conditioner State/Parameter Relationships	147
Table A-10 Power Distribution Parameters	148
Table A-11 Primary Power Conditioning Unit Parameters	149
Table A-12 Radar Load State/Parameter Relationships	150
Table A-13 Auxiliary Conditioner State/Parameter Relationships	153
Table B-1a Multiple Fault Diagnostic System Tests	168
Table B-1b Multiple Fault Diagnostic System Tests (cont'd)	169

List of Illustrations

Figure 2-1 Electrical Power System Block Diagram	14
Figure 2-2 Power Distribution Components Fault Tree	24
Figure 3-1 The Diagnostic Process	34
Figure 3-2 Qualitative Fault Detection Equivalent Circuit	38
Figure 3-3 Quantitative Fault Detection Equivalent Circuit	38
Figure 3-4 Signal and References versus Time	38
Figure 5-1 The Model Based Diagnostic Process	68
Figure 5-2 Multiple Fault Diagnostic System Browser	70
Figure 5-3 Electrical Power System Domain Model Browser	71
Figure 5-4 Electrical Power System Domain Model Browser Text Window	72
Figure 5-5 Intermediate Conditioner Popup Window	73
Figure 5-6 Cell Pack Popup Window	73
Figure A-1 Auxiliary Conditioner Characteristic Curve	141
Figure A-2 Energy Storage Unit, Cell Pack and Cell Characteristic Curve	143
Figure A-3 Intermediate Conditioner Characteristic Curves	146
Figure A-4 Primary Power Conditioning Unit Characteristic Curves	149
Figure A-5 Radar Load Characteristic Curve	151
Figure A-6 Solar Panel Characteristic Curves	152
Figure A-7 Transmit Receive Conditioner Characteristic Curve	154
Figure B-1 Transmit Receive Conditioner Testing	164

List of Appendices

Appendix A Detail Description of the Electrical Power System Domain

Model 131

Appendix B Tests Performed on the Multiple Fault Diagnosis System 160

Glossary of Terms

#symbolName - defines a Smalltalk or Prolog symbol of the name symbolName. In Digitalk's Smalltalk Prolog [Digitalk 1988b] "#symbolName" is equivalent to a standard Prolog [Clocksin and Mellish 1981] atomic constant "symbolName". In standard Prolog this would normally be written as "symbol_name" (Smalltalk however does not permit the character "_"). Additionally, symbols starting with an upper case character are viewed as a system wide global.

actual data - data received from the domain in the form of telemetry or command frames. Also all sensor readings from the electrical components within these frames.

APriori - the Smalltalk/Prolog class that defines the component inter-connectivity model

AuxCond - the Smalltalk class that defines the auxiliary conditioner

auxiliary conditioner - a power conditioner that provides power to secondary spacecraft loads (e.g. thermal control, attitude and orbit control, telemetry tracking and control, etc.). Receives power from the an intermediate conditioner.

bus - *spacecraft*: the complete spacecraft (Electrical Power System, Structure, Attitude and Orbit Control, etc.) with the exclusion of the payload systems. *Electrical Power System*: the electrical connections (wires and cabling) that connect major EPS components.

candidate - a component or hypothesis that is subject of the focus of attention at this time.

Cell - the Smalltalk class that defines the cell.

cell - a single battery cell.

CellPack - the Smalltalk class that defines the cell pack.

cell pack - a group of 20 cells connected in series.

component - *Electrical*: a discrete element of a circuit that is modeled in the software simulation. *Software*: a software module that addresses one complete aspect of the solution; a Smalltalk class or a group of Prolog clauses identified grouped into a single Smalltalk class.

deep knowledge - knowledge of the problem at hand (e.g. diagnosis) that includes

specific operational set points that will be used for reference for comparison in determining the outcome. In terms of the MFDS this knowledge is found within the model and certain knowledge bases (DiagRules)

diagnosis and hypothesis generation - the process where by symptoms in the form of data and faults detected are related to failure modes and one or more hypotheses describing this relationship are postulated.

DiagnosisSystem - the Smalltalk class that implements the diagnosis and hypothesis generation system component of the Diagnostic Process.

Diagnostic Process - a process that identifies and breaks down the constituent steps in diagnosis. Includes: fault detection, diagnosis and hypothesis generation, hypothesis space pruning, validation, test case selection and verification.

DiagRules - the Smalltalk/Prolog knowledge base that implements the DiagnosticSystem reasoning process.

domain - the electrical power system and constituent components, environmental factors that influence the EPS (i.e. solar eclipse, radiation damage to components, etc.), operational factors (radar duty cycle, battery reconditioning, etc.), others.

Electrical Power System - the application to which the diagnostic system is to be applied. Includes the constituent electrical components of the electrical power system: cells, cell packs, ESU, solar panel, primary power conditioner, intermediate conditioners, transmit receive conditioners, auxiliary conditioners and radar loads.

Energy Storage Unit - the battery of the EPS. Comprised of 5 series connected cell packs.

EPS - see Electrical Power System.

ESU - the Smalltalk class that defines the Energy Storage Unit.

expected data - data received from the model in the form of a telemetry frame. Also all sensor readings from the electrical components of the model within this telemetry frame.

FaultDetectionSystem - the Smalltalk class that implements the fault detection system. Includes qualitative and quantitative fault detection components.

fault detection - detecting the existence of an anomaly in the behaviour of the system in question without necessarily identifying the cause or significance of the fault.

hypothesis - a single conclusion drawn from an inferential process based on an analysis of available data and behavioural relationships. Included in fault hypotheses, diagnosis hypotheses, pruned hypotheses etc.

hypothesis space pruning - the action of reducing the number of hypotheses by filtering based on several criteria criticality of failure, strength of supporting evidence, mean time between failures etc.

IntCond - the Smalltalk class that implements the intermediate conditioner.

intermediate conditioner - a component of the EPS. An electrical power conditioner supplying power to two transmit receive conditioners and an auxiliary conditioner.

is(return, smalltalkMessage) - Prolog statement used to pass control to Smalltalk which invokes 'smalltalkMessage'. Binds the result of the Smalltalk message send to the variable return. Nonstandard Prolog form used in Digitalk's Smalltalk V/286 based Prolog [Digitalk 1988b].

load - see radar load.

meta level reasoning - a system that reasons about the inferential processes of lower level reasoning systems. This includes reasoning about identifying and applying the appropriate solution methodology rather than operating on the problem data directly. A supervisory or control mechanism that performs reasoning.

MetaLevel - the Smalltalk class that performs the analysis of the validation tests performed.

MetaRules - the Smalltalk/Prolog class that performs the reasoning component of the MetaLevel system.

MFDS - Multiple Fault Diagnosis System. Includes a software system encoded in Smalltalk and Prolog that implements the Diagnostic Process defined in this thesis and demonstrated on a Spacecraft Electrical Power System.

model - an encoded description of the target domain in the form of behavioural descriptions, logical relationships, etc. *In the context of this work a software simulation* of the application domain similar to those developed by engineers as part of a parametric analysis of a design.

PowerDist - the Smalltalk class that implements the power distribution pseudo component.

power distribution - the pseudo component that represented the composite of all

intermediate conditioners and lower level EPS components.

Ppcu - the Smalltalk class that implements the Primary Power Conditioning Unit.

Primary Power Conditioning Unit - an electrical power conditioner that regulates the output of the solar panel.

Prolog - a non-procedural language where logic statements are used to encode relationships and data. [Clocksin and Mellish 1981]

Prolog clause - a logical statement relating one or more antecedents to one consequent. The basic logic statement or predicate used in the Prolog language.

pruning - see hypothesis space pruning.

PruningRules - the Smalltalk/Prolog class that performs the PruningSystem reasoning process.

PruningSystem - the Smalltalk class that performs hypothesis space pruning.

pseudo component - an EPS model component defined for conceptual or programming convenience rather than existing in the EPS domain.

qualitative data - data that identifies the qualitative change in the data from the previous value to the present value. Indicated as a tri state logical value increasing, no change or decreasing, indicated as (+ 0 -) respectively.

qualitative fault detection - performing fault detection while reasoning with qualitative values of the data only.

QualitativeFaultDetection - the Smalltalk class that implements the qualitative fault detection system.

QualRules - the Smalltalk/Prolog class that contains the Prolog knowledge base that performs the qualitative reasoning process.

quantitative fault detection - performing fault detection while reasoning with numeric and logical (i.e. binary but not qualitative) values of the data only.

QuantitativeFaultDetection - the Smalltalk class that implements the quantitative fault detection system.

QuanRules - the Smalltalk/Prolog class that contains the Prolog knowledge base that performs the quantitative reasoning process.

radar load - a single RF transmit receive element of a phased array antenna.

RadarLoad - the Smalltalk class that implements the radar load.

shallow knowledge - a direct symptom/hypothesis reasoning step based on experiential knowledge rather than deep knowledge. May include numeric set points but only those that can be arrived at through minimal knowledge of how the component operates. For example: a tripped circuit breaker can be expected to interrupt current flow causing it to drop to zero.

Smalltalk - an object oriented message passing language having a class / subclass object hierarchy.

solar panel - an array of photovoltaic cells acting as an electrical power source

SolarPanel - the Smalltalk class that implements the solar panel.

spacecraft - a man made object that may orbit celestial bodies (planets, moons, the sun, etc.) or travel through space on a non-orbital path.

symptom - any sensory data that indicates unexpected behaviour.

telemetry - an array of data containing sensor and status values to be used in observing the operational health of the remote system. Any data frame similar to that which would be received from a spacecraft.

test case selection - the process of identifying an appropriate test case to be used in verification when the hypothesis is tested against the domain. The test case is to be constructed so that additional supporting or refuting evidence for the candidate hypothesis will be made available through its implementation.

TestRules - the Smalltalk/Prolog class that performs the reasoning associated with test case selection.

TestSystem - the Smalltalk class that performs test case selection.

tile - an EPS subassembly that contains one intermediate conditioner, two transmit receive conditioners, an auxiliary conditioner and eight radar loads. These components are connected as shown in figure 2-1.

tm - see telemetry.

transmit receive conditioner - a component of the EPS. An electrical power conditioner that supplies power to attached 4 radar loads and receives power from an

intermediate conditioner.

TrCond - the Smalltalk class that defines the transmit receive conditioner.

validation - the action where by a hypothesized fault is confirmed plausible through testing it against the model of the domain.

verification - the action where by a hypothesized and previously validated fault is confirmed true through testing it against the application domain.

ValidationSystem - the Smalltalk class that performs hypothesis validation.

VerificationRules - the Smalltalk/Prolog class that implements the verification reasoning process.

VerificationSystem - the Smalltalk class that implements the verification system.

1.0 Introduction

Diagnosis is a fact finding exercise that attempts to determine the underlying events that cause unexpected behaviour in systems. The stimulus that causes the change may arise from a variety of sources both external to and within the system under study. Here we are concerned with the identification of one or more failed or degraded components that may exist within the system, not external events that may have caused these failures to occur. The system behaviour is observed through sensors compared with known standards or references. The unexpected behaviour observed is analyzed in an attempt to determine the failed component.

Diagnosis is performed in all aspects of day to day life from discovering why a light bulb will not illuminate through to determining the cause of medical ailments. In the medical field the most widely known automated diagnostic system is Mycin [Shortliffe 1976]. Diagnostic techniques are as varied as the application domains to which the Diagnostic Process is to be applied. The solution methodology must be appropriate for the given class of problem. The diagnostic approach described in this document is demonstrated on a Spacecraft Electrical Power System (EPS). It is postulated that the approach outlined is appropriate in any domain (though typically engineering applications) where an accurate behavioural simulation can be developed.

The motivation for automating the diagnostic process is to increase the accuracy and speed of the diagnosis of the EPS and its components and to reduce the costs associated with traditional means. As spacecraft and space systems become larger and more complex, the demands on a traditionally structured ground-based operation to monitor and control the various subsystems increase dramatically. In his review of NASA projects

where AI based systems are used to monitor and control electrical power systems, Weeks [1987] notes that America's first space station 'Skylab' employed an eight kilowatt power bus and required the attention of a team of fifteen to twenty controllers to monitor the health and loading of the EPS. Clearly the costs associated with such an approach for each of the many subsystems on a spacecraft is prohibitive.¹

The EPS of an earth imaging radar spacecraft will be used to demonstrate the techniques presented in this thesis. The demonstration EPS has a complexity similar to that of the Skylab. The Space Based Radar (SBR) spacecraft EPS [Eatock 1990; Moodie and Maskell 1989] will provide the basis for much of the EPS design in the demonstration diagnostic system. The SBR EPS however, is presently a research activity and is several years away from possible program approval and eventual spacecraft design and launch. In areas where information is not available for the SBR EPS, the missing data will be drawn from that available for the RADARSAT spacecraft program [Adamovits and Jackson 1991; Adamovits 1989]. While the RADARSAT spacecraft is less complex and smaller than that presently envisioned for SBR², its operational environment and mission are very similar in nature. RADARSAT's design is now fixed as the spacecraft enters a manufacture and assembly phase prior to a 1994 launch. In this and later sections, a generic spacecraft EPS based on SBR and RADARSAT will be presented and used to provide the domain for the work of this thesis.

Radar payload spacecraft place further demands on the electrical power system that are significantly different than those of other spacecraft (or space stations). The Electrical

¹ The costs associated with a human based monitor and control system include: accuracy of the diagnosis, time available to respond, operator training, operations facilities and others.

² The SBR spacecraft EPS is expected to have one order of magnitude greater complexity and component count than RADARSAT.

Power System for radar spacecraft must deal with a highly inclined low earth orbit spacecraft with a radar payload having tens of thousands of RF transmit / receive elements demanding between 5 and 30 Kw of electrical power [Eatock 1990]. In the example presented, communications opportunities may be infrequent and of short duration; an eclipse may occur each orbit; the system is expected to operate with failed or degraded components; and short term power demands of the radar payload will exceed the nominal power output of the solar panels.

The diagnostic system described provides a mechanism to detect and diagnose faults that exist in the radar payload spacecraft EPS. The design of the diagnostic system is tailored to suit the EPS domain. Several aspects of the EPS domain influence the design of a fault detection and diagnostic system [Moodie and Maskell 1989]. These include:

- The EPS is comprised of more than 1500 discrete components. The system is complex and is expected to be difficult for control personnel to track.
- Sensor data is available from specific fixed electrical circuit locations only. Testing the circuit elements through probing actions is not possible.
- All data that is available, is provided at frequent time intervals (say 1 second for geosynchronous spacecraft) through a telemetry stream.
- Many spacecraft have frequent periods when contact is not possible (e.g. low earth orbit, high inclination orbits, deep space exploration).³
- Data may be erroneous (due to both telemetry RF interference and sensor failures).
- All operations are performed remotely.

³ Contact may be increased in many cases through the use of multiple ground stations situated at several locations around the earth or through the use of relay spacecraft.

- Replacement of degraded or failed components is not possible in orbit.⁴
- The system can be configured to isolate faulty components from influencing the remaining operational circuit elements.⁵

The diagnostic system presented in this thesis addresses multiple concurrent faults that may exist in the domain. To address the unique requirements of the spacecraft EPS the fault detection/diagnosis system presented relies on a Model (in the form of a software simulation) of the domain as a key element in the reasoning process. The model performs two roles. Initially, the model provides reference data to which the spacecraft data is compared during fault detection. The model also provides a test environment where filtered and ordered (pruned) hypotheses resulting from the fault detection and, diagnosis and hypothesis generation are validated. Successfully validated candidates are then tested (verified) in the domain system. Successful results of this second test (verification) will indicate not only how the model differs from the domain but how the model may be modified to be a better representation of the domain. Modifying the model in this way is a simplified explanation based learning system. Modifications to the model may however lead to divergence between the model and the domain. The correction of a misdiagnosis can be handled through specialized rules as more data becomes available. This is demonstrated in later sections (see discussion of the `breakerResetFailure` test in section 7.0).

⁴ This is the case with present day spacecraft. Proposed designs for the next generation of spacecraft include on orbit replaceable units (ORU). ORU replacement at present is not economically feasible for high altitude (i.e. geostationary) nor possible for planetary exploration spacecraft. A high degree of reliability and configurability thus must be built into present day spacecraft. This is expected to remain the case for the majority of spacecraft for the foreseeable future.

⁵ This is accomplished through fuses, circuit breakers, switches and relays. Several of these transitions are one way only (fuses and some switches), others may be reset (circuit breakers, switches and relays).

The reasoning systems used in the demonstration system contain largely shallow knowledge of the domain. That is, the knowledge structures (or rules) encode a minimum of hard engineering design details of the domain. Shallow based reasoning systems generally encode experiential knowledge of the domain. An example of the shallow reasoning system approach taken in this thesis is the fault detection system. In this system the fault detection rules compare the actual (domain) data to the expected data *within a model separate from the rule*. A deep reasoning system rule in contrast would compare the actual (domain) data with data contained *within the rule*. In the first case the *shallow rule accesses deep knowledge* located external to the rule and in the design presented in this thesis, knowledge located within the model. In the second case the *rule structure itself contains the deep knowledge*.⁶ In the demonstration system, deep knowledge rules represent less than 3% of the total. The diagnosis and hypothesis generation system contain most of these deep knowledge rules with approximately 10% of this rule base containing deep knowledge rules.

The domain system and the model are both, in the case of the demonstration system, simulations of the spacecraft electrical power system. The model and domain are completely independent. Although they are two instances of the same code, the model and domain are manipulated as a software simulation and a less flexible hardware circuit respectively. In practice, the simulation of the spacecraft EPS would be replaced by the actual on-orbit spacecraft system. The simulation that represents the model remains a necessary part of the Multiple Fault Diagnostic System (MFDS).

The objective of this thesis is to demonstrate *model based diagnosis* where the model is

⁶ It is recognized that several structures may be used in AI systems to represent deep knowledge these include: frames, logic statements, cases, and others. In this thesis a simulation is used instead.

a *software simulation* of the domain of interest. The software simulation contains most (although not all) deep knowledge in the system. It is postulated that this is a more natural location for such knowledge for AI applications in engineering systems. In such systems, simulations of this type are developed as a matter of course during the design and development of the domain system. Relying on these simulations can save a considerable amount of time in the knowledge acquisition stages. Diagnosis is defined as a *Diagnostic Process* where the task of diagnosis is divided into several discrete and independent steps (data acquisition, fault detection, diagnosis and hypothesis generation, hypothesis space pruning, verification, test case selection and validation). The demonstration system acts at an advisory level to operators and controllers of spacecraft systems. While opportunities for *meta level reasoning* exist, only a rudimentary level of meta reasoning is performed. The system relies instead on the operator to guide and control the reasoning process.

The remainder of the thesis is structured as follows: The spacecraft Electrical Power System (EPS) domain is introduced in section 2. The diagnostic problem and its subelements are presented in section 3. Artificial Intelligence approaches to model based diagnosis are reviewed and discussed in section 4. The design of the demonstration software system is described in section 5. A summary of the tests performed to evaluate the software system is presented in section 6. The results of testing the software and the level of success in meeting the goals of the project is discussed in section 7. Section 8 concludes the research activity and identifies areas for possible improvements.

Additional background information is provided in the appendices. Appendix A provides the reader with a more detailed look at the Electrical Power System introduced in section 2. Also included in Appendix A is a description of the EPS model and its constituent

electrical component models. Appendix B presents a more detailed look at the tests performed and resulting data summarized in section 6.

A related document titled: *"Model Based Reasoning Applied to the Diagnosis of Spacecraft Electrical Power System Faults: Software Description and Source Code Listings"* [Adamovits 1992] provides a description of the software source code and full software source code listing of all software developed during the course of this thesis work.

2.0 Spacecraft and the Space Domain

Spacecraft have been designed to serve a variety of missions including: unmanned space exploration, manned lunar exploration, communications, earth observation, astronomy and others. All spacecraft are operated at great distances from control centres responsible for monitoring the status and health of the on-board systems.⁷ These on-board spacecraft systems are either part of the payload (observation cameras, radar systems, relay communications systems, etc.) or spacecraft bus or platform (attitude and orbit control; tracking, telemetry and control; structural systems; thermal systems; electrical power system; etc.). The Electrical Power System (EPS) for a radar payload spacecraft will serve as the application domain for the diagnostic system presented in this document.

In the following subsections, relevant issues of the operational environment and the Electrical Power System (EPS) of a radar payload spacecraft will be presented. While the presentation and discussion centres on the spacecraft EPS much of the material has relevance to other space based systems and similar applications.

2.1 Mission Requirements and Orbit Mechanics

The purpose of an Earth orbiting radar spacecraft is to survey, on demand, areas of the earth as it passes over them. The major constraints on the mission are the orbital coverage of the earth, the payload and housekeeping energy requirements, and the available energy

⁷ The monitoring of sensor data and control of many routine operations of on-board systems on manned missions is also performed remotely by ground station based operators.

on the spacecraft. Inevitably, there will be times when more radar exposures are desired than can be achieved.

The selection of an optimal orbit for a radar spacecraft typically results in low altitudes in an effort to minimize RF power requirements and a high inclination to increase the total image coverage of the earth. In order to map out a large percentage of the earth, the orbit plane of a radar spacecraft is highly inclined to the earth's equatorial plane. Retrograde orbits (i.e. inclination greater than 90 degrees) are common. With any inclined orbit, the orbit plane precesses about the earth at a rate dependent on the angle of inclination (decreasing with inclination), the radius and ellipticity of the orbit. As the spacecraft orbits the earth, the earth spins on its axis beneath the spacecraft. These factors define the radar coverage, the frequency and duration of contact possible (spacecraft to ground stations), and the frequency and duration of eclipse operation.

Once the orbital mechanics have been determined from the mission requirements, many constraints defining the spacecraft system, including the ground support and the EPS requirements, will be fixed. A typical spacecraft of this class is the Canadian Radarsat. In this case, the following preliminary orbital estimates apply:

- Orbit altitude: approximately 800 km,
- Orbit period: approximately 101 minutes,
- Eclipse: maximum 18 minutes, 73 days long season centred on summer solstice,
- Orbit is retrograde (i.e. spacecraft is nominally passing in a direction opposite the earth's spin).
- Single ground station contact: 12 minutes contact maximum (at 5 degrees elevation above horizon); typically 3 orbits of contact totalling 29 minutes duration (or less) followed by 4 orbits of no contact. Cycle repeats twice every

day.⁸

Earth observation spacecraft tend to be sun synchronous. That is, the spacecraft orbit plane is at a constant angle to the sun vector at any time of the year (ignoring earth declination). Most spacecraft of this class have day-night orbits (i.e. they observe both sunlit and dark sides of the earth in a single orbit). While also sun synchronous the Radarsat spacecraft has a dawn-dusk orbit. Earth observation spacecraft however need not be sun synchronous and as a result may experience different ground contact and eclipse characteristics than the example presented.

2.2 Operational Requirement for Electrical Power System

Electrical power is required for the monitoring and control of virtually all systems that comprise the spacecraft.⁹ For these systems, other than reaction control systems, electrical power is also the primary energy source that enables the system to function. Electrical power system component degradations and failures impact the effectiveness and could prematurely terminate the spacecraft's mission. The requirement for a reliable source of electrical power by other space systems makes the EPS important to the very existence of the spacecraft.

⁸ This data is based on a single ground station at a 45 to 50 degree latitude.

⁹ Portions of some systems do not require electrical power. Examples of these are: passive thermal radiators and the fixed mechanical structure of the spacecraft. Heaters in the thermal system and moveable components of the mechanical structural system will however require electrical power.

The EPS is required to operate continuously over the life of the spacecraft. The design life of the spacecraft is determined largely by the mission profile and may range from a few months for very low orbit altitude spacecraft through tens of years for deep space exploration spacecraft. Several distinct phases of operation exist. For earth orbiting spacecraft these are: launch, transfer and final orbits. Here we are concerned with operation of the EPS once the spacecraft has attained final orbit.

The radar payload places intermittent high power demands on the EPS which are unlike other on-board systems. The operation of the radar system requires high RF power transmissions followed by periods when the echo signal is received and processed. At times when radar imaging is not performed, the radar system will be switched to a reduced power or standby mode of operation. The duty cycle (transmit / receive verses standby) and peak electrical power demand by the radar system will determine the design of the EPS. For the Space Based Radar program the following typical values can be expected [Moodie and Maskell 1989]:

- The power output of the solar panels is in the order of 40 kw,
- The primary load is a phase array radar antenna having on the order of 10,000 elements,
- When the radar load is turned on, the power requirements during sunlit operation exceed the solar panel capacity by a factor of 2. The difference is made up by discharging the batteries.
- The duty cycle of the radar load is 50% on an orbit by orbit basis,
- During eclipse operation the battery is required to meet all the power requirements of the spacecraft,
- The batteries may be recharged (as required) during sunlit periods when the radar

loads are turned off.

The EPS is also required to have a certain degree of autonomy. The level of autonomy is determined by the opportunity for ground station intervention and the response time required. As noted in the previous section, for spacecraft with high inclination angle low earth orbits (e.g. RADARSAT, SBR), contact with a single ground station is infrequent and of short duration. The delays associated with data transmission precluded handling certain emergency responses in ground based facilities. Several emergency events requiring immediate response will cause breakers to trip (over current) or components to switch to standby mode (under voltage) without ground station intervention. The analysis of the causality of these events and the detection and diagnosis of component degradations is performed through ground based processing. Improved communication contact between the spacecraft and the ground station will significantly increase the opportunity for ground based systems to handle emergency conditions. This may be made possible through the use of multiple ground stations or other spacecraft to relay the signal to ground.

2.3 Electrical Power System Overview

Spacecraft electrical power systems vary in their design as required by the mission, mass, volume, efficiency, reliability, and other factors. A few of the design trade offs performed include:

- primary power source: photovoltaic, thermal (nuclear and solar), chemical reaction, and others,
- energy storage: batteries (NiCad, NiH, lithium, others), capacitors, thermal,

- power distribution: type (AC, DC), voltage levels, frequency (for AC), regulated or unregulated bus voltage,

The EPS described in this section and used as the application domain for the diagnostic system presented in this thesis has the following high level design:

- photovoltaic primary power source,
- NiH (Nickel Hydrogen) batteries as the sole means for energy storage,
- a three tier bus system with unregulated, coarsely regulated and finely regulated DC voltages,
- three levels of power regulators: primary power conditioner; intermediate conditioner; and transmit receive conditioner and auxiliary conditioner (the last two conditioners at the same effective level in terms of power regulation).

The EPS provides power to the spacecraft loads (radar loads plus other spacecraft platform systems) which have a variety of power profiles. Duplication of components and the ability to alter the configuration the power distribution network is used to provide a high level of operational capability. While the operational capability is increased, the total power available to the spacecraft loads may be reduced when components have failed.

The EPS is divided into several parallel strings. Each of these strings can be considered independent of the remaining strings (for the purposes of this discussion). The solar panel power, battery capacity and loads are all equally distributed between the parallel strings. This approach reduces the power requirements of the individual circuit elements to a more acceptable level and reduces potential single point failures by a factor equal to the number of strings used.

Figure 2-1 Electrical Power System Block Diagram

As all strings may be treated identically, the term EPS will refer to a single string from this point forward. The EPS (string), displayed in Figure 2-1, is comprised of the following key elements with the quantity indicated in {}:¹⁰

Auxiliary Conditioners {120} (AuxCond)

Energy Storage Unit or battery {1} (ESU)

Cell Packs {20} (CellPacks)

Cells {100} (Cells)

Intermediate Conditioners {120} (IntCond)

Primary Power Conditioning Unit {1} (PPCU)

Radar Transmit/Receive Modules {960} (RadarLoads)

Solar Panel {1} (SP)

Transmit and Receive Conditioners {240} (TRCond)

The above components are connected together by electrical busses (primary bus, intermediate bus and trCond bus). The busses are not modeled as discrete components but are represented in the voltage and current balance equations of the model (i.e. Kirchhoff's Laws. An additional pseudo component was defined. This component, referred to as the power distribution component (PowerDist), lumps the 120 intermediate conditioners (and all down stream components) into one component. The power distribution component does not model a EPS component as such and was defined purely for programming convenience.¹¹

¹⁰ The name in brackets () is the class name of the Smalltalk class that implements this component for the EpsSimulation. These names are used in the descriptions of the diagnostic system and software in later sections.

¹¹ Appendix A provides additional information on the individual component's models.

2.4 Failure Mode and Complexity Analysis of the EPS Domain

The components that comprise the EPS may degrade and fail in some way over the life of the spacecraft. In this section we review the failure modes of the key components and attempt to identify the complexity of the diagnostic problem in terms of the total number of possible faults that may occur. We will limit our discussion to single fault cases initially, then expand it to identify the two fault cases that are detectable.

The Electrical Power System contains some 9 unique EPS component classes and 1548 components in total. All component classes are modeled as a minimum in their normal behavioral modes to permit the domain and model simulations to run effectively. The 5 most common component classes (intermediate conditioner, auxiliary conditioner, transmit receive conditioner, radar loads and cells) also include fault models. These classes represent 1540 of the 1548 components that comprise the EPS.

Several critical EPS components have internal processors that manage and perform fault recovery management for the component. The Primary Power Conditioning Unit (PPCU) has several internal components that are available to replace degraded or failed components within the PPCU. The Energy Storage Unit (ESU) also includes processors that perform cell health monitoring and charge management. The processors used to monitor and manage the PPCU and ESU provide only a minimum of data to the ground based operators through the telemetry stream. Fault detection and recovery is handled by these processing elements.

The Solar Panel does not have internal processing elements but possesses a passive fault tolerant system through the use of series and parallel sub-strings of cells and fuses. The

photovoltaic cells within the solar panel will degrade over time due to radiation effects¹² and impacts with micro-meteorites. The damage caused by micro-meteorites will appear as an open or short circuit in one or more of the sub-strings. An open circuit in any sub-string will cause a minimal impact on the total power output of the solar panel. A short circuit of any single sub-string will cause a high current flow through the sub-string. This high current flow will blow the fuse associated with the sub-string ultimately isolating the fault as an open circuit.

As the PPCU, ESU and solar panel are internally fault tolerant and are not externally controllable, a complexity analysis of these components is not performed. The remaining components (intermediate conditioner, transmit receive conditioner, auxiliary conditioner and radar loads) referred to here as the *power distribution components* are the subject domain of this thesis. The remainder of the complexity analysis performed here centres on the power distribution components.

Figure 2.1 illustrates the interconnection between the components of the EPS. Each of the EPS components has one or more possible faults. Multiplying the number of a given class of component in the EPS by the number of possible faults for that component results in the number of single fault cases possible in the EPS (6540) as noted in table 2-1.

¹² Degradations due to radiation damage occur over several years and are monitored by observing the trend in output voltage and current from the solar panel. It can be expected that over a 10 year period the solar panel output will drop approximately 15%. As the loss in solar panel output is a very slowly occurring trend, it is not addressed in this thesis.

Component	faults types ¹³	number of components in the EPS	total possible faults
radar loads	5	960	4800
trCond	3	240	720
auxCond	3	120	360
intCond	3	120	360
cell	3 *	100	300
cellPack	0 **	5	0
ESU	0 **	1	0
solar panel	0 ***	1	0
ppcu	0 ***	1	0
Totals	17	1548	6540

Table 2-1 Single Fault Complexity Analysis of the EPS Components and Faults.

- * While the cell has 3 faults, these are managed by the ESU control electronics.
- ** The ESU (energy storage unit) contains a cell management and fault recovery system. In terms of storage devices, the ESU and cell packs are strings of cells and thus are pseudo components only.
- *** This component is internally fault tolerant and thus appears externally to have no faults in the single fault case.

It should be recognized that the analysis performed here is not a detailed review of all possible permutations and combinations of faults that can occur but will present an indication of the scale of the problem domain. Several faults for a given component with different causality are lumped together if they present identical symptoms. (See Appendix A for further details on the failure modes of the individual components.)

¹³ See Appendix A for a description of the component faults.

2.4.1 Multiple Fault Complexity Analysis

In this section we will focus our attention on the power distribution components and consider the multiple fault case (2 or more faults). Multiple fault cases can be divided into two groups independent faults and faults located in closely related components. We will consider 2 or more faults found on the same tile to be closely related. Faults found on dissimilar tiles are defined as independent.

Examining the number of possible single faults for the components of one tile we generate the table 2-2:

Component	faults types	number of components in one tile	total possible faults (on tile)
radar loads	5	8	40
trCond	3	2	6
auxCond	3	1	3
intCond	3	1	3
Totals	14	12	52

Table 2-2 Single Fault Complexity Analysis of a Tile.

There are a total of 52 single faults that can occur on each of the 120 tiles that comprise the EPS. If we consider all possible combinations of unrelated tile 2 fault cases, we arrive at $19 * 10^6$ cases. If we consider the unrelated tile 3 fault cases we arrive at $79.0 * 10^9$

possible cases.¹⁴¹⁵

Multiple faults that are found on a single tile are expected to have a higher probability of occurrence in the EPS. The local interaction of the components however, is expected to cause false symptoms and a degree of fault masking. While several faults can be simulated at one time, many are impossible to detect. For example:

Consider the case where a given transmit receive conditioner has a loss of efficiency fault and the intermediate conditioner providing power to the transmit receive conditioner is open circuit. The fault of the transmit receive conditioner is of little consequence since without power, efficiency has little meaning. Note also that an open circuit for the intermediate conditioner will also cause several faults to be indicated for all attached components (transmit receive conditioners, auxiliary conditioner and all loads). These are not in fact faults but instead an example of alarm propagation.

Discounting all cases where the higher level conditioner has a open circuit or tripped breaker reduces the multiple fault cases considerably. Figure 2-2 displays the possible faults that can occur for components associated with a given intermediate conditioner. To

¹⁴ For 2 unrelated faults: 120 possible first tiles with 52 possible faults each, 119 possible second tiles with 52 possible faults and the order does not matter. That is:
 $(120 \text{ choose } 1) * 52 * (119 \text{ choose } 1) * 52 / 2 = 6,240 * 6,188 / 2 = 19,306,560$ two fault cases.
For three faults: $19,306,560 * (118 \text{ choose } 1) * 52 * 2/3 = 79.0 \cdot 10^9$ unrelated 3 fault cases.

¹⁵ It must be noted that this is not a probability of failure calculation but an indication of the number of unrelated tile faults. The probability of 2 or three unrelated faults occurring together is very low.

interpret the diagram the branches are followed until no further lower level components are found. For example:

- If an intermediate conditioner has an open circuit or a tripped breaker, faults can not be detected in lower level components.
- If the intermediate conditioner is normal or has a loss of efficiency, detection and diagnosis of faults is possible at least to the next level. This includes the components: auxiliary conditioner, and both transmit receive conditioners. The detectable faults for these components are: tripped breaker, open circuit and loss of efficiency.
- If faults are detectable at the transmit receive conditioner level (i.e. the intermediate conditioner is normal or has a loss of efficiency), and the given transmit receive conditioner is normal or has a loss of efficiency then, the detection and diagnosis of faults is possible for the radar loads attached to this component.

The possible two fault cases are summarized in tables 2-3a and 2-3b:

intermediate conditioner condition	auxiliary conditioner detectable faults	transmit receive conditioner detectable faults	radar loads detectable faults
normal	3	3	5 *
efficiency fault	3	3	5 *
tripped breaker fault	0	0	0
open circuit fault	0	0	0

Table 2-3a Fault Masking due to Intermediate Conditioner Faults

* depends on the health of the transmit receive conditioner.

transmit receive conditioner condition **	radar load detectable faults
normal	5
efficiency fault	5
open circuit	0
tripped breaker	0

Table 2-3b Fault Masking due to Transmit Receive Conditioner Faults

** assumes that intermediate conditioner is not open circuit or have a tripped breaker.

To determine the number of *single tile two fault cases* that can be detected, we will examine Figure 2-2 and the above tables.

- Intermediate conditioner has a loss of efficiency - 49 detectable 2 fault cases:
 - 9 faults can be detected at the auxiliary conditioner / transmit receive conditioner level.

- 40 faults can be detected at the radar load level (8 loads with 5 possible faults each). This analysis assumes that transmit receive conditioner faults are not present.
- A given transmit receive conditioner has a loss of efficiency (intermediate conditioner is normal) - 92 detectable fault pairs:
 - 20 faults can be detected in the loads attached to this transmit receive conditioner.
 - 20 faults can be detected in the loads associated with the other transmit receive conditioner.
 - 6 faults can be detected with the auxiliary conditioner (3) and the other transmit receive conditioner (3).
 - 46 additional faults can be detected when we view the problem from the other transmit receive conditioner.
- A given load has one of 5 faults - at least 100 detectable 2 fault cases:
 - 100 fault pairs can be detected in the loads attached to the other transmit receive conditioner (4 loads with one of 5 faults each, 1 load with one of 5 faults = $4*5*5$).
 - Several other faults may be detected in loads associated with the same transmit receive conditioner. The analysis here is incomplete as these components are under instrumented and considerable ambiguity exists in the available data.
- The auxiliary conditioner has one of 3 faults - 44 detectable 2 fault cases:
 - 4 faults can be detected in either of the two transmit receive conditioners not already addressed.
 - 40 faults can be detected in the radar loads (8 loads with 5 possible faults each).

Figure 2-2 Power Distribution Components Fault Tree

The above analysis results in 285 single tile, two fault cases being detectable (i.e. symptoms of one fault do not mask the second fault). This analysis applies to one tile only. As there are 120 intermediate conditioners, this results in 34,200 (i.e. $120 * 285$) possible two fault cases within the same tile that are not masked. Further analysis can be performed to superimpose the unrelated tile case. Consider the following example: one tile has 2 faults, another unrelated tile has 1 fault. This can occur in $105 * 10^6$ ways.¹⁶

The actual number of faults that the Multiple Fault Diagnostic System (MFDS) presented in later sections can actually detect and diagnose is expected to be lower than the levels specified in this section. The success of the MFDS is limited by the completeness of the knowledge bases and the ability to adequately address ambiguities that may occur in the reasoning process.

¹⁶ Calculated as: $(120 \text{ choose } 1) * 285 * (119 \text{ choose } 1) * 52 / 2 = 120 * 285 * 119 * 52 = 105.8 * 10^6$.

3.0 The Diagnostic Problem

The term *diagnosis* implies several connotations to the layman. Typically, diagnosis is associated with the medical field. This belief is further substantiated by several sources, one dictionary definition states:

diagnosis "1a) the identification of diseases from the examination of symptoms. b) an opinion so reached 2a) through the analysis of facts or problems in order to gain understanding b) an opinion reached through such analysis." [Collins 1982]

The first definition refers to the medical diagnosis. The second definition make reference to the analysis of facts or problems to gain understanding. While the above definitions of diagnosis are true, there are several issues that are not adequately described. The Diagnostic Process includes several additional steps that are not immediately evident. In this section we will introduce and attempt to define the terms diagnosis and Diagnostic Process as precisely as possible.

Diagnostic Process is the activity of identifying the underlying causes for unexpected changes in a system's behaviour. While similar to the above dictionary definition of diagnosis, the diagnosis process includes several key reasoning steps that must be completed along the way. These include acquiring raw data through testing hypotheses to confirm their validity. For the purposes of this discussion the Diagnostic Process is defined to include the steps listed below:

- 1 data acquisition
- 2 fault detection

- 3 diagnosis and hypothesis generation
- 4 hypothesis space pruning
- 5 hypothesis validation (testing) on a model of the system
- 6 test case selection
- 7 hypothesis verification (testing) on the real (domain) system
- 8 observing new data supporting or refuting the selected hypothesis

The analysis of the new data that becomes available in step 8 may provide sufficient supporting information to accept the hypothesis selected and validated (step 5) as being an acceptable diagnosis or may require that the previous Diagnostic Process be repeated (steps 1 to 5). In many cases, the Diagnostic Process can be repeated from some intermediate step depending on the failure recognized in the last full cycle. This will typically entail returning to the diagnosis and hypothesis generation or hypothesis space pruning steps. In the event of a total misdiagnosis, attaining a correct solution may require returning to the data acquisition stage.

The control and management of spacecraft systems is performed remotely for all unmanned spacecraft and for several systems of manned spacecraft. The personnel responsible for monitoring the telemetry data from the spacecraft and implementing corrective actions through a command stream are called *spacecraft operators*. The Diagnostic Process noted above will be described here in the general spacecraft operations sense.¹⁷ Consider the following scenario:

¹⁷ The Diagnostic Process is typical of the approach used in the spacecraft industry today. Inputs from the spacecraft design and operations community has provided a basis for much of the formulation Diagnostic Process approach generally. The Diagnostic Process and the activities of spacecraft operators and control centres described here are based on personal communications with several personnel: David Andean (Canadian Department of Communications), Ed Fuchs and Barry Turner (Telesat Canada).

A spacecraft operator is monitoring the status of several spacecraft each with a few thousand status and sensor points. Up to this point all sensor data appear to be within acceptable range when unexpectedly one spacecraft indicates several sensors out of range. Not recognizing the combination of symptoms, the spacecraft operator notifies superiors who resolve the problem and suggest the implementation of corrective measures. The spacecraft returns to successful operation and the operator returns to his monitoring tasks.

What has really happened? The diagnosis as described, is as simplistic as the dictionary definition. The Diagnostic Process however, is considerably more complex. This more detailed Diagnostic Process is described in a general sense from the spacecraft operations perspective a means to describe the underlying processes. They are:

- 1 Data acquisition* - Data acquisition is performed from the spacecraft on-board systems' sensors, through the spacecraft to ground station RF telemetry links, terrestrial data links from the ground station to the command centre and telemetry decoders. The output of the data acquisition system is floating point and boolean data. This data may be stored by some means for possible future reference (disk, digital tape, etc). ¹⁸
- 2 Fault detection* - As with data acquisition, fault detection is an ongoing and continuous activity limited only by the data availability. In the typical spacecraft control centre, fault detection is accomplished through both automated and manual means. In both cases received telemetry data is compared with expected data. The reference or expected data may be found in operations manuals, data bases or the

¹⁸ As noted previously, the availability of data may be limited by orbit and number of ground stations.

operators mental model of the sensor in question given the expected state of the system.

Automated fault detection is typically accomplished by ground station based computers performing a quantitative analysis only. Computer based *quantitative fault detection* is accomplished by comparing the received data against upper and lower set points or limits. A fault is detected when the telemetry data deviates from the range defined by these limits. The limits used in this comparison are not time varying or dynamic in nature and are typically set manually by the spacecraft operator. At times when transients are expected to occur, the limits are set quite wide to reduce the possibility of false alarms. In many cases these widely set limits serve little no value in monitoring the rate of transition. This can only be accomplished by manual means. Fault detection performed in this way is effective in meeting the needs of largely static data only.

Manual fault detection is accomplished through the actions and observations made by the spacecraft operator. The operator may have noticed that the numeric value of one or more telemetry data values exceeds the expected level by more than an acceptable bound. This is analogous to the computer based quantitative fault detection noted in the preceding paragraph. Alternatively, the operator may compare *expected trends* in a given data parameter with those observed. This comparison is referred to here as *qualitative fault detection*.

Several areas exist for the above approach breaking down. The spacecraft operator, as mentioned previously, may be expected to monitor several spacecraft each with several hundred to a few thousand sensors. The opportunity for overwhelming the

operator with several hundred unrelated alarms is very real. This may cause the distraction of the operator by a minor fault in one spacecraft system when another system or spacecraft warrants his attention. Another problem is alarm propagation. That is, a single point failure may propagate several hundred alarms as data values trip limit checks. The lack of a focus of attention or alarm filtering facility can cause problems and delays in arriving at an effective solution.¹⁹

- 3 *Diagnosis and hypothesis generation* - Given a number of detected faults, the operator will perform the initial diagnosis of the situation. If the problem is viewed as minor in nature, he may be authorized to effect a solution immediately noting the changes made to his superiors. In the event that more serious problems have been detected, or several hypotheses exist as to the cause of the problem the next level of expert may be brought in to examine the data.

An expert (or for more serious problems, a committee of experts) will be identified to examine the data and assist the operator in solving the problem. In the event that a committee of experts is assembled²⁰, it will include one or more of spacecraft systems experts, the spacecraft operator, senior spacecraft operators, designers and others. Each of the personnel will have expertise key to solving the particular problem at hand. This committee will formulate hypotheses as to the faults that exist and diagnose the situation.

¹⁹ When confronted with several thousand alarms, operations personnel have great difficulty in focusing in on the relevant indicators. An example of this is the Three Mile Island and other nuclear reactor failures where the loss of control resulted in considerable cost and destruction.

²⁰ Such committees are typically only formed in the event of potential significant loss of the spacecraft capability (i.e. total loss of the spacecraft, loss of primary payload capability, loss of significant reaction control fuel, etc).

In the event that others are brought into assist, they can be expected to repeat several of the steps performed by the operator. They will examine the data that identified the fault and review the fault detection results. In either case (i.e. operator or committee), the personnel performing the diagnosis will approach the problem from one of two directions. They will apply the available telemetry data and detected faults to their knowledge of failure modes to find the set of faults hypotheses that hold when the specific combination of symptoms were detected (forward reasoning). Alternately, they will search their knowledge of known fault hypotheses for those that are supported by the evidence (i.e. telemetry data and faults detected) that presently exists (backward reasoning).

Based on the diagnosis, the diagnostician(s) will formulate a set of hypotheses of possible diagnoses. Each of these hypotheses may be supported by some subset of all the data available. It should be noted that the hypotheses may not be consistent together but each individually must be plausible.

- 4 *Hypothesis space pruning* - The set of hypotheses resulting from the previous step will be examined discarding those that are not strongly supported or are contrary to that expected based on other believed hypotheses. This is performed in an attempt to reduce the work in the remaining steps. Hypotheses may be rejected based on a variety of criteria (known operational state of the spacecraft, solar conditions, etc).
- 5 *Hypothesis validation* (on a model of the system) - The hypotheses that remain following pruning are all plausible given the problem at hand. One hypothesis will be selected and this will become the basis of more detailed examination. In the

event that the dynamics of the situation are not well understood, the selected hypothesis may be tested on a model of the spacecraft system simulating the situation under study.

The model used to simulate the problem situation may be a mathematical or theoretical model, computer software based, electronic or mechanical hardware based and may include actual (ground based) spacecraft hardware²¹. When flight hardware is not available for the model but the domain is well understood and accurately modeled by other means, these models may provide precise numerical evidence supporting or refuting the anticipated behaviour. When the domain is less well understood, the simulation activity provides more qualitative trends rather than precise results.

If it has been determined that the hypothesis is a plausible description of the observed behaviour, the hypothesis can be considered to be validated. If this is not the case, the hypothesis will be rejected and another hypothesis will be examined. This will be continued until the personnel examining the situation are satisfied that a selected hypothesis is valid.

- 6 *Test case selection* - Once the reviewers have selected a hypothesis that is believed to be the most likely, a test is selected that will provide either further supporting or refuting evidence when applied in the domain.

²¹ Often several spacecraft of a given design will be manufactured over a period of a few years. Problems identified in spacecraft launched earlier may be verified on the ground on subsequent spacecraft's flight systems. Information gained in this way will often initiate several design refinements in the subsequent spacecraft.

- 7 ***Hypothesis testing*** (on the real system) - Once a hypothesis has been selected (and validated in step 5), the spacecraft configuration is modified through the application of the test scenario (selected in 6). The test may require firing thrusters, rotating the spacecraft on some axis, setting components on or off, etc. The intent is to either resolve the problem or to provide further supporting or refuting evidence to be examined in an effort to improve the belief in the hypothesis describing the fault.

- 8 ***Observe results*** (new data supporting or refuting the selected hypothesis) - The spacecraft's telemetry data will be observed carefully following any change made to its configuration through the application of the test case. That is, the examiners will look for further data that the hypothesis tested does represent the fault condition postulated. If verification of the diagnosis (test hypothesis) is not observed, the hypotheses that suggested the specified treatment may then be refuted. This new information will be used to refine the model so that it remains consistent with the domain in the future. If other hypotheses remain on the pruned hypothesis list, the process is repeated from step 5 through step 8. This process continues until all hypotheses have been examined and refuted or accepted.

The Diagnostic Process described here can be applied to several domains. In the remaining subsections of section 3, several important concepts of the Diagnostic Process are described from the generalized domain perspective. The Diagnostic Process described here has been demonstrated on a simulated spacecraft electrical power system (EPS), described in section 4. Section 5.0 presents the Diagnostic Process as implemented in this software system and in terms of the spacecraft EPS.

Figure 3-1 The Diagnostic Process

3.1 Fault Detection

Considerable confusion exists in the definition of the terms fault detection and diagnosis.

It is defined here as:

fault detection - detecting the existence of an anomaly in the behaviour of the system in question without necessarily identifying the cause or significance of the fault.

Fault detection does not include identifying the specific component(s) that caused the faulty behaviour nor the changes of the specific component that may have caused the new behaviour²².

Fault detection can rely on a variety of techniques to identify the existence of a fault. Typically, any sensory indicator that is outside some expected range can be classed as a fault. While the sensors may take any form, we will limit our discussion to those that are typically found in engineering systems. The sensor output typically provides numeric or logical readings in its measurement of voltage, current, position, time, velocity, status, state, etc. Sensor precision and accuracy will affect the accuracy of the fault detection process. If the sensor resolution is too low, the error bound on the detection of faults will increase accordingly. Sensor failures further complicate the situation.²³

Described in terms of an equivalent electrical circuit, a fault detection system can be considered to be a comparator. The comparator in this case has two inputs, one connected to a domain sensor and the second to some reference model that contains expected values (given the state of the domain system). For logical sensory data, the fault detection system

²² The fault detection system may however indicate the location where the fault was detected. Note that one component failure may result in several related components having faults indicated.

²³ Sensor degradations and failures are not addressed in the MFDS software implementation.

may be viewed as an exclusive OR gate. That is, when both inputs are logical 0 or logical 1 the output is 0, when the inputs are dissimilar (i.e. 1 0 or 0 1) the output is 1. This concept is extended in the following paragraphs in the introduction of the qualitative and quantitative fault detection systems as implemented in the Multiple Fault Diagnostic System.

Qualitative techniques can provide an indication of a fault condition. Qualitative fault detection is applied to partially processed data, rather than sensor data directly. In this case, the direction of change of a variable is analyzed. Consider the example:

A rheostat controlling the current flow through a light bulb is changed such that it has lower electrical resistance. The current flowing through the light bulb can be expected to increase resulting in a brighter light.

It can be expected that if the resistance is reduced that the bulb will be brighter. The reference input (representing the light intensity) to the comparator would indicate increasing values are expected. If in this example, the light was no brighter, a *qualitative* fault would be detected.

To describe the qualitative change of the data, a tristate logic system is defined. This logic system represents decreasing, unchanged or increasing data denoted as (- 0 +) respectively. Qualitative data may be considered to be the first derivative (over time or state change) of the sensory data. The qualitative fault detection system compares two qualitative data values and provides a binary logic result. That is, two identical inputs (i.e. {(- -) (0 0) (+ +)}) result in a binary logic output of 0. An output of 1 results for other pairs of possible states (i.e. {(- 0) (- +) (0 -) (0 +) (+ -) (+ 0)}).

An analogy to this is the two input exclusive OR gate in (binary) logic systems. The exclusive OR gate, provides an output if either but not both inputs indicate a logical one (i.e. {(1 0) (0 1)} results in a 1 output, {(1 1) (0 0)} results in a 0). If the Exclusive OR

gate was extended to handle a tristate input rather than binary inputs, the functionality in figure 3-2 would result. The operation of the qualitative fault detection system implemented in the MFDS is further described in section 5.1.1.

The comparator analogy also applies to numeric (i.e. floating point) sensor data. Fault detection in this case is referred to here as quantitative fault detection. A quantitative fault detection system checks to ensure that the sensor data received is between expected high and low reference limits (i.e. limit checking). An equivalent circuit in this case includes two voltage comparators and an AND gate. The voltage comparators each have two inputs (for signal and reference). When the signal exceeds the reference voltage, the output of the comparator becomes a logical 1. Arranging 2 voltage comparators in a circuit with high and low reference voltages and tying the comparator's outputs together through an AND gate will provide a logical 1 output whenever the signal output becomes too high or too low. This is illustrated in figure 3-3.

The reference values for quantitative fault detection need not be static. In a case where time or state varying data is expected, reference levels may be similarly varied. Returning to the equivalent circuit (figure 3-3), the reference levels may be provided by signal generators with state or time varying characteristics. The upper and lower limit of acceptable values would then vary to more closely monitor the expected sensor data (Figure 3-4). The normal state dependent characteristics of a signal are thus analyzed in terms of the expected behaviour plus or minus some tolerated error bound²⁴. Such a

²⁴ The tolerated error bound is necessary to filter out small errors in modelling, variations due to the sensor precision and accuracy, inherit noisiness in the data and other data sampling errors.

Operational considerations may also influence the error bound. At times of critical operations or manoeuvres, the error bound may be changed to detect the slightest variation. At other times it may be relaxed to reduce the number of false detections.

Figure 3-2 Qualitative Fault Detection

Figure 3-3 Quantitative Fault Detection Equivalent Circuit

Figure 3-4 Signal and References vs. Time

system permits the tracking of time varying data if accurate models can be developed to provide the reference levels. In the quantitative fault detection system implemented and presented in section 5.2.2, the model of the domain is used to provide the reference values to the quantitative fault detection system.

3.2 Diagnosis and Hypothesis Generation

Diagnosis is the process of identifying the underlying mechanism that has caused anomalous behaviour to be observed. Hypothesis generation is the formulation of one or several plausible conclusions or hypotheses based on the diagnosis reasoning. As noted in the previous section, we have identified fault detection as a separate activity from diagnosis. A diagnostic system need not be exercised if faults have not been detected. Partitioning the fault detection activity from diagnosis can potentially save considerable unnecessary processing time.

Diagnosis is generally performed using a different approach when compared to fault detection. Fault detection as described in the previous section is performed through a comparison between the expected and actual data observed. Diagnosis typically requires the addition of deeper reasoning to be successful in complex domains. In simpler domains, sufficient data may be available to completely identify all possible faults that exist immediately. In this case, the fault detection system may provide sufficient information to lead one to the faulty component directly.

The approach to diagnosis can be as varied as the domains in which it is applied. In the medical field, accurate models are not available for several aspects of the human

organism. Several systems interact in as yet poorly understood ways, many individuals respond differently to the same medication, side effects of treatment often occur in unexpected ways. Diagnostic systems in the medical domain typically rely on clinical evidence gathered over several years to form the basis of the knowledge base and reasoning process. Information gathered in this way is classed as experiential in nature. Medical diagnosis includes a seemingly qualitative nature in the reasoning process.

Science and engineering applications typically have considerably more information and accurate models of the mechanisms than medical applications²⁵. These models, when available, typically form the basis of the reasoning process. Software models in this environment may be encoded in several ways. The ultimate source of the information however, remains the underlying theories and theoretical models available that describe the domain.

It has been noted that the availability of information and its precision will influence the possible approaches to diagnosis. Ultimately, the goal is to perform the diagnosis with sufficient accuracy and with a minimum of resources. While deep models may be possible due to the readily available information, this may not be the most resource affective approach. Factors that will influence the approach include:

- Availability of design information or theoretical domain model(s),
- Availability of experiential knowledge (one or more experts),
- Resource limitations (computational resources, development time, etc.).

²⁵ It is recognized that while models do exist in the medical profession for several of the chemical and physiological aspects of humans, much remains only poorly understood. It is suggested that model based reasoning may be applied to a small proportion of the possible medical diagnostic applications.

Generally, diagnosis systems are classed by the reasoning approach and the level of modelling in the knowledge applied to perform the diagnosis. These include:

- Knowledge based systems typically containing experiential knowledge and little in the way of encoding the underlying domain mechanisms (shallow reasoning),
- Detailed understanding of the components that comprise the domain (deep reasoning),
- Complex structural descriptions (e.g. logic descriptions, simulations, other models) of the domain (model based),
- Direct effort from a hypothesized goal and search for supporting data or evidence (backward chaining or reasoning),
- Direct effort into finding the hypothesis that can be derived from the data presented (forward chaining or reasoning).

Knowledge used in the reasoning process also has a wide variance in where it is acquired and level of domain understanding necessary. Experientially based knowledge is typically the source for shallow reasoners. Detailed design knowledge can permit the development of deeper reasoning and model based systems.

The choice of forward or backward reasoning approaches is typically made based on the computational load imparted by the domain knowledge and its structure. Backward reasoners are appropriate where the number of consequents (hypotheses) is far fewer than the number of antecedents (raw data points) that need to be examined. In this case several hypotheses may be examined in parallel. The set of plausible hypotheses reduces as refuting data is identified resulting in fewer hypotheses being proposed as plausible. Forward reasoners are most appropriate when the number of raw data values (antecedents) is far fewer than the number of possible diagnosis hypotheses. In forward reasoners, the

available data is typically examined in parallel resulting in some small set of plausible hypotheses. The goal in both forward and backward reasoners is to identify supported and plausible hypotheses for further examination.

In the science and engineering domains, a considerable wealth of information is generally available to the developer of diagnostic systems. This information is in the form of design (or parametric) models, simulations, theoretical models, design equations, etc. The availability of this deep knowledge permits a variety of approaches in the development of deep reasoning or model based systems.

The presence of deep knowledge does not preclude the development of shallow based reasoning systems. It was noted previously that great benefit can be derived from relying on a model (simulation) to contain the deep knowledge alone. The resulting reasoning systems then are quite shallow in nature. It is the thesis of this document that this simplifies the knowledge base development and maintenance process significantly.

The diagnosis and hypothesis generation system concludes with a list of all plausible hypothesized faults from the diagnosis that has just been completed. While duplicate hypotheses do not exist, the list of hypotheses is not a minimal list nor are all hypotheses contained within the list necessarily well supported. The reduction of the hypothesis list into a consistent minimal list is performed by the next step Hypothesis Space Pruning.

3.3 Hypothesis Space Pruning

The next activity in the Diagnostic Process is to reduce the list of hypotheses resulting from the diagnosis and hypothesis generation step into a minimal ordered list. It should be noted that several extraneous or poorly supported hypotheses may exist as a result of the Diagnostic Process. The goal of the hypothesis space pruner is to eliminate hypotheses from the list of hypotheses that are in fact symptoms of other faults, thereby creating a minimal list, and to order this minimal list of hypotheses from the most likely candidate through the least likely.

Hypothesis pruning is an important aspect of the reasoning process. Performed effectively it will reduce the later computation considerably. The Diagnostic Process through the hypothesis generator will result in several hypotheses being made available for consideration. Several hypotheses however may be invalid (i.e. based on poorly supported or erroneous data or reasoning). These poorly supported hypotheses are removed from the list and from further consideration in the Diagnostic Process. The resulting list is smaller but more specific and better supported.

Of the remaining hypotheses, not all of these are equally specific in identifying the faulty component(s). Some hypotheses may be quite general in nature and identify problem areas rather than problem components. For example: one hypothesis may identify an assembly of 10 components as having a fault, another may identify one component of this assembly as being the faulty component. While both hypotheses may be equally valid, the higher specificity offered by the later hypothesis is of greater value in locating the cause of the fault.

Information brought to bear in ordering the hypotheses may include hypothesis specificity, MTBF (mean time between failures), environmental factors, operational issues, etc. The relative importance of the component to the system's operation will also influence the ordering of the hypotheses.

3.4 Hypothesis Validation

Hypothesis validation involves testing a selected hypothesis against the model of the domain to determine if it indicates a plausible fault condition given constraints that exist (environmental, configuration, operational modes, previous faults, etc.). The hypothesis validation step receives as input, an ordered minimal list of hypotheses resulting from hypothesis space pruning. One hypothesis is selected from this list and the model is altered to represent the fault indicated by the selected hypothesis. The intent is for the resulting operational state of the model to be such that it generates data as if it included the hypothesized fault condition. The data resulting from the altered model is examined to determine the accuracy of the hypothesis in describing the behaviour observed in the domain. If the model now describes the domain behaviour it is possible that the revised model is a better approximation of the domain.

Validation starts by selecting a test hypothesis from the minimal list of ordered hypotheses. The selected hypothesis is used to identify the component that is potentially at fault. All relevant hypothesis and data associated with the identified component are recorded for later reference. The model is then altered to represent the fault identified in the selected hypothesis. The fault detection; diagnosis and hypothesis generation and; hypothesis space pruning steps are repeated and the resulting hypotheses and data is

compared with the previously recorded data. If the model is a better representation of the domain than previously, fewer faults will be indicated for the component being examined. If no faults remain for the identified component, the model is now assumed to be a better representation of the domain. Note that throughout this process the domain has not been altered or modified in any way.

If the model can not be modified to represent an accurate model of the domain in its present fault state, then the hypothesis is rejected and another is selected and tested. This continues until a hypothesis can be validated against the model. Once an appropriate hypothesis has been verified on the model the hypothesis is forwarded on for later testing in the domain (verification).

Several refinements to the above approach are possible. As an example, a trade off may be performed between the most appropriate candidate from the list and the single most appropriate test serving several candidates²⁶. Several selection criteria may be used including:

- Identify the single test that can maximize the number of hypotheses that can be accepted or refuted.
- Identify the single most critical component hypothesized to be at fault and perform the test that would verify the component's health (i.e. select the first element on the pruned hypotheses list).
- Identify the test that will disrupt the system the least degree possible.
- Use Mean Time Between Failure (MTBF) information of the identified candidates and/or failure modes identified.

²⁶ This later issue however, is not addressed in the software implementation described in later sections.

If during the selection process a 'candidate test' has been identified (instead of a single hypothesis), the single test performed may provide information relevant to several hypotheses. Each of the several hypotheses may be provided supporting or refuting information as to their validity. If several hypotheses appear to be valid, given the test with the model, the test is forwarded on for testing in the domain (verification). Another possible refinement is the analysis of failed hypothesis candidates. The results of this analysis may be used to revise the selection criteria for future candidates that possess similar or related characteristics.

3.5 Test Case Selection

Once a hypothesis has been selected and validated as plausible against the model, a second test will be identified that will provide further supporting evidence that the identified fault condition does in fact exist in the domain. Through the actions of the validation system the domain and model can be expected to possess a fault and to represent the same fault respectively. If the model accurately represents the domain, performing identical actions on both the domain and model will result in identical behaviour being observed. The actions to be performed are those that will provide the greatest supporting evidence verifying or refuting the hypothesis. The test selected will attempt to meet the above criteria. The application of the test and analysis of the resulting data is the responsibility of the next step, verification.

3.6 Hypothesis Verification

Hypothesis verification involves applying the selected test case, then observing and analyzing the results. As noted in the previous section, the model and domain can be expected to behave identically (with regard to the issue under investigation) if the hypothesis identified is correct. The analysis performed by the verification system includes the examination of new or previously undetected faults and the review of remaining faults associated with the hypothesis under investigation.

In the event that the verification step does not in fact identify the component at fault, then the hypothesis is rejected and the process is repeated from hypothesis validation step forward (sections 3.4, 3.5 and 3.6). In the event the hypothesis has been proven true, the model of the domain can be revised to align it with the domain.²⁷ The alteration of the model in this way can be considered to be a simplistic learning system. That is, the system learns the changing operational characteristics of the domain as hypotheses are detected, diagnosed and proven true.

²⁷ Both the fault detection and the diagnosis steps rely on direct comparison between the model and the domain sensor data. Not revising the model will result in repeatedly identifying faults that have already been diagnosed. Revising the model to represent an accurate representation of the domain will prevent this unnecessary processing. Possible errors can however be induced into the model. These errors result from making erroneous changes to the model during the process of attempting to keep the model and domain consistent. These issues are discussed in later sections of this document.

4.0 Artificial Intelligence Based Diagnosis

The diagnosis of systems can be performed in a variety of ways, fault tree analysis, case analysis, symptom based, model based and others. These and other approaches may be implemented using manual or automated means. Automated, computer based approaches may rely on procedural or knowledge based techniques. In this section Artificial Intelligence (AI) techniques appropriate for diagnosis are reviewed and placed in the context of the Diagnostic Process described in the preceding section.

This section also introduces the Multiple Fault Diagnosis System (MFDS), an AI based software implementation of the Diagnostic Process presented in section 3. The MFDS is demonstrated on the power distribution portion of the spacecraft Electrical Power System presented in section 2. The MFDS relies principally on model based reasoning and rule based techniques in its implementation. Details of the MFDS design and its implementation are presented in section 5.

Fault detection and diagnosis have been a popular application domain on which to demonstrate AI based reasoning techniques. Expert systems have been developed to address such diverse domains as:

- Medical diagnosis: Mycin [Shortliffe 1976].
- Fault detection in the presence of large data sets (i.e. > 1000 state variables) [Anderson and Nystrom 1985],
- Diagnosing of spacecraft subsystems [Krishnamurthi, Mayer and Friel 1986; Baker and Marsh 1990],
- Diagnosing spacecraft electrical power systems [Adams et al 1985; Baruah and Campiche 1986; Momoh and Zhang 1990; Ringer, Quinn and Merolla 1990;

Walters et al 1990]

- Communications networks for spacecraft [Tang 1990], and for telephone switching networks [Goyal et al. 1985]

The goal of diagnostic systems is to "Identify [the] malfunction (disease) that best accounts for observed discrepancies (symptoms) in system behaviour" [Hamscher and Patil 1989] (page MA1-1). In terms of the spacecraft Electrical Power System (EPS) that is the application domain of this work, the goal is to detect the existence of faults, to identify the circuit element(s) that are at fault and in many cases to prescribe corrective action.

Many diagnostic systems (both automated and manual approaches) implement a variation on the Diagnostic Process introduced in section 3.0. The AI approach to diagnosis often has wide variations. Some systems rely on model based techniques [Davis 1984; Genesereth 1984; deKleer and Williams 1987; Hamscher 1991; Reiter 1987], experiential knowledge [Shortliffe 1976], qualitative modelling [Kuipers 1986], combined qualitative and quantitative reasoning [Williams 1991]. In the following subsections AI approaches to the Diagnostic Process implemented in the MFDS are presented and reviewed.

4.1 Rule Based Systems

Earliest efforts in developing knowledge based systems relied on logic of various forms to develop a closed world (or nearly so) description of the domain. The logic structures used were unfamiliar and difficult for many experts to understand. This difficulty limited the direct participation of experts in knowledge base development and maintenance. The knowledge (rule) structure of many rule based systems improved situations by providing

the experts and system developers with a common and more easily understood language in which to describe the domain.

Rule based systems also provide a means to address real world problems that have a less well founded basis than logic based reasoning. The rule based approach tends to encode relationships between known facts (varying data of the domain in question) with rules that an expert may use to explain their reasoning process. These 'rules of thumb' are often based more on experiential knowledge rather than a formal foundation in logic, statistical or probabilistic analysis.

The Mycin system developed by Shortliffe [Shortliffe 1976] is perhaps the best known of the experientially knowledge rule based systems. Mycin addresses the medical problem domain of selecting the appropriate antibiotic medication for the treatment of infectious bacterial diseases. Throughout the reasoning process, Mycin interacts with the user by asking questions as required in order to provide sufficient data to perform the reasoning. As reasoning proceeds, laboratory tests or observed physiological conditions may be required to be input into the systems²⁸. Following the completion of reasoning process, Mycin provides an explanation of the reasoning that led to the prescribed medication.

Mycin uses a backward chaining inferential approach ([Shortliffe 1976] page 121) using Certainty Factors (CF) contained in the rules to guide the reasoning process. Certainty Factors encode a subjective probability value rather than statistical or probabilistic data. The CF value indicates the level of belief or disbelief that the given object described in the rule is valid. CF are encoded in the individual rules and are real numbers in the range:

²⁸ The interactive nature of Mycin and other similar systems can be quite disconcerting for the user. Questions are occasionally asked in areas that the user believes to have no relevance to the present problem and reasoning process.

-1 through +1. A CF of +1 indicates absolute certainty that the hypothesis is true, -1 indicating absolute certainty that it is false ([Shortliffe 1976] page 98). As the CF tends toward 0 the certainty of the value as either true or false can not be drawn.²⁹ During the reasoning process Certainty Factors are propagated with the rules creating new CF values in a way similar to Bayes Theorem.³⁰ These new CF values are used to guide and direct the reasoning as the level of belief indicated by the CF rise and fall.

The problem with the certainty factor approach is that its theoretical validity is hard to determine. It is stated that in this domain statistical or Bayesian models are not appropriate. Shortliffe states rightly that given the poor models available of the domain, such approaches have little validity in the medical profession. Certainty factors were thus developed to better suit the medical application at hand. The problem arises in that the reasoning process is guided in what appears to be quite arbitrary ways. It is not apparent if certainty factors were eliminated that the diagnosis would be any less precise.

Mycin appears to be fully adequate in meeting the goal of diagnosing bacterial infections and prescribing the appropriate medication. Mycin (along with many medical expert systems) however, is not now in clinical use [Weiss and Kulikowski 1984] page 68. The lack of acceptance of Mycin and other medical expert systems may be based more on medical scepticism and legal concerns rather than unacceptable results.

In his concluding remarks on how Mycin may be improved, Shortliffe suggests dynamic ordering of rules and dynamic ordering of conditions within the rules as means by which

²⁹ The believed disbelieved values are a simplification of the system defined by Shortliffe. These include: known, notknown, definite, notdefinite [Shortliffe 1976] page 102.

³⁰ The Certainty Factor calculation, though similar to Bayes Theorem, is complicated by the fact that both the level of belief and disbelief must be maintained [Shortliffe 1976] page 177.

the system may be made faster (page 151-152). Prescreening is also identified as an alternate to the above (page 153).

4.1.1 Multiple Fault Diagnostic System Rule Bases

The Multiple Fault Diagnostic System (MFDS) includes several rule based reasoning elements. In its implementation of the Diagnostic Process (described in section 3), the MFDS relies on distinct and independent rule bases for fault detection, diagnosis and hypothesis generation, hypothesis space pruning, verification, test case selection and validation.³¹ The rule bases of the MFDS do not rely on the certainty factors (as used by Mycin), statistical or probabilistic approaches to guide the reasoning process. Several reasoning elements of the MFDS include focus of attention code and rules.

The focus of attention process starts with the fault detection system. This first level identifies components on a tile by tile basis as having a discrepancy. These are identified by examining the telemetry data received from the spacecraft and comparing this with expected data. In the event that differences greater than a specified threshold are detected, for any component within the tile, then all components on that tile become *fault detection candidates* for further examination by the reasoning elements within the fault detection system.

The focus of attention process within the fault detection system is performed entirely by procedural code (through set comparisons in Smalltalk). The reasoning elements of the

³¹ Each of these elements of the MFDS also contain procedural and object oriented portions written in Smalltalk.

fault detection system do not perform further focus of attention analysis.³² The present approach is effective in reducing the computational requirement significantly. In the present design for the EPS there are 120 tiles. In the event that only one power distribution component has a fault, the focus of attention in the fault detection system provides a 120:1 reduction in reasoning to be performed.

The diagnosis and hypothesis generation component also perform a focus of attention analysis. In this case the diagnosis rules operate almost entirely on the results of the fault detection system. Faults detected and presented to the diagnostic system are used to guide the diagnosis.

The focus of attention reasoning of the diagnostic system operates as follows: In the event that the fault detection system has detected a fault with a component, then the component is added to the *diagnosis candidates* list. If however, no faults were detected, then this component may not be examined further (unless it is identified as a diagnosis candidate by other means). If any component has an entry on the diagnosis candidates list, then the diagnosis rules associated with the component may be examined (a given rule however will only be applied if its other antecedents are also met). The application of a rule may also cause other components to be added to the diagnosis candidates list. This process continues until all diagnosis candidates have been examined. The inclusion of this approach into the diagnostic process was observed to reduce the computation required by a factor of approximately 5 to 8 times. The actual improvement is dependent on the location of the fault and number of false symptoms that appear.

³² At the point of initially examining the raw data, a finer grain in identifying components to focus on may not be sensible. Partitioning the problem to too low a level may hinder the diagnostic activity by requiring additional complexity in the knowledge base with no increase in diagnosis accuracy.

The hypothesis space pruning system is itself a focus of attention system. The pruning system reduces the list of hypotheses from those resulting from the diagnosis and hypothesis generation system to a smaller set containing plausible supported hypotheses only. This smaller set of hypotheses is examined by later systems (verification, test case selection and validation). Reducing the number of hypotheses to be examined later reduces the processing time by the same factor. The hypothesis space pruning system reduces the hypotheses set by a factor between 5 and 10 times³³.

The focus of attention approach implemented by the reasoning systems of the MFDS is believed to provide the benefits of dynamic rule ordering and prescreening as suggested by Shortliffe in his recommendations and concluding comments [Shortliffe 1976] pages 151-153.

The rule bases of the MFDS include both forward and backward reasoning approaches. The fault detection system examines the telemetry data by forward reasoning only. The diagnosis and hypothesis generation system initially reasons backward (i.e. does a given component have a fault). Once a component is identified as having a fault, it is examined by forward reasoning (i.e. the hypotheses identified by the fault detection system lead to what diagnosis hypothesis). The pruning system is also a bidirectional reasoning system. Initially reasoning backward to identify components that have identified diagnoses then reasoning forward from the available diagnoses to identify the most likely hypotheses. The remaining systems of the MFDS perform forward reasoning only.

³³ The possible reduction in hypothesis set size is dependent on the components actually at fault. For example: If an intermediate conditioner is at fault, following diagnosis and hypothesis generation, several erroneous hypotheses will exist for lower level components (transmit receive conditioners, auxiliary conditioner and radar loads); The presence of a fault at the lowest level, a radar load, will result in fewer erroneous hypotheses existing prior to pruning.

Details of the operation and function of the reasoning systems of the MFDS are presented in section 5. Section 5 also includes information on the rule structures, the reasoning systems organization and data structures of the elements of the MFDS.

4.2 Model Based Reasoning Systems

Considerable research has been directed at diagnostic systems that utilize model based reasoning approaches [Davis 1984; Genesereth 1984; deKleer and Williams 1987; Hamscher 1991; Reiter 1987]. In this section, we will briefly review previous works in model based reasoning for diagnosis. In the next section, section 4.2.1, model based reasoning will be placed in the context of the Multiple Fault Diagnostic System, a model based reasoning system applied to spacecraft Electrical Power System fault detection and diagnosis.

In model based diagnostic systems, the model is typically used to provide data of the expected behaviour which is compared and contrasted with the observed behaviour from the system under examination. Models in these application domains are based on theories or experimental data from the domain. Although reasoning components of the resulting inferential system may also include experiential knowledge, the reasoning process is largely based on the deep knowledge contained within the model. The models are *not* based on *experiential knowledge*. In contrast, the Mycin system does not contain a model (by this definition) and relies almost entirely on experiential knowledge.

To be effective, model based reasoning systems require an accurate model of the domain in order to reason about the expected behaviour. Models should also be complete and

independent. This implies that the models of the system components are to be complete and independent of the structure of the system in which they are found ([Kuipers 1986] page 290 referring to deKleer). Models used in AI based software systems may be encoded in many alternate ways. These include: frames, logic based systems, software simulations, object oriented programming approaches and others.

Initial efforts in developing reasoning systems addressed the problem from a logician's prospective. Logic based models used in these systems are difficult to develop as a closed world description of the problem is required. To be effective, a well founded description of the domain (in terms of a consistent logic definition) is a necessity. Models however are rarely complete. Absolute understanding of the situation is in many cases allusive. A circuit model may accurately represent the electrical properties, but not the physical properties (size, location, thermal characteristics, etc.) of the components at all. While the electrical model is of considerable value, the non-electrical properties of the domain can not be reasoned about. These non-electrical factors may however have a strong influence on the final diagnosis. In the diagnostic field multiple representations are often used to provide a means of encoding side effects or alternate perspectives [Davis 1984]. For example in an electrical circuit, knowledge of adjacent pins possibly shorting, or an adjacent component's heat output may be of value in the reasoning process of chain reactions of failures.

The system developed by Davis [Davis 1984] provides the multiple model (or multiple perspective) approach as outlined above. The components of the circuit are viewed from several perspectives which are encoded in the constraints of the rule structures. For the digital circuit application domain used here, several models are defined from both the functional and physical perspective. The functional models include: the logic properties

of the digital component and the electrical characteristics of its interface. The physical models include intended electrical interconnection, physical structure of adjacent pins shorting, adjacent components interacting thermally, capacitance coupling of adjacent printed circuit tracings, etc. This system performs diagnosis by examining the inconsistency between the model and the domain data. In an effort to identify the component that is at fault, the constraints of the model are one by one suspended and the model is repeatedly checked for consistency with the domain. This constraint suspension approach is similar to the early justification based truth maintenance system (see [McDermott 1991] for a unifying theory of reason maintenance).

Fesq and Stephan [Fesq and Stephan 1989] demonstrate the approach of Davis on their Fault Management System that detects and isolates faults in spacecraft electrical power domain. In this system the fault detection system operates in a closed loop with a simulated spacecraft electrical power system.³⁴ It is not apparent that this system can address multiple concurrent faults or operate successfully with already identified faults.

Models as noted can take many forms. Generally, the model used provides some level of numerical, logical or qualitative calculation with a numerical or logical conclusion. (Qualitative modelling and reasoning systems are described in the section 4.3.) Models may be based on sound scientific theories or black box relationships between input and output nodes of a component based on observed results. Frequently, black box model definitions of individual elements are in terms of piece wise continuous functions over some finite range of values. Several individual black box model elements are combine together to form the overall model of the system. This is the approach demonstrated in

³⁴ One potential problem with this system that does not appear to be present with Davis' is the need to define the constraints in both a forward and reverse direction. I suspect this is a limitation of the knowledge base interpreter used.

the MFDS model described in section 4.2.1 and section 5.

It is important to recognize that models of any sort have limitations in their ability to accurately depict the systems behaviour. Finite limitations may exist for the range of parameters that the component models and resulting system can be successfully be operated in. As with all knowledge bases, exceeding the limits of the knowledge can negatively impact the accuracy of the final conclusion drawn. That is, the system draws conclusive results that are in fact based on an incorrect extrapolation of correct relationships. This is referred to as brittleness of the knowledge base or reasoning. That is, the system not recognizing the limitation of its knowledge, draws conclusions on incorrect or incomplete model definitions. Brittleness in the reasoning (in varying degrees) remains a problem with all AI based systems.

4.2.1 Model Based Reasoning in the Multiple Fault Diagnostic System

The Multiple Fault Diagnostic System is a model based reasoning system that relies on a *software simulation* as the model and rule based components (as previously described) to perform the reasoning processes. The model used by the MFDS is a discrete event, continuous simulation capable of modelling both the normal and failed component conditions of the Electrical Power System. The simulation used as the model is similar to simulations created by designers of electrical and electronic systems during the process of evaluating alternative processes and design approaches.

The MFDS model is unlike those developed by Davis, Genesereth [Davis 1984;

Genesereth 1984] and others in that it does not define the domain in terms of logic statements but instead relies on an object oriented simulation. Rothenberg [Rothenberg 1989] (page 75) defines modelling in the following terms:

"Modelling in its broadest sense is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer, cheaper than reality, instead of reality for some purpose."

It is the belief of this author that the use of a simulation based model is simpler, easier to maintain and has greater utility than logic based models. The basis for these claims will be presented in following paragraphs.

The model (or simulation) used in the MFDS was developed using object oriented techniques. The simulation is constructed from instances of component classes and displays the composite behaviour of the instances' individual behaviours interacting within the simulation. Each of the individual component instances may have its behaviour modified to match the evolving behaviour of the domain. When performed in an automated way by reasoning elements of the MFDS, model parameter revision may be considered to be an explanation based learning system. Systems capable of learning are referred to as a third generation expert system by Widman and Loparo [Widman and Loparo 1989] (page 13).³⁵

The use of an object oriented simulation as the model has several advantages over logic based approaches. As mentioned in the proceeding paragraph, the simulation is

³⁵ Widman and Loparo [Widman and Loparo 1989] define generations of expert systems as:
First generation - experientially based reasoning systems (e.g. Mycin and similar systems);
Second generation - systems based primarily on model based reasoning and deep knowledge rather than experiential approaches (e.g. systems previously described by Davis, Genesereth, and others.
Third generation - expert systems that focus primarily on learning or model parameter tuning from examples.

constructed from instances of EPS component definitions in the form of software object class instances. As suggested by Kuipers [Kuipers 1986], these component descriptions are complete and independent of the structure of the system in which they are found. The object classes that describe the EPS components localize the data into the object (data hiding) and provide default component behaviour descriptions in the form of a code block. The replacement of an individual instance's behavioural description with a new code block can be performed as the MFDS learns the new behavioural characteristics. The behavioural description of the EPS component can also be modified in subtle ways. If the detected change in behaviour is slight, the parameters of the existing behavioural description can be tuned to improve the component model's ability to represent the domain.

In all cases of component behavioural model replacement or parameter tuning, the intent is to revise the model so that it provides a better representation of the domain. Tracking changes in the domain as they are discovered, prevents rediscovering known faults in the future.³⁶ This approach also simplifies the structure of the reasoning systems so that the reasoning systems operate by comparing the model behaviour and the actual EPS behaviour. Alternate forms of learning typically used in rule based or logic based approaches involve developing specialization rules, performing exception handling and others. It is believed that the simulation based approach greatly simplifies learning when compared to logic or purely rule based learning systems. The details of how the model is developed, updated and utilized in the MFDS will be described in greater detail in section 5.

³⁶ The repair of spacecraft is generally not possible in orbit. The EPS is expected to operate to some reduced level of capability if component degradations and failures have occurred.

4.3 Qualitative Reasoning Systems

Qualitative reasoning and qualitative models are based on qualitative behaviour constraints and the physical parameters that describe and define the domain. In Kuipers [Kuipers 1986] qualitative simulation is described as the key to qualitative clausal reasoning. Qualitative structure and behaviour are presented as abstractions of differential equations and differentiable functions. The goal here is to develop models of the domain in terms of "... qualitative descriptions of the deep mechanism, capable of representing incomplete knowledge of the structure and behaviour of the mechanism" ([Kuipers 1986] page 289).

Qualitative reasoning attempts to describe the behaviour of complex systems through the use of: constraint equations derived from examination of the physical structure of the system and from initial conditions that exist at the time the simulation was started. The result of the simulation is a description of the domain in terms of a set of possible future states. These state variables are tri-state values (+, 0, -) that describe the expected trend in the specific parameter.

The state variables may be considered as representing the direction of change of the state variables that describe the system. That is, the sign of the first derivative of the data (as noted in section 3.1). While this is a simplification of the approach by Kuipers and others, it will prove sufficient for the description that follows. Consider the example from Kuipers (page 294-295):

"Two flasks (A and B) partly filled with water are connected together by a small tube at the bottom. The flasks are at equilibrium, that is the flow of water between them is zero. At this time a small volume of water is added to flask A. Initially, the pressure of water in flask A is higher than flask B. This causes water to flow between them through the connecting pipe. In time, the pressure will be balanced as the two flasks reach a new equilibrium. At this point the level of water in both

flasks will be higher than the initial conditions. The level in A will have dropped from the level attained immediately following the addition of water."

Constraints that exist in this example are: "

- Each tank has a pressure which depends on its amount of water.
- The rate of flow through the tube depends on the difference between the pressures.
- A flow through the tube increases the amount in one flask and decreases the other. " (From Kuipers page 294).

Qualitative simulation of the above system results in predicted qualitative behaviour:

- a The level of A will be lower than immediately following the addition of the water and the level in B will be higher.
- b The level of the two flasks will be higher than initially.
- c Equilibrium will occur eventually.

Quantitative analyses however, are not possible. The absolute level of the water in either tank is not known nor is the size of the connecting tube. This lack of deeper level information does not permit the calculation of the time required to reach equilibrium. As the qualitative reasoning proceeds, this lack of quantitative reference data will cause the accuracy of the reasoning to drift from reality. Kuipers identifies periodic *landmarks* where errors in the qualitative simulation (QSIM) are corrected against quantitative references.

4.3.1 Qualitative Reasoning in the Multiple Fault Diagnostic System

The fault detection system of the MFDS includes a qualitative reasoning component. Unlike QSIM, this system does not include a qualitative simulation but instead a

qualitative interpretation of a quantitative simulation (and the received telemetry data) is used. The model generates quantitative expected telemetry data which is compared with the previous data to create a *qualitative data frame* (expected). The qualitative reasoning system compares this qualitative data frame to the domain qualitative data (actual) generated in the same way. The model is updated at regular intervals providing the equivalent reference values similar to the landmark values of QSIM. At this point the similarities between Kuippers work and the fault detection system diminish. The qualitative fault detection system of the MFDS compares two qualitative values, rather than attempting to predict future states qualitatively. The qualitative fault detection system is described in greater detail in section 5.2.1.

4.4 Control of the Reasoning Process

The reasoning process in any AI based system requires a control mechanism to direct the appropriate next action to be performed. Typically, these control mechanisms are procedural in nature³⁷, hidden within the rules³⁸, include a reason maintenance facility or are based on higher level reasoning systems (metalevel reasoners) that reason about the lower level reasoning processes.

³⁷ Procedural control mechanisms include: fixed rule selection mechanisms (specificity, certainty factors, RETE algorithm, etc.), procedural classically structured code written to perform actions in fixed sequences, any similar mechanism.

³⁸ Consider the example of an inference engine that fires rules with a highest specificity first (i.e. those that have the greater number of antecedents). This inference engine can be artificially forced to select a given rule if it contains several dummy antecedent clauses. Hiding the control mechanism in the rules in this way results in exceptional difficulty in determining the actual control structure.

In systems where the knowledge is contained in several autonomous (or nearly so) subelements, the need for a control mechanism in computationally limited environments becomes quite clear. In Hayes-Roth [Hayes-Roth 1985] an architecture is presented where a metalevel blackboard reasoning system controls and manages the problem solving activity. This system is demonstrated in a planning application in which the knowledge sources in the lower level blackboard develop sub-plans (or plan elements). The solution develops as the higher level system examines the sub-plans posted on the problem solver's blackboard and determines the best way to control the reasoning process (using a higher metalevel blackboard).

In Russell and Wefald [Russell and Wefald 1991] various approaches at metalevel reasoning as applied to active agent system control systems are reviewed and contrasted. The architectures for the metalevel reasoner include: distributed control to the individual active agents, procedural approaches (referred to as concrete), and an active agent based control system (i.e. agents that are assigned the reasoning task only).

Russell and Wefald present the concept of computational utility of the active agents in a computationally limited environment. Utility is defined as the anticipated utility to the overall reasoning process. The utility of an action is reasoned about without actually performing the action. As with all meta reasoning systems there exists a cost trade off between intrinsic utility of the action and the cost (time) of the deliberation of the added utility of performing the action; is it better to just perform the action or to reason (deliberate) about the usefulness (utility) of the action in the global sense (page 363). There is an practical limit to the extent that metareasoning can be performed before the resources expended offset its benefit.

Several other approaches to the control problem include maintaining the truth of the reasoning process in data bases. The actual control mechanism in this case is located in other mechanisms (rules, procedural etc). The data bases however, are a key component of the control process. The content of these data bases is used to guide the reasoning process and is updated and revised during the reasoning as additional information becomes available. These systems are generally referred to as Truth Maintenance Systems (TMS) [McDermott and Doyle 1980], [Doyle 1979], [McAllester 1980], Assumption-based Truth Maintenance Systems (ATMS) [deKleer 1986a; deKleer 1986b] and more recently unifying these theories into generalized Reason Maintenance Systems [McDermott 1991]. A variety of approaches are used to record the reasoning process as it occurs using as a basis: logic, justifications, environments and assumptions.

Reason maintenance systems tend to be computationally costly. In the ATMS [deKleer 1986a; deKleer 1986b], nodes are generated to hold supported consistent environments as they are developed. These nodes can be retracted as the supporting evidence is withdrawn from the environments in the node. The process of checking and updating the nodes is computationally costly. In fact, the computational time required increases exponentially with the number of nodes. Considerable effort is being expended to bring this to a linear rate. For systems handling several thousand data values, it is not apparent that the benefits offered by reason maintenance systems offset the costs given the present exponential growth in processing time with increased complexity.

Approaches relying on the use of metalevel reasoning to guide the reasoning process are thought (by this author) to be more effective than truth maintenance techniques. The need for metalevel reasoning systems is not apparent in simple problems. That is in problem domains where the inferential process is quite limited in scope or in the extreme case

where direct symptom-diagnosis is possible. The Diagnostic Process defined in section 3 will however benefit from a supervisory system to monitor the process through the various steps.

4.4.1 Control Mechanisms in the Multiple Fault Diagnostic System

The MFDS includes two basic control mechanisms, the focus of attention systems for the reasoning components and metalevel reasoners used to evaluate the verification and validation steps. The focus of attention mechanisms described in section 4.1.1 are designed to improve the efficiency of the reasoning system by reducing unnecessary inferencing. Metalevel reasoners examine the level of success in the validation and verification steps and attempt to direct future actions into areas of greatest return.

Data flow through the MFDS is fundamentally pipe lined. Each step in the Diagnostic Process must complete before initiation of the next step. The necessity to perform meta level reasoning over the fault detection, diagnosis and pruning steps is not present. The actions performed by the validation and verification steps however provide an opportunity to demonstrate this technique. The MFDS at present includes only a rudimentary metalevel reasoner for the validation system. This system presently examines the level of success in validating the test hypothesis as being plausible. Possible future additions include controlling the other lower level reasoning systems or actively controlling the model.

5.0 Multiple Fault Diagnostic System Design

In this section and following subsections, the design of the Multiple Fault Diagnostic System (MFDS) is described in terms of: operational capability and high level structures and constructs. The intent here is to describe the solution structure and the reasoning process rather than the software that implements this process.³⁹ While the software implementation is not described in detail, code fragments (or equivalent pseudo code) are occasionally used to provide supporting examples to the discussion. For a more complete description of the domain and software models, see the Appendix A and the software description document [Adamovits 1992]. In this later document additional information in the form of MFDS software code description and the source code listings for all software that comprises the MFDS system, EPS models and user interfaces is presented. This information will prove useful for those interested in examining and testing the system further.

The Multiple Fault Diagnostic System (MFDS) is a software implementation based on the Diagnostic Process introduced in section 3 and depicted in figure 3-1. The MFDS relies on a model of the domain to provide reference data representing the expected behaviour. The architecture of this system is shown in figure 5-1. As noted previously, the MFDS is demonstrated on a spacecraft Electrical Power System. References to components refer to the electrical components of the EPS.

The MFDS relies on a software model (or simulation) of the domain as a key element in the reasoning process. The MFDS is constructed largely using shallow reasoning systems

³⁹ Where appropriate, the name of the Smalltalk class that implements the software module is identified in brackets (e.g. (MetaRules)). However, the software is not described in detail.

Figure 5-1 The Model Based Diagnostic Process

(knowledge bases). These shallow reasoning systems contain little of the underlying EPS design (deep knowledge) beyond a component connectivity model. It will be demonstrated that locating the *deep knowledge* of the domain solely within the model is an effective means of encapsulating this knowledge. It will also be demonstrated that by revising this model as diagnoses are hypothesized and later validated, unnecessary further processing of *known faults* is not performed saving future computations. The model thus tracks the domain in an automated manor similar to explanation based learning systems.

The Multiple Fault Diagnostic System is presently an experimental system developed to explore the concepts and ideas presented in this thesis. As this is the case, the MFDS is not fully developed from the perspective of a spacecraft operator concerned with managing the EPS⁴⁰. While the system may be limited from an end user perspective, the MFDS adequately demonstrates the processes and concepts presented in this thesis.

In the present implementation, the MFDS user interacts with a multi text pane browser, Figure 5-2. Through menu operations the user can, step by step, initiate the processing of each of the elements of the MFDS. The results from this processing are displayed in text form on the relevant window pane.

⁴⁰ Limitations in the existing implementation are: the MFDS does not address all possible faults in the domain (see Appendix B.3). The user interface was designed and developed to suit the MFDS developer rather than an end user, an explanation system is not provided, etc.

Figure 5-2 Multiple Fault Diagnostic System Browser

Window panes are provided to access the following MFDS elements:

- Data Interface; Fault Detect System (including qualitative and quantitative fault detection),
- Diagnosis and Hypothesis Generation System,
- Hypothesis Space Pruning System,
- Validation System,
- Test Case Selection System and,
- Verification System.

Figure 5-3 Electrical Power System Domain Model Browser

The MFDS also includes a graphical interface to the *MFDS model and domain simulation*, Figure 5-3 and the supporting textual list pane, Figure 5-4⁴¹. Faults are injected into the domain through popup windows and menu selections, Figure 5-5 and 5-6. The operation of the MFDS is then initiated to detect, diagnose, prune, validate and verify the faults simulated in the domain. During fault validation, the model will be modified by the MFDS to better represent the domain. Modifications performed to the model can be

⁴¹ The browser graphical window and text window for the model and simulation appear identical to the user with the exception of the text name on the title bar and the colour of the window. Figures 5-3 and 5-4 display the simulation windows. The model windows are titled: "EPS Model Browser" and "EPS Model Data".

Figure 5-4 Electrical Power System Domain Model Browser Text Window

Figure 5-5 Intermediate Conditioner Popup Window

Figure 5-6 Cell Pack Popup Window

examined through the model's graphical interface window menu system and the text pane associated with the browser. During the reasoning process, the reasoning elements of the MFDS write a brief log of their activities to the Smalltalk System Transcript pane.

The Multiple Fault Diagnostic System starts the fault detection and diagnostic process by acquiring data from the Model and EPS Domain simulation. This is accomplished by the

Data Acquisition System described in section 5.1. The Fault Detection System (section 5.2) initiates the qualitative (section 5.2.1) and quantitative (section 5.2.2) fault detection systems. The Data Acquisition System and Fault Detection Systems are initiated from a single menu selection from the fault detection pane of the Diagnostic System Browser. Following the successful detection of faults that may exist, these are displayed on the fault detection pane of the browser. The user may choose to examine these or proceed to the Diagnosis and Hypothesis Generation System.

The Diagnosis and Hypothesis Generation System (section 5.3) is also initiated through a menu operation, in this case from the menu pertaining to the diagnostic pane. Following the successful completion of the diagnosis, the hypotheses resulting are displayed on the diagnosis and hypothesis generation pane of the Diagnostic System Browser.

The next pane on the browser initiates the Hypothesis Space Pruning System also through a menu operation. The ordered reduced list of hypotheses is displayed on the pane following completion of the pruning system.

The Validation System is associated with the next pane in the sequence. Initially, the candidate that will be examined is selected through a menu operation. The candidate is identified and listed on the browser pane. With a second menu selection, the validation process is initiated. The validation terminates with a message in the validation pane indicating the level of success in validating the candidate hypothesis. (The MFDS to this

point may be considered to be a generate and test system. The candidate is selected (generated) through the actions of the proceeding 4 systems. The validation system then performs the test on the model.)

The Test Case Selection System proceeds to select an appropriate test for the Verification System. The candidate for this test is the same candidate identified in the validation system tests. This process is also initiated through a menu operation and the resulting selected test is displayed on the test case pane of the browser.

The final step in the MFDS reasoning process is verification. Again one pane of the MFDS browser window is reserved for this activity. As before, the verification process is initiated through a menu selection and the results are displayed on the verification pane.

A seventh pane, not assigned to a specific reasoning system, provides the facility to perform fault detection, diagnosis, pruning and candidate selection with a single menu selection. This pane is located on the right most side of the window Diagnosis System Browser. Further details of the operation of the MFDS are provided in the software description document [Adamovits 1992].

The MFDS was implemented using the Smalltalk/V 286 language from Digitalk Inc. [Digitalk 1988a] and a subset of the Prolog language [Digitalk 1988b; Clocksin and

Mellish 1981] supplied with Smalltalk/V 286. The MFDS software was developed on an Intel 80386 33 Mhz based PC clone computer with 4 megabytes of RAM storage.

5.1 Data Acquisition System

The Data Acquisition System (DataInterface) provides the interface between the MFDS reasoning elements and the two simulations used in the Multiple Fault Diagnosis System that represent the model and the domain. This data interface isolates the model and domain from the reasoning systems by maintaining a duplicate copy of key data structures. Isolating the reasoning systems from the model and domain ensures that, at the time the reasoning process is initiated, the relevant data remains consistent until completion.

Data maintained by the Data Interface includes the following arrays for the Domain⁴²:

commands - the commands array indicates the commanded circuit breaker positions for those components so equipped and commanded radar load on/off status. The data within the commands array are logical values only.

oldCommands - as above but one simulation update older.

tm - The telemetry array indicates the full telemetry frame that is available from the spacecraft electrical power system (i.e. Domain). All data that is normally available from the spacecraft is located in this frame. This includes: component

⁴² The data contained within the tm and command arrays is noted in Appendix A on a component by component basis. Maps of the data arrays are found in volume 2 as part of the software description.

breaker status information⁴³ (logical values), voltage, current and power levels (floating point variables).⁴⁴

oldTm - as above but one simulation update older.

The following data arrays are maintained for the Model:

expectedTm - as in tm but contains the expected values of tm generated by the Model instead,

oldExpectedTm - as above but one update older.

The data interface additionally provides the capability to initialize the model to match the domain. This can be performed in one of two levels of initialization:

- Initialize the commanded status of the radar loads only. Or,
- Initialize the charge level of the battery cells and commanded status of radar load components to match the domain.

Both these actions may be performed through the action of menu selections from the user interface provided by the Diagnostic System Browser. In normal operation, the MFDS will initialize the model by setting the commanded status of the radar loads only. Other initialization values will not be set (cell charge levels, circuit breaker status, etc.).⁴⁵

⁴³ Note: Breaker status is included in both the telemetry stream and the command stream. The values of these status words have very different meanings: The *telemetry* stream reports the breaker status, (i.e. whether it has tripped or not); The *command* stream is used to command the breaker to reset after it has been determined that it has tripped.

⁴⁴ Note: The tm, oldTm, expectedTm and oldExpectedTm frames do not include temperature readings that would normally be available on a spacecraft as these are not modeled.

⁴⁵ Normally the MFDS is not aware of faults as they occur. Through the action of the reasoning systems within the MFDS, faults are detected, diagnosed, etc. Initializing the MFDS model to be identical from the onset to the domain is not realistic and defeats the purpose of the MFDS reasoning process.

breaker status information⁴³ (logical values), voltage, current and power levels (floating point variables).⁴⁴

oldTm - as above but one simulation update older.

The following data arrays are maintained for the Model:

expectedTm - as in tm but contains the expected values of tm generated by the Model instead,

oldExpectedTm - as above but one update older.

The data interface additionally provides the capability to initialize the model to match the domain. This can be performed in one of two levels of initialization:

- Initialize the commanded status of the radar loads only. Or,
- Initialize the charge level of the battery cells and commanded status of radar load components to match the domain.

Both these actions may be performed through the action of menu selections from the user interface provided by the Diagnostic System Browser. In normal operation, the MFDS will initialize the model by setting the commanded status of the radar loads only. Other initialization values will not be set (cell charge levels, circuit breaker status, etc.).⁴⁵

⁴³ Note: Breaker status is included in both the telemetry stream and the command stream. The values of these status words have very different meanings: The *telemetry* stream reports the breaker status, (i.e. whether it has tripped or not); The *command* stream is used to command the breaker to reset after it has been determined that it has tripped.

⁴⁴ Note: The tm, oldTm, expectedTm and oldExpectedTm frames do not include temperature readings that would normally be available on a spacecraft as these are not modeled.

⁴⁵ Normally the MFDS is not aware of faults as they occur. Through the action of the reasoning systems within the MFDS, faults are detected, diagnosed, etc. Initializing the MFDS model to be identical from the onset to the domain is not realistic and defeats the purpose of the MFDS reasoning process.

5.2 Fault Detection System

The primary elements of the Fault Detection System are the qualitative and a quantitative fault detection components. In addition to these, the Fault Detection System also includes data filters and a focus of attention component to guide and control the fault detection reasoning process and a Prolog knowledge base that contains a component inter-connectivity definitions.

Data found in the Data Acquisition System relevant to both the qualitative and quantitative fault is processed partially here before being presented to the qualitative and quantitative fault detection systems. The fault detection system filters the telemetry data by performing a rudimentary comparison between like data values in the *actual* and *expected* data arrays. Data pairs that differ in value, more than a specified threshold level, are stored in difference arrays. The content of these difference arrays is used to identify candidates for the qualitative and quantitative fault detection systems to examine in greater detail. Two difference arrays are created:

Difference in expected telemetry (diffExpTm) - a comparison between the actual telemetry from the domain (tm) and the expected telemetry (expectedTm) from the model⁴⁶.

Difference in commanded positions and breaker status values (diffCommands) - this comparison is between the actual command array from the domain (commands) and the previous command array (oldCommands).

The qualitative and quantitative fault detection systems additionally process the available data to suit their respective reasoning approaches to fault detection.

⁴⁶ The data structure is that of a Smalltalk dictionary. The keys to the dictionary are data array index positions where the difference was detected between the two arrays being compared. The value at a given dictionary key is a 2 element array containing the actual and expected data values from the source arrays.

2

PM-1 3 1/2" x 4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

PIONEERS IN METHYLENE BLUE TESTING SINCE 1974

MicroD

10800 COUNTY ROAD 5, BURNVILLE, NH 03037 USA
TEL: 603 485 7800 FAX: 603 485 7801 TLA 940000400

The qualitative and quantitative fault detection system is not concerned with the causality of the fault detected. Causality of the symptoms observed will be resolved in later stages of the Diagnostic Process. The reader may notice that several faults may cause a given rule to fire. The faults detected by the fault detection system are not conclusive. The shallow analysis of the component failures does not stand detailed scrutiny. The intent here is to detect, not diagnose faults. The conclusions drawn from the fault detection systems are to be viewed individually as one of several plausible faults.

5.2.1 Qualitative Fault Detection

The qualitative fault detection system that is part of the MFDS is based on a simplified model of qualitative reasoning or physics described earlier (section 4.3). Typical approaches rely on a description of the domain in terms of partial differential equations [Kuipers 1986]. Such a model does not exist within the MFDS. Instead, a quantitative simulation of the spacecraft electrical power system is available. *Qualitative interpretation* of this model's *quantitative data* provides the qualitative data needed for fault detection.

The qualitative interpretation of a data item involves comparing the present value with the previously available value. If the present and past values are equal (within some tolerance) then the qualitative indicator for this data is 0 (i.e. no change). If the present data is higher or lower than the previous data then +1 or -1 are indicated respectively. This qualitative interpretation of the quantitative data is applied to each element of the two dictionaries (diffCommand and diffExpTm) created by the Fault Detection System earlier. As noted previously, each key of these dictionaries point to a two element array of expected and actual values. These two values are compared to create the qualitative

value (+1 0 -1) as appropriate.⁴⁷ The result of this activity is the creation of two new dictionaries qualDiffExpTm and qualDiffCommand. These dictionaries contain keys identical to the source dictionaries (diffExpTm and diffCommand) and qualitative values (-1 +1) (0 is implied for all others).

Once the qualitative data bases are created, candidates for further examination are identified. In the present system, when a component is identified as a candidate all components immediately related to it are also examined. This includes: the intermediate conditioner, both transmit receive conditioners, the auxiliary conditioner and all the radar loads attached to these transmit receive conditioners.

The Qualitative Fault Detection component (QualitativeFaultDetection) is similar to the comparator introduced in section 3.1 and illustrated in figure 3-2. As implemented, the system performs a comparison between the derived data from the model and the data from the EPS domain. The operation of the qualitative comparator is summarized below:

model data (expected values)	-1	-1	-1	0	0	0	1	1	1
domain data (actual data)	-1	0	1	-1	0	1	-1	0	1
Qualitative Fault Detection results	-	F	F	F	-	F	F	F	-
F fault detected									
- no qualitative fault detection output									

Table 5-1 Qualitative Fault Detection

⁴⁷ Note: The diffCommand and diffExpTm dictionaries contain increasing or decreasing quantitative pairs only. A qualitative value of 0 (no change) is implied for all other data not identified in the diffCommand or diffExpTm dictionaries.

A Prolog based reasoning system (QualRules) performs the actual qualitative fault detection task. This knowledge base is partitioned on an EPS component by component basis (intermediate conditioner, auxiliary conditioner etc.). The comparison (expected to actual) of the qualitative data is performed by the Prolog clause `qualTest`:

```
qualTest(expected, component, parameter) :-  
    qualData(component, parameter, actual) ,  
    eq(actual, expected).
```

The clause reads as follows:

```
If  
    the value bound to the 'expected' variable equals the value identified by the EPS  
    'component's' qualitative data value identified by 'parameter' symbol,  
then  
    succeed (i.e. return true).
```

The following is a sample rule from QualRules; `auxCond` - rule 2:

```
auxCond(auxCond) :-  
    qualTest(0, auxCond, #state)48 ,  
    qualTest(-1, auxCond, #power) ,  
    qualData(#intCondNum, intCond) ,  
    assertz(qualHypotheses  
            (#intCond, intCond, auxCond, #openCircuit)).
```

The rule reads as follows:

```
If  
    the circuit breaker was not tripped or reset (i.e. state = 0) and,  
    the power output decreased (power = -1).  
then  
    openCircuit may be a possible fault for this component.
```

⁴⁸ Several syntactic differences exist between standard Prolog [Clocksin and Mellish 1981] and Digitalk's Smalltalk Prolog.

- The character '#' defines the following characters as a symbol name.
- Variables in Smalltalk Prolog begin with lower case only. The character '_' is not permitted in variable names.
- A variable that has a first character capitalized is a Smalltalk global. These are not used in Prolog. Although these definitions are nonstandard Prolog forms, they remain consistent with Smalltalk's definition.

This rule was written with the following knowledge of the operation of the device:

- the auxiliary conditioner is modeled as a constant power device.
- if the breaker for this component did not trip then no change is expected in the power output of the component. That is, normal steady state operation of the auxiliary conditioner is indicated by state = 0; power = 0.
- Observing a decrease in power (-1) where none was expected (0) is identified as a fault (as defined in the above table).

The remaining rules associated with the auxiliary conditioner detect other qualitative discrepancies and faults that can occur with this component. Similar rules identify faults for the remaining components.

In summary, the qualitative fault detection system examines dictionaries of quantitative difference pairs between the expected and actual values and generates equivalent qualitative dictionaries. These lead to the examination of selected candidates by the qualitative reasoning based fault detection system. This reasoning system confirms that the qualitative discrepancy detected are in fact symptoms of faulty components and identifies plausible faults. The output of the qualitative fault detection system is a list of plausible fault hypotheses. These will be presented later to the Diagnosis and Hypothesis Generation system for further processing.

5.2.2 Quantitative Fault Detection

The quantitative fault detection system (QuantitativeFaultDetection) detects faults through the comparison of actual and expected numeric and logical data values (section 3.1 and figures 3-3 and 3-4). Inputs to this system are the raw data within the data interface and the data preprocessed by the fault detection system. Further processing of the data arrays (similar to that performed for the qualitative fault detection system) is not required.

Initially, candidate components for further examination are identified. As in the qualitative fault detection system, when a component is identified as a candidate all components immediately related to the component (i.e. on the same tile) are also examined.

A Prolog based reasoning system (QuanRules) performs the actual quantitative fault detection task. This knowledge base is partitioned on a component by component basis (intermediate conditioner, auxiliary conditioner etc.). The following is a sample rule from QuanRules; trCond - rule 1:

```
trCond(trCond, intCondNum) :-  
    quanTest(trCond, #current, result) ,  
    assertz(quanHypotheses(#intCond, intCondNum, trCond,  
                           #currentError, result)).
```

The rule reads as follows:

```
If  
    when tested the identified transmit receive conditioner has a current error greater  
        than some threshold  
then  
    indicate that the error exists and store it with the supporting data
```

The above rule relies on the quanTest rule to perform the comparison:⁴⁹

```
quanTest(component, parameter, valueArray) :-  
    quanData(component, parameter, valueArray) ,  
    is(actualValue, valueArray content at: 1) ,  
    is(expectedValue, valueArray content at: 2) ,  
    not( eq(actualValue, nil)) ,  
    not( eq(expectedValue, nil)) ,  
    is(sum, actualValue + expectedValue) ,  
    not( eq(sum, 0.0)) ,
```

⁴⁹ Note: Much of the work performed by this Prolog clause is in fact performed by Smalltalk through the use of the Prolog *is* operator. The *is* operator is necessary as several of the language constructs normally available in Prolog are (in this version) only available through Smalltalk. The *is* operator is used extensively in the knowledge bases that comprise the MFDS.


```

                                "return true on anything over 2% difference"
is(result, (( 1 - ( ((actualValue - expectedValue) abs)/sum )
                                                    < 0.99))) ,
eq(result, true).

```

This clause reads as:

```

if
    the following error checks all pass:
        is there data in the quantitative database for this query and,
        the data not nil and,
        the summation of this data is not zero and,
        the difference between the actual value and the expected is greater than a 2%
        difference,
then
    the clause returns true.

```

It will be noticed that the quanTest passes only with a less than two percent error. In the present system, the model is an exceptionally accurate representation of the domain (they are two instances of the same classes). Differences of two percent in this case are of significance. In most cases however, the tolerance on the test will by necessity be greater to accommodate small inaccuracies in sensor readings and errors in the model in representing the domain.

The system examines the diffTm and diffCommands dictionaries of quantitative difference pairs and identifies candidate components for further examination. The reasoning system then confirms that the quantitative discrepancy detected (in the dictionaries) are in fact symptoms of faulty components and identifies plausible faults.⁵⁰ The output of the quantitative fault detection system is a list of plausible fault hypotheses. These will be presented later to the Diagnosis and Hypothesis Generation system for further processing.

⁵⁰ If conformation can not be done, the discrepancies are not examined further.

5.2.3 A Priori Knowledge Base

The a priori knowledge base (APriori) contains the power distribution component interconnection model (for a single tile) encoded in Prolog. This knowledge base is used by the qualitative and quantitative fault detection, the diagnosis and test systems to identify the adjacent components to the component of interest in the circuit. As these other knowledge bases are general in nature, they do not contain a knowledge of the component interconnection in their rule structure.⁵¹ For completeness, the entire APriori knowledge base is listed here:

```
connections(x,y,z) :-  
connections(#intCond,y) ,  
connections(y,z).
```

```
connections(#intCond, #trCond1).  
connections(#intCond, #trCond2).  
connections(#intCond, #auxCond).
```

```
connections(#trCond1, #trCond1Load1).  
connections(#trCond1, #trCond1Load2).  
connections(#trCond1, #trCond1Load3).  
connections(#trCond1, #trCond1Load4).
```

```
connections(#trCond2, #trCond2Load1).  
connections(#trCond2, #trCond2Load2).  
connections(#trCond2, #trCond2Load3).  
connections(#trCond2, #trCond2Load4).
```

It will be noticed that the knowledge base is quite small and compact. Presently, the MFDS does not detect faults or diagnose components that are not associated with the power distribution and loads. Adding these additional EPS components will not complicate the connectivity model knowledge base excessively. The other knowledge

⁵¹ This is generally true. In a few cases however, the connectivity information is partially encoded within the rules to reduce the computational time required.

bases however can be expected to grow in size.

Queries to the APriori knowledge base from the other knowledge bases are made through the Prolog consult routine. The knowledge bases thus remain independent although accessible as required. The following is an example of a consult query

```
consult(connections(#trCond2, y), self aPriori).  
consult(connections(x, y, #trCond2Load3), self aPriori).
```

In the first consult Prolog will return a list containing all the loads attached to this transmit receive conditioner ((trCond2Load1) (trCond2Load2) (trCond2Load3) (trCond2Load4)). In the second, the list will contain the single result of one intermediate conditioner and transmit receive conditioner ((intCond trCond2)).

To summarize, the component interconnections are modeled within the APriori knowledge base. This knowledge base is accessed as required by the other reasoning systems of the MFDS through Prolog consult operators.

5.3 Diagnosis and Hypothesis Generation System

The diagnosis and hypothesis generation system (DiagnosisSystem) through its inferential component (DiagRules) further analyses the faults detected by the fault detection system's qualitative and quantitative reasoning components, returning where necessary to the telemetry and command arrays for additional data. The diagnostic rule base performs a slightly deeper analysis of the available data than that previously performed by the fault detection system.

Initially, the system identifies candidates for examination from the hypothesis lists generated by the qualitative and quantitative fault detection systems. This process starts from the source electrical components (intermediate conditioner) then proceeds through the lower level power distribution components (transmit receive conditioners and auxiliary conditioners) thorough to the loads.

Faults at lower levels of the component electrical path hierarchy will often appear as symptoms at higher levels. If a fault was indicated by the fault detection system at a given electrical component level, components at other levels may also be identified as candidates. The examination of the candidates may result in hypotheses being generated regarding specific component diagnoses. These hypotheses are asserted into the database (diagHypotheses) for later examination by the pruning system. An example diagnosis rule is; transmit receive conditioner - rule 5:

trCond(intCondId, trCond) :-

```

quanHypotheses(#intCond, intCondId, trCond, #currentError, dataArray),
is(actualCurrent, dataArray content at: 1) ,
is(expectedCurrent, dataArray content at: 2) ,
gt(actualCurrent, expectedCurrent) ,
connections(#intCond, trCond, aLoad) ,
storeData( diagData(#Candidate, #intCond, intCondId, aLoad) ) ,
storeData( diagHypotheses(#intCond, intCondId, trCond,
                          #highLoadPowerDemand)).

```

The rule reads as:

if

there exists a fault indicated in the quantitative fault detection hypothesis base
and,

the actual current is greater than the expected,

then

find the loads attached to this transmit receive conditioner (using the APriori or connections knowledge base) and,

identify these loads as possible candidates in the database and,

identify this transmit receive conditioner as having a high load power demand in

the hypothesis database.

The rule indicates that the power demand on the transmit receive conditioner is higher than expected (as indicated by the current).⁵² It is likely that fault lies with an unspecified lower level component (i.e. one or more of load1... load4). Note that this clause does not indicate a fault with the transmit receive conditioner. Instead, a refinement of the fault detected *currentError* is made resulting in the hypothesis *highLoadPowerDemand*.

In many cases the fault can be identified as isolated to the component presently being examined through the application of deeper knowledge of the components capability and expected operational characteristics. The following is an example from; transmit receive conditioner - rule 4:

```
trCond(intCondId, trCond) :-  
    quanHypotheses(#intCond, intCondId, trCond, #voltageError,  
        dataArray),  
    is(actualVoltage, dataArray content at: 1) ,  
    or( lt(actualVoltage, 11.9) , "normally tightly regulated to 12.0"  
        gt(actualVoltage, 12.1) ) ,  
    gt(actualVoltage, 10.0) , "below 10.0 V, loads go to stand by"  
    storeData( diagHypotheses(#intCond, intCondId,  
        trCond, #voltageRegulationFault)) .
```

The rule reads as follows:

```
if  
    there is a transmit receive conditioner with a voltage error and,  
    the voltage outside the range  $11.9 < v < 12.1$  and,  
    the voltage greater than 10.0,  
then  
    assert this transmit receive conditioner has a voltage regulation fault
```

⁵² As the trCond, in normal operation, has a tightly regulated output voltage, voltage errors do not appear unless the trCond has lost voltage regulation capability. See next example.

The above rule includes deeper knowledge of the transmit receive conditioner. That is knowledge that:

- The component normally has a very tightly regulated output voltage.
- If the voltage output varies even 0.1 volt from the expected value then, the transmit receive conditioner can be considered to have failed.
- If however the voltage drops lower than 10 volts then other faults may exist (open circuit or tripped circuit breaker for example) and these will be diagnosed by other rules.

Relying on the model to provide reference (expected) values to compare to the domain (actual) values reduces the need for deep knowledge significantly. The diagnosis knowledge base (DiagRules) is largely a shallow analysis of the faults, telemetry and command data. Only 10 % of the clauses of the knowledge base may be considered to contain deep knowledge of the electrical components as demonstrated in the preceding example. The remainder contain a shallow interpretation of the domain data.

The shallow rules within the diagnostic knowledge base (DiagRules) rely on the deep knowledge within the model for reference comparison data to the domain and on the comparisons already made by the fault detection system. The model provides reference voltage, current and status values that are dependent on the health of the components and operational state of the EPS (solar state, load on/off status etc.). The model does not now contain reference set points for comparison of certain faults (e.g. in the previous example, the reference values of 11.9 and 12.1 volts) however this could be easily added. This last change would remove all deep knowledge from the knowledge base and place it in a common knowledge structure, the model simulation.

The diagnosis and hypothesis generation system asserts hypotheses into a database directly for later filtering by the hypothesis space pruning system. Hypothesis generation is thus not a separate step but occurs through the action of the diagnosis rules.

5.4 Hypothesis Space Pruning System

Hypothesis space pruning (PruningSystem) reduces the hypothesis list resulting from the diagnostic system's actions to a minimal set of ordered hypotheses. The pruning system operates on the diagnosis hypotheses and a knowledge of the component interconnection only. Inputs from other sources (telemetry data etc.) are not required. The pruning system relies on knowledge of how certain faults can cascade (symptoms, faults and diagnoses) to lower level components and filters out those hypotheses that are in fact side effects of other component's faults.

The Pruning System starts by searching the diagnosis hypothesis database (diagHypotheses) for any hypotheses that identify intermediate conditioners as having faults. The source code that initiates this activity is - intCond rule 0:

```
intCond() :-  
    diagHypotheses(#intCond, intCond, #intCond, _),  
    intCond(intCond).
```

If the diagnosis hypotheses database contains intCond hypotheses, then the intermediate conditioner hypotheses in the database are examined further by the remaining rules in the pruning system rule base. Once supported hypotheses have been identified for all the intermediate conditioners having faults, this process is repeated for the remaining components (trCond, auxCond and loads).

The identification of supported hypotheses relies on the existence of higher level components having faults that propagate symptoms to down-stream components. If higher level components have been identified as having key faults then we may be able to discount the given component's fault hypothesis. An example for the transmit receive conditioner is:

A open circuit in an intermediate power conditioner can cause the components that it supplies power to (i.e. auxCond, both trCond and the attached loads) to experience symptoms of a similar nature. Thus if the intermediate conditioner is open circuit then we can most likely disregard these other diagnoses.

The actual Prolog clause that is used to perform this action for the transmit receive conditioner is - trCond rule 2:

```
trCond(intCondId, trCond) :-
    diagHypotheses(#intCond, intCondId, trCond, #openCircuit) ,
    diagHypotheses(#intCond, intCondId, #intCond, #lowCondPowerDemand) ,
    not( diagHypotheses(#intCond, intCondId, #intCond, #openCircuit)) ,
    not( diagHypotheses(#intCond, intCondId, #intCond, #breakerTripped)) ,
    storeData(pruneHypotheses(#intCond, intCondId, trCond, #openCircuit)) ,
    fail().
```

This Prolog clause reads as follows:

If

the trConditioner has been identified as being open circuit and,
the intermediate conditioner has a lower than expected power demand and,
the intermediate conditioner is not open circuited and,
the intermediate conditioner does not have a tripped breaker

then

assert this trConditioner as open circuit into the pruneHypothesis database.

In this example it will be noticed that all processing is performed at a symbolic level and that the knowledge structure is very shallow. In this case shallow reasoning implies that the knowledge base has a minimal knowledge of how the components interact beyond filtering certain lower level faults out in the event of higher level component faults. In the EPS application domain this approach appears to be effective in eliminating poorly

supported hypotheses while still identifying what will later be proven to be a correct diagnoses.

5.5 Hypothesis Validation System

The Hypothesis Validation System (ValidationSystem) selects a hypothesis from the pruneHypotheses database and coordinates several actions that are required to validate the hypothesis as being plausible. Both these actions are initiated through menu selections in the diagnostic system browser pane assigned to the validation system.

During the selection process the validation system selects the first hypothesis in the list found in the pruneHypotheses database.⁵³ To ensure that this candidate is not affected by later processing, it is asserted into the validCandidate database. The validation system will attempt to alter the configuration of the model to emulate the fault identified in the hypothesis suspected to exist in the domain. This is accomplished as follows:

- The candidate is selected through a menu operation,
- The validation process is then initiated through another menu operation. Control of the validation process is then passed to the meta level reasoner (MetaLevel) and its rule base component (MetaRules). These systems perform the following actions:
 - All hypotheses that reference the candidate are stored in a list for later reference (metaInitData),
 - The model is modified and the fault detection, diagnosis and hypothesis

⁵³ The pruneHypotheses database contains the ordered minimal list of supported hypotheses. The first element in the pruneHypotheses database is thus the hypothesis of greatest importance in this domain.

generation and hypothesis space pruning steps are repeated (through the action of Smalltalk methods within the ValidationSystem).

- All hypotheses that reference the candidate are again stored in a list for later reference (metaFinalData),
- The meta reasoning system examines the hypotheses that remain and notes this final list of hypotheses to the user. These hypotheses are examined to determine the errors in the reasoning process, the appropriateness in the selection of the candidate and others. In the present implementation, alterations made to the model are effective in modelling the faults in the domain. As such, the meta level reasoner is required to perform little in the way of model revision failure analysis. This does not permit exhaustive testing of the effectiveness of the meta level system.

The validation system provides Smalltalk methods that manipulate the model and initiate the analysis of the reasoning process. The validation system contains no inferential process as such and relies heavily on the analysis provided by the Meta Level System (MetaLevel) and its rule base. The metalevel reasoner is used to examine the results of the validation activity. In the present system, the validation system has been demonstrated to be effective in altering the model to represent the suspected domain faults. As a result of this, little demand is placed on the reasoning ability of the metalevel reasoner.

5.6 Test Case Selection System

The test case selection system (TestSystem) and its rule base (TestRules) select the appropriate test to be performed in the domain (and in some cases mimicked in the

model). The purpose of these tests is to provide supporting or refuting evidence *by modifying the domain* (and possibly the domain and model together) then observing the results. The individual test cases are found within the clauses of the TestRules class. The candidate hypothesis from the validation system is used as a key to identify and select the appropriate test to be performed during the verification step (in the present MFDS implementation test case selection is effectively a simple table look up⁵⁴). The selected test is stored in the Prolog database variable testCase.

In terms of the MFDS and the EPS, limitations of the domain do not permit all detected, diagnosed, pruned and validated hypotheses to be tested to conclusion. As the EPS domain has a finite number of fixed sensor points, the ability to obtain additional supporting or refuting information is limited. A result of these limitations is that not all validated hypotheses have associated test cases. This limitation could be alleviated through the use of additional sensor locations. The impact on the spacecraft design may however prevent this from occurring (i.e. design limits on mass, volume, telemetry bandwidth, etc.).

An example of a test case is presented in the following section in the discussion of the Verification System.

⁵⁴ For larger systems with many entries in the test case database, more advanced data base look up techniques may be needed. Alternately, Case Based Reasoning methods may be used to select and adapt an appropriate test case.

5.7 Hypothesis Verification System

The hypothesis verification system (VerificationSystem) provides links to the model, domain and the remainder of the MFDS to enable it to perform the actions identified in the test selected in the previous step. The verification system analyzes the results of performing the test and reports these to the user.

The test case selected previously contains the sequence of actions to be performed. During the verification step, identical actions are performed on both the model and the domain. The EPS domain however, is not as easily modified as the model⁵⁵ (faults are never intentionally injected, available data may be considerably more limited, etc). Reliability considerations require a test be devised that will impact the domain to the least extent possible. The application of the test must not induce other component's failures.

Each test contains the sequence of steps to be followed to provide supporting or refuting evidence for the hypothesis that is under test. The sequence will include one or more instances of the following steps in the appropriate order:

- record data relevant to the candidate under test into the identified data base. This includes all the hypothesis databases identifying the component in question.
- performing some action on both the model and the domain. This may include resetting breakers, altering the position of switches, etc.
- propagating the new model and domain data through the fault detection, diagnosis and hypothesis generation, and hypothesis space pruning steps of the Diagnostic Process.

⁵⁵ Although in this instance both the model and domain are simulations, the domain simulation will be manipulated in ways that are typical of hardware systems.

- examine the data before and following the test to determine the level of success.

A typical test sequence is noted in the following example. The fault scenario is as follows:

An identified intermediate conditioner in the domain is suspected as having a tripped circuit breaker. The hypothesis that the intermediate conditioner has this fault has been detected and diagnosed and finally validated. As a result of the actions of the validation system, the model now emulates and is experiencing the conditions associated with a tripped circuit breaker for this component (i.e. the model will be experiencing the fault that the domain is hypothesized to have). These faults will be removed by resetting the circuit breaker in both the model and the domain. The MFDS will again be run generating new data regarding the faults that exist in the domain. The results of the fault detection, diagnosis and hypothesis generation and hypothesis space pruning systems are again examined to determine the level of success in identifying the appropriate fault and removing (or isolating) the cause.

For this example, the test that was asserted into the testCase Prolog database by the test case selection system is from rule 1 for the intermediate conditioner:

```
(#RecordData, 1) ,
(#Model, #resetBreaker, #intCond, intCondNum, #intCond) ,
(#Domain, #resetBreaker, #intCond, intCondNum, #intCond) ,
(#Propagate) ,
(#RecordData, 2) ,
(#Evaluate, 1, 2) .
```

This case applies to an intermediate conditioner with an open circuit and reads as follows:

```
record the data existing in the hypothesis bases and,
reset the breakers in both the model and the domain and,
propagate the resulting model and domain data through the diagnostic system
```

(fault detection, diagnosis and hypothesis generation, hypothesis space pruning steps) and,
record the new data in the hypothesis bases and,
evaluate the results comparing the data in database 1 and 2.

The evaluation step compares the expected change in behaviour with the observed behaviour. If the test succeeded in restoring normal operation in the domain by resetting the circuit breaker, then the test action was a success. If this is not the case, then the newly generated data will be further examined in an effort to determine the cause of the discrepancy.⁵⁶

This approach to verification is applied to faults that occur in the remaining components of the EPS. For each component fault that can be verified,⁵⁷ a test case will be identified. The application of the test case will provide the supporting or refuting information necessary to verify the fault. Presently, the MFDS can test and verify open circuit and tripped circuit breaker faults for the intermediate conditioners, transmit receive conditioners and auxiliary conditioners.

5.8 Reasoning Control Mechanism

The reasoning *control* mechanisms in the MFDS are contained largely within the meta level reasoning system attached to the validation system and the evaluation component associated with the verification system. The system can be operated in an entirely

⁵⁶ In the present system the evaluation component reports to the user the level of success attained in the test. It does not reason about or perform a detailed analysis of the results. This additional functionality is expected to be added in later versions.

⁵⁷ Several faults can not be verified due to the limited number and location of sensors etc.

automated manner.⁵⁸ That is, in addition to the step by step operation by the user of the fault detection, diagnosis and hypothesis generation, hypothesis space pruning, validation, test selection and verification systems, these steps can be executed in order without interruption.

The control mechanism of the Smalltalk based components of the MFDS is procedural in nature, the same sequence of events occurs each time the module is accessed. This is not the case with the Prolog based reasoning systems. The input data to the reasoning components initiates the creation of candidate lists. The reasoning is thus guided by external events (the faults) that have occurred.

The *focus of attention* approach to identifying areas of interest is used and has proven effective in reducing the scope of the problem to a more manageable level. As noted in section 4.1.1, the focus of attention approach will reduce the computational load by the approximate levels:

Reasoning System	improvement
fault detection	120 times
diagnosis	5-8 times
hypothesis space pruning	5-10 times

Table 5-2 Improvement due to Focus of Attention

While the focus of attention approach is not a sophisticated meta level reasoning system, it has proven very effective in reducing the computational load.

⁵⁸ Although this is not yet implemented it is expected to be added shortly. Several portions of the software needed to implement such actions already exist.

Later in this document (section 7.2) the limitations of the Prolog implementation used for the MFDS will be discussed. This implementation of Prolog has a list size limitation of approximately 950 elements. Without the focus of attention approach, this boundary may be reached for complex multiple fault problems. Following the introduction of the focus of attention approach into the MFDS during its development, the list size limitation of this version of Prolog no longer proved to be a problem.

5.9 The Electrical Power System Model

The existence of a representative *model* is the key to any model based reasoning system. The knowledge structure that contains the model may take a variety of forms: rules, frames, procedural code, neural nets, etc. The model of a reasoning system applied to an electrical circuit may contain one or more of: electrical behaviour, component position, component connectivity, thermodynamic and/or other descriptive characteristics of the domain. The key to a good model is that it provide an accurate representation of the aspect of the domain that is of interest.

Correctly identifying the required level for the elements in the model is important to designing an effective yet minimum complexity simulation. As noted previously, the EPS has a finite number of sensors and, circuit probing actions are not possible. The practical limit of decomposition of the components of the EPS is the level at which sensory data is available. Lower level models add complexity and provide little benefit.

The EPS circuit model is created through the interconnected instances of electrical component models. The EPS is thus not modeled as a single entity but displays a

behaviour that is a composite of the behaviour of the component models and their interaction. The electrical component models are black box descriptions of the circuit elements that comprise the EPS (as noted in section 2) and are encoded in Smalltalk classes. The black box component definitions contain the describing equations that relate the dependent and independent parameters for the component. For example, a first order behavioural model of the solar panel is based on a piece wise continuous equation that describes the voltage / current characteristic curve of photovoltaic arrays:

Boundary values:

voltage	current
170 volts	0 amps
160 volts	25 amps
0 volts	30 amps

Describing equations:

voltage = $170 - ((10/25) * \text{current})$ for $0 < \text{current} < 25$
voltage = $160 - ((160/5) * \text{current})$ for $25 < \text{current} < 30$

Similar equations define the remaining components of the EPS. It should be recognized that the underlying behaviour of the solar panel is not needed to define its characteristic equations or anticipated behaviour. *The underlying mechanisms are not required once the boundary values and describing equations are available for the component.* For the solar panel example, this includes: the underlying photovoltaic cell construction, the series parallel connection of the cells, physical dimensions or other aspects of its construction.

A Smalltalk class (EpsComponent) defines the data structures and methods common to all components. Subclasses of this class define the EPS components solar panel, primary power conditioner, energy storage unit (battery, a group of cell packs), cell pack (a group of cells), cells, intermediate conditioners, transmit receive conditioners, radar loads and

auxiliary conditioners.⁵⁹

The electrical component models define the electrical characteristics of the circuit elements of the EPS only. Other aspects of the components are not modeled. The EPS components fall into three groups:

- primary power source (solar panel, primary power conditioner),
- storage elements (energy storage unit, cell pack, cell) and
- power distribution and loads (intermediate conditioners, transmit receive conditioners, auxiliary conditioners and the radar loads).

The update cycle, coordinated by an instance of the EpsSimulation class, requires that an electrical balance be attained between the circuit components. Initially, the power *available* from the primary power source and the power *demand* from power distribution and loads is compared. If power is available and not demanded, then this energy is available to recharge the storage elements (i.e. cells). If the reverse is the case, then the storage elements are called on to meet the power deficit. Throughout the update process the voltage and current levels will be defined and revised as the magnitude of the I^2R losses for the components are determined.

The EPS component models are coded directly in Smalltalk code blocks. The use of code blocks permits the behaviour of the component to be changed on a component instance by instance basis. This enables any instance of a component class to have significantly different behaviour than the default class definition. The redefinition of the component model is used when modelling certain component faults. This modelling approach will

⁵⁹ The EPS is described in Appendix A. Additional detail on the model is presented in Appendix A.2 and the component models in Appendix A.3. The EPS circuit block diagram is illustrated in Figure 2-1.

also prove useful in an environment where manufacturing tolerances cause slight variations in behaviour between components of the same class.

6.0 Testing the Multiple Fault Diagnostic System

The Multiple Fault Diagnostic System was developed over a several month period. During the design, development and implementation phases, testing was performed on a continual basis. As new functionality was added, it was tested and evaluated. The success or failure of early developments spurred revision and changes to preliminary designs and later developments. Many of these early tests were performed only at the most rudimentary level and were intended to check the basic operation of the component or system. These early tests are not documented here.⁶⁰

As the MFDS continued to be developed, the model and domain simulation became available to provide simulated spacecraft telemetry data. In these later stages, the MFDS took on the form and function described in section 5. Eventually, the system became sufficiently complete to demonstrate the appropriateness and utility of the MFDS approach. These tests did not provide a formal system level software validation and verification, nor was that the intent. The tests performed demonstrated the level of capability of the reasoning systems so far attained and the utility of the approach. This section summarizes these later tests and the system's response. A more complete description of the tests and the test results is presented in Appendix B.

Full MFDS functional tests were only performed following integration of the components into near complete modules (fault detection system, diagnosis and hypothesis generation system etc.). Initial system level testing revealed several errors and omissions in the knowledge bases. The knowledge bases were revised to alleviate these problems. It should

⁶⁰ Several Smalltalk classes however have a class method called *example* that creates an instance of the class and exercises it providing an indication of some of the functionality offered by the class.

be recognized however that the knowledge bases while largely complete remain at a prototypical level. Several faults can not be simulated in the domain nor adequately addressed in the MFDS. Those faults that can be simulated are noted in Appendix A Section A.3 and in tables A-1, A-2 and A-3.

The tests performed on the completed MFDS fall into the following groups:

- i tests of the EPS simulation (used for the model and domain) to generate both normal and failure mode data,
- ii tests of the functionality and completeness of the reasoning systems (fault detection, diagnosis and hypothesis generation, etc.),
- iii tests of the user interfaces (MFDS browser, model and domain browsers, data windows, etc.).

The functionality of the simulator that was tested in group i is listed here by component:

Intermediate conditioner: normal operation, open circuit, tripped circuit breaker, loss of efficiency.

Transmit receive conditioner: normal operation, open circuit, tripped circuit breaker, loss of efficiency.

Auxiliary conditioner: normal operation, open circuit, tripped circuit breaker, loss of efficiency.

Radar loads: normal operation, loss of efficiency (electrical), loss of efficiency (RF), open circuit, stuck in the on position and stuck in the off position.

Solar panel: normal operation.

Primary power conditioner: normal operation.

Energy storage unit: normal operation (faults are modeled in the cells).

Cell pack: normal operation (faults are modeled in the cells).

Cell: normal operation, loss of charge, short circuit, open circuit.

Tests of the functionality and completeness of the knowledge bases were performed to demonstrate the capability of the MFDS in addressing each component's faults as identified in the above list. Several of these tests were performed in parallel demonstrating the multiple fault capability of the MFDS.

The user interface was tested, evaluated and extended throughout the above development process. The present user interface in many cases provides the developer with data not normally presented to end users. This additional data system the development and debugging the software and knowledge structures. The abundance of data however often masks the few key values of interest. Data is presented to the user with a minimum of semantic aids and in many cases simple Prolog database dumps are performed.

The user interface developed for this work does not provide several of the features now common in AI based systems (natural language, explanation generation, others). The intent of the MFDS user interface is to provide a display of the reasoning systems' data and hypothesis bases and to provide access to the Smalltalk debug facilities. The user interface meets these goals. It is not expected that the present user interface will find application on-line in a spacecraft operations centre however.

In section 2.4, a failure mode and complexity analysis of the EPS domain was presented. The analysis indicated that some 6540 single and 34200 two fault cases were detectable for the power distribution components of the EPS. A single tile however, contains 4 unique classes of EPS components (intermediate conditioner, transmit receive conditioner, auxiliary conditioner and radar load). The tile contains 2 transmit receive conditioners, 8 radar loads and one each of the remaining components for a total of 12. The MFDS addresses faults that occur in these components on an individual basis. Duplicate components require only the reapplication of the existing rules for that component class. Based on this, there exist 52 single faults that can occur on a tile.⁶¹ Examining the tile

⁶¹ the tile contains: 1 intermediate conditioner with 3 possible faults, 1 auxiliary conditioner with 3 faults, 2 transmit receive conditioners with 3 faults each and, 8 loads with 5 faults each. Multiplying this out results in 52 single faults on each tile.

from a component class perspective results in 14 unique component faults possible.⁶²

The ability of the MFDS to detect and diagnose each simulated fault for each *class* of component was tested. For example, the transmit receive conditioner class of EPS component has 3 possible faults (open circuit, tripped breaker, loss of efficiency). The MFDS was tested against these three faults to determine its capability. Similarly, the 5 faults associated with the radar loads, 3 for the auxiliary conditioner and 3 for the intermediate conditioner were also tested. The specific EPS components for these tests were selected randomly and all tests for a given component class were performed in parallel on other instances of the same component (e.g. intermediate conditioners number 2, 6 and 10 were selected to perform the intermediate conditioner tests in parallel). The test results are summarized in table 6-1.⁶³ All tests performed were completed successfully.

Several spot checks were performed to demonstrate the ability of the MFDS to address multiple fault cases. These tests were performed both on separate tiles (intermediate conditioners and related lower level components) and on the same tile (within the limitation specified in section 2.4). The MFDS completed many of these tests successfully. Limitation in the knowledge bases and the configurability of the EPS domain however limited the level of testing possible. Available time did not permit testing the more than a small sample of the number of possible multiple fault cases that exist.

⁶² Calculated as 3 faults for each of the 3 conditioner classes and 5 for the radar load class totaling 14.

⁶³ Tables B-1a and B-1b in Appendix B provide additional detail of the tests performed and note the hypotheses reported by each of the reasoning systems (one or more hypotheses may be reported for any given fault).

component/ simulated fault condition	FD QL	FD QN	diag	prun	val	test case	ver
auxiliary conditioner							
open circuit	x	x	x	x	x	x	x
tripped breaker	x	x	x	x	x	x	x
loss of efficiency	x	x	x				
intermediate conditioner							
open circuit	x	x	x	x	x	x	x
tripped circuit breaker	x	x	x	x	x	x	x
loss of efficiency	x	x					
radar load							
stuck on	x	x	x				
stuck off	x	x	x				
loss of efficiency (electrical)	x	x	x				
loss of RF efficiency	x	x	x				
open circuit	x	x	x	x	x	x	x
transmit receive conditioner							
open circuit	x	x	x	x	x	x	x
tripped breaker	x	x	x	x	x	x	x
loss of efficiency	x	x	x				

Table 6-1 Single Fault Test Summary

Notes for table 6-1:

x reasoning system addresses this fault successfully. May however be identified under another name or a more general hypothesis than the fault simulated (see appendix B)

' ' (blank) this fault not addressed at all by the present knowledge base

FD QL qualitative fault detection

FD QN quantitative fault detection

diag diagnosis and hypothesis generation system

prun hypothesis space pruning system

val validation system

test case test case selection system

ver verification system

7.0 Analysis and Discussion of the Test Results

It is believed that the Multiple Fault Diagnostic System, the EPS domain simulation, the EPS Model, the browsers and list panes meet the performance goals and requirements of the domain (section 2 and Appendix A), the Diagnostic Process (section 3) and the MFDS design (section 5). As noted in the Section 6 the functionality of the MFDS reasoning components has been demonstrated to work *to the extent that the knowledge bases are specified*. While the MFDS does not yet address all possible faults within the EPS, several important concepts have been successfully demonstrated. Notably the use of a simulation in model based reasoning, a software implementation of the Diagnostic Process, learning in the form of model revision and maintenance, the reliance on easily maintained shallow based reasoning systems, and several others (see section 8.0 Conclusions).

The MFDS can successfully fault detect, diagnose, prune, verify, identify a test case and validate then *recover* from small number of fault scenarios.⁶⁴ The fault scenarios that can be recovered from in this domain are tripped breakers for the power conditioners. By recovery it is implied that the action performed during validation will not only provide further supporting evidence of the fault but will also remove the detected fault in the domain. Faults scenarios where recovery is possible are:

- Intermediate conditioner - breaker tripped
- Transmit receive conditioner - breaker tripped
- Auxiliary conditioner - breaker tripped

It should be noted that the events that caused the tripped breaker are assumed in this case

⁶⁴ Note that the goal of the MFDS is not to identify fault recovery actions but to detect and diagnose faults. Recovery from fault conditions is normally left to the spacecraft operator or other software systems to implement.

to be transient (or one time) events. These may be due to solar flares, static discharge and other events that caused a spike tripping the breaker but, caused no other damage to the component(s). In the event that the breaker was tripped due to other reasons (e.g. high power demand by down stream components), resetting the breaker will likely only cause it to trip again. This will be indicated as a breaker reset fault by the MFDS.

The detection of a *breaker reset fault* was demonstrated successfully for all classes of components equipped with circuit breakers (intermediate conditioner, transmit receive conditioner and auxiliary conditioner). The detection of this fault required the application of the MFDS in two passes as the fault of a tripped breaker was first detected, then changed to a breaker reset fault following the failed attempt to reset the breaker. To perform this test the following test scenario was used:

- the down stream components of one intermediate conditioner of the domain were set to have very low efficiencies. The low efficiency of the lower level component caused an excessively high current causing the intermediate conditioner's circuit breaker to trip.
- the MFDS was run successfully detecting, diagnosing, pruning, verifying and selecting a test. The test candidate identified from the above actions was: tripped circuit breaker. (The fault causing the breaker to trip can not be detected as supporting data is masked by the higher level component's tripped breaker.)
- while validating the hypothesis, tripped circuit breaker, the test selected was applied. This test required resetting the circuit breaker and reapplying the MFDS actions from the fault detect step through to the verification step.
- when the validation system analyzed the data resulting from the verification step, it was recognized that the breaker did not in fact reset. This new fault caused the validation system to identify to the user the new hypotheses intermediate

conditioner breaker reset fault and open circuit and retract the previous hypothesis breaker tripped.

What is of interest in this example is that the MFDS detected and diagnosed the tripped circuit breaker and verified this hypothesis as plausible given the data presented. When additional data became available, the previous hypothesis was retracted and replaced by another.

In the event that the conditioner has an open circuit,⁶⁵ no corrective actions are possible. An actual open circuit can not be recovered from as the internal components and operation of the conditioner expected to be irrecoverably altered. The MFDS however assumes that the status indicator from the circuit breaker is reading incorrectly and responds by attempting a breaker reset. As in the previous scenario, the validation system will attempt a breaker reset. A failed breaker reset in this case however will not result in the breaker reset fault hypothesis being generated as a tripped breaker was never indicated previously by the MFDS. Instead the verification system will note the failed recovery and identify the failure to the user. The open circuit fault in this case was provided with further supporting evidence during validation and remains the believed hypothesis.

The MFDS does not presently have the capability of calculating the new operational efficiencies that a component may be operating under (based on the telemetry data). While this limits the ability for the validation and verification systems to examine efficiency related hypotheses, the MFDS can successfully detect and diagnose the presence of these errors for the following components: transmit receive conditioner, auxiliary conditioner and radar loads. The present system however, does not have the

⁶⁵ That is, the power output is zero however, the breaker status line indicates the breaker has not tripped.

ability to resolve which component is experiencing the loss of efficiency.

Loss of efficiency faults are detected by examining the power output from the higher level component and contrasting this with the power output from the lower level component. Consider the example:

- comparison of one intermediate conditioner's expected (model) and actual (domain) data indicates that the power output of the intermediate conditioner is higher than expected.
- The power output from the auxiliary conditioner and the two transmit receive conditioners are at the expected levels (i.e. expected and actual values match).
- the fault detection and diagnosis systems will indicate that a conditioner has a loss of efficiency. With the available data, which conditioner (the auxiliary conditioner or one of the transmit receive conditioners) can not be determined.
- further tests in the domain may be performed to acquire sufficient additional data (not presently possible). These tests would require changing the load demand by the transmit receive conditioners and auxiliary conditioner one by one then observing the results.
- by observing the change in sensor readings, the identification of the faulty conditioner should be possible. For example, say that transmit receive conditioner 1 has a loss of efficiency and the radar loads attached to this conditioner are on, If the intermediate conditioner is observed have a current *error* of 1 amp higher than expected.
then switching off all radar loads attached to this transmit receive conditioner can be expected to reduce the power demand on the intermediate conditioner such that the *error* in current is less than the 1 amp level previously observed.

Problems that arise from this approach are that the radar loads are not normally controlled in this way. While it may be possible to perform this test, no recovery action is possible. It is expected that realistic spacecraft operations centre approach is to note minor losses of efficiencies and watch for further degradations. Significant losses of efficiency will manifest themselves in tripped circuit breakers with the resulting loss of the relevant components for use by the Radar Imaging System.

Several problems arise in performing tests such as these in the real (rather than simulated) EPS domain. For example, the loads attached to the auxiliary conditioners are presently assumed to be constant power devices. These loads are in fact many and varied. They include: heaters, reaction control wheels, control electronics and several other constant and pseudo-random power demand devices. The *average power* requirements of these loads are what is now modeled as a *constant power* sink. While this may be acceptable for this demonstration system, it may prove a hinderance in practice. The time domain nature of these loads is expected to be difficulty to predict and thus model (other than by stochastic means). While these loads represent less than an estimated 15% of the total power of a radar spacecraft, their variability may impact the accuracy of model based reasoning approach demonstrated.

An alternate source of data that is not presently exploited in the MFDS is the Radar System. It is expected that the absence of a receive signal from any of the radar loads may indicate failures in the radar loads or other EPS components providing power to the loads. It is not apparent at this time that the Radar System will supply such information to other systems such as the MFDS. The examination of the radar image on the ground may provide an indication of such failures.

7.1 Limitations of the Application Domain

The verification of many faults requires that additional data be made available by the Domain system. Where possible, this additional data is acquired through the application of the test cases that have been selected. The present design of the EPS and the anticipated reluctance of spacecraft operators to implement actions that are not part of the radar imaging mission hinders this process.

The EPS demonstration domain has a minimal degree of control and a fixed number and location for the sensors. Within the EPS circuit, probing actions are not possible. The present design may permit on/off control of the radar loads however, it is not apparent that this is possible outside the control of the payload Radar System. While the EPS circuit is well instrumented, the internal state of the components is not available. The DC-DC converters used in the intermediate conditioners, transmit receive conditioners and auxiliary conditioners have several modes of failure. The ability to resolve these modes of failure however is low. Several failure modes can only be grouped into quite broad categories (open, loss of efficiency, tripped circuit breaker, etc.). While limitations of the telemetry bandwidth, sensor mass and volume remain for spacecraft designers, the demands of increasingly complex systems with high reliability requirements may result in a change in priorities favouring more automation and increased sensor counts.

The high reliability requirement of the spacecraft community can not be overlooked. Spacecraft are often expected to operate over a period of up to 20 years without maintenance actions being performed. A spacecraft is expected to maintain a high level of operational capability throughout its life. This high reliability requirement results in a very slow and gradual evolutionary change from one spacecraft generation to the next.

A conservative approach is pervasive in all levels from design through manufacturing and operation. As an example, spacecraft operations personnel are reluctant to switch components on or off unnecessarily due to possible failures making any transition a permanent event. It is expected that as space systems become larger and more complex that space industry accept automated approaches to operations similar to that demonstrated by the MFDS.

7.2 Notes on the Implementation Environment

The MFDS was implemented in Digitalk's Smalltalk V/286 [Digitalk 1988a] and their implementation of Prolog [Digitalk 1988b]. Smalltalk V/286, while not as full featured as other Smalltalk development environments (notably Smalltalk 80 from Parc Place), proved fully satisfactory in meeting the object oriented programming needs of this work. The programming environment tools proved more than adequate in providing effective editors, code browsers, debuggers etc. The software tutorial and manual generally proved useful in finding the various classes and methods. Several language and reference manuals were used during the development of the MFDS [Digitalk 1988a, 1988b; Clocksin and Mellish 1982; Goldberg and Robson 1985].

The object oriented programming paradigm of Smalltalk proved ideal for the MFDS. This was particularly true for the simulation used for the model and domain. In this case 9 unique component classes were defined then a simulation was construct resulting in some 1548 component instances. The locality of the data to the individual instances negated the need to define each separately, significantly easing the programming burden.

The Prolog implementation offered free with the Smalltalk V/286 language proved effective for experimental use although somewhat limited as a final implementation version of Prolog.⁶⁶ The advantage of this version of Prolog is that it is fully integrated into the Smalltalk V/286 environment. Message sends are fully supported between the two languages. Accessing Smalltalk from Prolog in fact becomes a necessity as much of the expected functionality of Prolog is only available through message sends to Smalltalk. This version of Prolog is very modular enabling several knowledge bases to exist concurrently with little overhead.

Problems with Digitalk's Prolog will likely prevent it from becoming much more than an interesting prototyping environment. The basic data structure of the Prolog implementation is the Smalltalk class List added as part of the Prolog installation. A problem arises as a result of the way that List was implemented. Lists grow as recursive expansion of the list tail (as in Prolog, Lisp etc.). While this is expected from a programming conceptual view, it also appears to be the way that List was implemented in the Smalltalk environment. The time required to access lists is linearly proportional to the length of the List. Accessing latter elements of long lists is slow due to the number of elements that must be traversed before reaching the element of interest. If the list is over approximately 950 elements in size, stack overflow can be observed with a debug walk back window popping up.⁶⁷ If this version of Prolog is to be used for larger systems, the underlying storage structure of List should be implemented using the Array, Dictionary or similar Collection subclass. Tests indicate that the maximum size of a collection (and its

⁶⁶ This does not imply Prolog is not effective, only that this particular version was somewhat limited in its capability.

⁶⁷ Note: This is not a physical *memory* limitation per say but a limit in the maximum stack size offered by Digitalk for Smalltalk and Prolog. Stack overflow was not observed to occur under any other circumstances.

subclasses) is limited only by the available free memory. Collection size limitations can then be resolved through the installation of additional RAM. This will enable lists to use the more effective storage mechanisms of Smalltalk.

Prolog clauses are not recognized by the debugger and as such several of the features of the debugger are not available (setting break points, examining the point where failure occurred etc.). A minor patch was written to list the Prolog source code (for the classes that use Prolog) on the debugger edit pane however the above problems remain. It was also noticed that continuing execution following a Prolog *halt()* statement did not necessarily result in expected data being generated following the 'resume' or 'restart' debugger menu selections. It appears that program stack operations are not adequately stored (or possibly retrieved) when execution is passed to (or returned from) the debugger. This may be an additional failing in the integration of Prolog and Smalltalk.

The code generated by the Prolog compiler did not appear to be efficient with regard to execution times. The execution times observed for Prolog are considerably longer than expected. As timing considerations are a secondary consideration of this work, computational performance enhancements were for the most part not implemented.

Smalltalk source code is considerably more compact than Prolog for equivalent functionality (where this is possible). Prolog source code represents 25% of the total page count of the MFDS⁶⁸. While the reasoning systems are a key part of the MFDS, they represent the smaller portion of the total software development. The use of an object oriented system with added reasoning capability is believed to be more effective than the

⁶⁸ The commented source code for the MFDS is 184 pages in length (10 point Times Roman single spaced). This is distributed as: 138 pages Smalltalk and 46 pages of Prolog.

reverse approach (e.g. object oriented Prolog).

7.3 Notes on the Implementation

The Smalltalk/Prolog implementation of the MFDS has several areas that could be improved. These are centred largely on reducing the time required for program execution and improving the user interface.

To optimize the MFDS to reduce the time required for program execution will require a more formal analysis of the program 'hot spots' than is possible here. However, several observations made while developing and testing the system may lead to a more effective implementation. These observations lead to the suggestions that follow:

- The fault detection system components performing both qualitative and quantitative fault detection may be more effectively accomplished through set operations. The change from Prolog inferencing to Smalltalk set operations will not affect the functionality of these components and is not expected to be difficult to implement. Smalltalk's Collection class and its subclasses (including Set) appear to be quite efficient in their implementations. Collection comparison methods appear fast and efficient.
- Several of the remaining Prolog based components could also be converted to Smalltalk. The disadvantage of this approach is that knowledge base maintenance may become more difficult to perform.
- The Simulation (i.e. model and domain) presently performs a recalculation of all component values with every update. This is performed even if the components operational environment does not change (e.g. solar conditions for the solar panel,

on/off state for the loads etc.). By carefully observing changes that occur and only updating when required considerable time could be saved.

- Performance improvements may also be realized by recodeing the system in another language (C++, Ada, etc.). It is not apparent that the ease of program development in Smalltalk is also available in these alternate languages however.
- An alternative to Prolog may be found in a current Canadian Space Agency R&D project. The goal of the project is to provide the Smalltalk programmer an advanced reasoning system development environment. Presently, the rule based component of this new environment implements a variation of the CLIPS language with Smalltalk environment enhancements.

The Simulation text and graphics panes provide acceptable means by which to interact with the domain and model. MFDS Browser presently provides a text based listing of the results of the various inferential processes. The user interface for the MFDS can be improved through several changes and additions:

- The addition of graphical displays flashing hypothesized failed components on the Model graphics pane, bar charts of expected verses actual sensor readings,
- An explanation system that better describes the reasoning process would also be advantageous,
- An indication of the degree of completion in the reasoning process.
- The hardware used proved adequate in presenting the data in a R&D environment. In an eventual application domain, a high resolution colour display (1280x1024) may prove beneficial.
- Computational performance improvements may be realized through faster processing hardware. A 486/50 Mhz can be expected to be approximately 75-80% faster than the 386/33 Mhz computer used.

- The Digitalk V/286 environment while proving fully adequate in developing this system appears to be superseded by the Microsoft Windows 3 based version Smalltalk V/Win. This later environment is known to be approximately 15-20% less computationally efficient than Smalltalk V/286. These losses in efficiency can be recovered through faster processors as noted above.

8.0 Conclusions

The work presented in this thesis touched on several areas relevant to AI based diagnostic applications in engineering systems. Contributions made in model based diagnosis and diagnosis in general are:

- The concept of a *Diagnostic Process* defined as several discrete steps from data acquisition through testing the fault hypothesis against the domain.
- The implementation of the Diagnostic Process in separate cooperating reasoning systems each performing the individual steps.
- The demonstration of the Diagnostic Process using *model based reasoning* to detect and diagnose faults on a spacecraft electrical power system.
- The use of a *simulation* that contains nearly all the deep knowledge of the domain as the *model*.
- The ability of the MFDS to learn the changing and evolving behaviour of the Domain and to modify the model of the domain to mimic the domain.
- The knowledge bases used in the reasoning process are constructed largely from shallow knowledge of the domain.
- The verification of candidate hypotheses on the model before testing in the domain environment.
- The use of the object oriented programming language *Smalltalk* to develop the simulation and all non-inferential code.
- The use of *Prolog* as the knowledge base development language. Prolog was used for all inference based systems development.
- The Multiple Fault Diagnostic System was demonstrated on a complex domain, a spacecraft Electrical Power System with 1548 discrete components, 3,456 state

variables⁶⁹ and more than 40,000 possible detectable faults.

Each of the above issues have been addressed to some extent in the literature (as noted in section 3). The combination of the above techniques and approaches is believed to be unique to the Multiple Fault Diagnosis System.

The formulation of a problem solution initially involves identifying the goals and requirements to be placed on the resulting system. These requirements and goals are broken down identifying sub-goals and lower level requirements until manageable modules are identified. The *Diagnostic Process* presented in this thesis identifies a solution methodology that is an extension of what has been used in the spacecraft operations control centres (see Appendix A.7). This Diagnostic Process approach provides not only a minimal list of diagnoses but diagnoses that have been validated against the model of the domain as being plausible. The automated validation of the hypothesis is expected to provide the users with more confidence in the reasoning process than alternate approaches. Finally, the test case selection and verification steps are not intended to remove responsibility from those presently performing this task but to assist them in completing their work.

Model based reasoning systems have been demonstrated in several domains. These systems typically used models encoded in rules, frames or other traditional AI data structures. Recently though there is a recognition that simulations may bring a significant contribution to AI based systems [Widman and Loparo 1989]. The computer based simulation software field has several tools and techniques that may provide a better means

⁶⁹ Combined telemetry and command data array size when duplicates are removed (intermediate, transmit receive and auxiliary conditioner breaker states are found in both)

by which to encode deep knowledge of the domain. The Multiple Fault Diagnostic System demonstrated that this is indeed possible for fault detection and diagnosis applications.

The model used in this system was developed using very traditional software simulation techniques. The model was constructed through the use of Smalltalk objects that were used as building blocks. Objects defined the behaviour of individual electrical circuit components of the spacecraft Electrical Power System (EPS). Once each component's object definition existed, instances were made as necessary to represent all the elements in the circuit. Update of the model followed a two pass power demand / available sweep through all components.

The knowledge bases used in the reasoning systems contained largely shallow knowledge of the domain. It is postulated that by constructing the knowledge bases with shallow knowledge eases the knowledge acquisition / knowledge maintenance tasks significantly. Knowledge of the domain contained in the model is easily maintained by virtue of the modularity and locality of the descriptions of the constituent parts of the simulation.

Smalltalk 7286 [Digitalk 1988a] was used as the primary implementation language. The inherit locality of data and object oriented nature of Smalltalk proved ideal in encoding the model and proved very useful in writing the remaining portions of the code. The windowing environment offered in Smalltalk proved fully adequate in developing the user interfaces.

Prolog was used to develop the reasoning mechanisms of the MFDS. The implementation of Prolog used [Digitalk 1988b] proved adequate but computationally quite slow. It is felt that an alternate version of Prolog conforming more closely to standard Prolog [Clocksin

and Mellish 1982] or some other rule based environment may prove more appropriate in the future. Alternatively, other reasoning techniques may prove fruitful. It is felt that as the knowledge structures are quite simple, more procedurally based techniques may be effective. The fault detection system for example may be better implemented through set operations. In this case expected and actual telemetry would be compared directly as sets.

References

- Adamovits, Peter J. 1992. "Model Based Reasoning Applied to the Diagnosis of Spacecraft Electrical Power System Faults: Software Description and Source Code Listings" Carleton University, Systems and Computer Engineering, SCE-92-15 268 pages.
- Adamovits, Peter and Jackson, Eric. 1991. "Spacecraft Electrical Power System Fault Detection/Diagnosis and Resource Management". Machine Intelligence for Aerospace Electronic Systems, AGARD conference proceedings 499, September Pages 6-1 - 6-11.
- Adamovits, Peter J. 1990. "Multiple Fault Diagnosis of Spacecraft Electrical Power Systems". Sixth CASI Conference on Astronautics, Ottawa, November, pages 68-76.
- Adams, Thomas L.; D'Ambrosio, Bruce; Fehling Michael R.; Schwartzberg, Sonia; Barton, John and Kaiser, Don. 1985 "An Artificial Intelligence Approach to Autonomous Power System Maintenance and Control". Proceedings of the 20th Intersociety Energy Conversion Engineering Conference, SAE P-164, pages 1.472-1.478.
- Anderson, J.B. and Nystrom, S.D. 1985. "Expert Software Design for Detection / Classification of Performance Anomalies in a Large Scale System", Proceedings of Robotics and Expert Systems-1985, Instrument Society of America, NASA/Johnson Space Centre June 27-28, pages 265-271.

- Baker, Carolyn G. and Marsh, Christopher A. 1990. "Failure Diagnosis and Impact Assessment Prototype for Space Station Freedom", Proceedings of the 1991 Goddard Conference on Space Applications of Artificial Intelligence, 1-2 May, pages 97-111.
- Baruah, Arati B.; Campiche, James B. 1986. "An Expert System for Electrical Power Automation on a Spacecraft", Proceedings of ROBEX'86 The Second Annual Workshop on Robotics and Expert Systems, NASA/Johnson Space Centre. June 4-6. Instrument Society of America. pages 317-325.
- Clocksin, W.F. and Mellish, C.S. 1982. "Programming in Prolog", Springer-Verlag, 279 pages.
- Collins. 1982. "The New Collins Concise Dictionary of the English Language", William Collins Sons and Co. Ltd. Glasgow.
- Davis, Randal. 1984. "Diagnostic Reasoning Based on Structure and Behaviour" Artificial Intelligence, Elsevier Science Publishers, Volume 24, Pages 347-410.
- Digitalk. 1988a. "Smalltalk/V 286 Tutorial and Programming Handbook", Digitalk Inc., 561 pages.
- Digitalk. 1988b. "Prolog.Doc", (Digitalk Prolog for Smalltalk/V operation manual), Text file provided with Smalltalk/V286 release. Digitalk Inc, 16 pages.
- Doyle, Jon. 1979. "A Truth Maintenance System", Artificial Intelligence 12 Elsevier

Science Publishers, Pages 231-272.

Eatock, Brian C. 1990. "Progress on DND's Space-based Radar R&D Project", Sixth CASI Conference on Astronautics, Ottawa, November, pages 452-463.

Fesq, Lorraine M. and Stephan, Amy. 1989. "On-Board Fault Management Using Modelling Techniques", Proceedings of IECEC-89.

Genesereth, Michael R. 1984. "The Use of Design Descriptions in Automated Diagnosis" Artificial Intelligence 24, pages 411-436.

Goldberg, Adelle and Robson, David. 1985 "Smalltalk-80 The Language and its Implementation" Addison-Wesley Publishing 716 pp.

Goyal, Shri K.; Prerau, David S.; Lemmon, Allan V.; Gunderson Allan S. and Reinke, Robert E. 1985. "COMPASS: An Expert System for Telephone Switch Maintenance", Expert Systems, Vol 2, No 3, July, pages 112-126.

Hamscher, Walter and Patil, Ramesh. 1989. "Model-Based Diagnosis" course notes from a 1/2 day course at the Eleventh International Joint Conference on Artificial Intelligence, IJCAI, 21 August 1989, 159 pages.

Hamscher, Walter C. 1991. "Modelling Digital Circuits for Troubleshooting", Artificial Intelligence 51, Elsevier Science Publishers, Pages 223-271.

Hayes-Roth, Barbara. 1985. "A Blackboard Architecture for Control" Artificial

- Intelligence 26 Elsevier Science Publishers, Pages 251-321.
- deKleer, J. and Williams B.C. 1987. "Diagnosing Multiple Faults". Artificial Intelligence 32(1), Elsevier Science Publishers, Volume 32, Pages 97-130.
- deKleer, Johan. 1986a. "An Assumption Based TMS". Artificial Intelligence 32(1), Elsevier Science Publishers, Volume 28, Pages 127-162.
- deKleer, Johan. 1986b. "Problem Solving with the ATMS". Artificial Intelligence 32(1), Elsevier Science Publishers, Volume 28, Pages 197-224.
- Krishnamurthi, Murali; Mayer, Richard J. and Friel, Patricia G. 1986. "Robot Fault Diagnosis Using Deep and Shallow Modelling Approaches", Proceedings of Robotics and Expert Systems-86 Conference, Instrument Society of America, pages 269-278.
- Kuipers, Benjamin. 1986. "Qualitative Simulation" Artificial Intelligence Volume 29 Elsevier Science Publishers, pages 289-338
- McAllister, David A. 1980. "An Outlook on Truth Maintenance", MIT AI Laboratory Memo 551, MIT. (available from NRC on microfilm)
- McDermott, Drew. 1991. "A general framework for reason maintenance", Artificial Intelligence 50 Elsevier Science Publishers, Pages 289-329.
- McDermott, Drew; Doyle, Jon. 1980. "Non-Monotonic Logic 1", Artificial Intelligence

13 Elsevier Science Publishers, Pages 41-72.

Momoh J.A. and Zhang Z.Z. 1990. "Knowledge Based and Integrated Monitoring and Diagnosis in Autonomous Power Systems" Proceedings of the 1990 Goddard Conference on Space Applications of Artificial Intelligence, 1-2 May, pages 191-199.

Moodie, M.H. and Maskell, C.A. 1989. "Electrical Power Distribution on Space based Radar Satellites" IEEE AES Magazine, November.

Reiter, Raymond. 1987. "A Theory of Diagnosis from First Principles", Artificial Intelligence 32(1) Elsevier Science Publishers, Pages 57-95.

Ringer, Mark J.; Quinn, Todd M. and Merolla, Anthony. 1990. "Autonomous Power System Intelligent Diagnosis and Control", Proceedings of the 1990 Goddard Conference on Space Applications of Artificial Intelligence, 1-2 May, pages 153-167.

Rothenberg, Jeff. 1989. "The Nature of Modelling", Artificial Intelligence, Simulation and Modelling, Widman, Lawrence E.; Loparo, Kenneth A. and Nielsen, Norman R. editors, John Wiley and Sons, pages 47-74.

Russell, Stuart and Wefald, Eric. 1991. "Principles of meta reasoning", Artificial Intelligence volume 49, Elsevier Science Publishers, Pages 361-395.

Shortliffe, Edward Hance. 1976. "Computer-Based Medical Consultations: MYCIN"

Elsevier Science Publishers, 264 pages.

Tang, Y.K. and Wetzel C.R. 1990. "NASA Ground Terminal Communication Equipment Automated Fault Isolation Expert System", Proceedings of the 1990 Goddard Conference on Space Applications of Artificial Intelligence, 1-2 May, pages 139-145.

University of Chicago Press. 1982. "The Chicago Manual of Style" Thirteenth Edition, Chicago, 738 Pages.

Walters, Jerry L.; Petrik Edward J.; Roth, Mary Ellan; Van Truong, L.; Quinn, Todd and Krawczonek, Walter M. 1990. "Autonomous Power Expert System", Proceedings of the 1990 Goddard Conference on Space Applications of Artificial Intelligence, 1-2 May, pages 147-156.

Weeks, David J. 1987. "Artificial Intelligence and Space Power Systems Automation", Proceedings of the third Conference on Artificial Intelligence for Space Applications, 2-3 November, pages 109-113.

Weiss, Sholom M. and Kulikowski, Casimir A. 1984. "A Practical Guide to Designing Expert Systems. Rowman and Allanheld, 174 pages.

Widman, Lawrence E. and Loparo, Kenneth A. 1989. "Artificial Intelligence Simulation and Modelling: A Critical Survey", Artificial Intelligence, Simulation and Modelling, Widman, Lawrence E.; Loparo, Kenneth A. and Nielsen, Norman R. editors, John Wiley and Sons, pages 1-44.

Appendix A Detail Description of the Electrical Power System Domain Model

A.1 Overview of EPS Hardware Design and Development	132
A.2 Description of the Electrical Power System Model	133
A.2.1 Top Level Code Descriptions	133
A.2.2 EPS Component Modelling	134
A.2.3 Simulation Update Cycle	137
A.3 Description of the Component Models	139
A.4 Sensor Data Expected to be Available in the Domain	154
A.5 Sensor Points Available on the EPS Model	155

A.1 Overview of EPS Hardware Design and Development

The spacecraft Electrical Power System (EPS) that is modeled for this work is based on a preliminary design of an experimental breadboard system presently under development at CAL Corporation under contract to the Canadian Department of National Defense (DND). As the specific details of this work are classified by the DND, electrical parameters for the components have been altered. The general architecture is however representative of an electrical power system for spacecraft of this class.

At the time the model was initially encoded, the circuit architecture followed the structure presented here. Since that time, several design changes have been made by CAL Corporation to the EPS regarding the circuit design and component operational characteristics. While changes of this type are typical for any design/development program, it was felt that the original design still adequately represented the general case phased array radar payload spacecraft EPS domain for the purposes of demonstrating the diagnostic system presented in this thesis. Thus at this time, the model does not have a direct parallel in a spacecraft EPS.

The differences between the EPS model used to demonstrate the concepts and techniques of this thesis and an eventual flight system EPS will not affect the overall architecture of the diagnostic system. In the event that the diagnostic system developed here is to be demonstrated on the breadboard environment, the knowledge bases and several of the data structures will have to be revised. As the overall software architecture is not expected to change, it is not expected that this will cause undue hardship.

A.2 Description of the Electrical Power System Model

The Electrical Power System that is modeled in Smalltalk as a simulation that contains some 1548 components as seen in the circuit diagram, figure 2-1. While there is a multiplicity of components, there are only 9 unique component classes in the entire circuit. The circuit is constructed of repeatedly duplicating these 9 components to construct the resulting EPS. The following is a brief description of this simulation, how it is constructed and the merits of the object oriented approach.

A.2.1 Top Level Code Descriptions

The various simulation component definitions are all subclasses of the Smalltalk class called EpsComponent. Within the EpsComponent class several methods and instance variables common to most components are defined (voltageIn, voltageOut, etc.). In Smalltalk, all instance variables and methods are inherited to all subclasses. (Unused methods and instance variables however, do not result in unnecessary overhead.) Inheriting several of the common instance variables and instance variable access methods results in a considerable reduction in programming for the lower level component definitions.

The simulation is constructed and controlled by an instance of the Smalltalk class 'EpsSimulation'. Following instantiation, initialization of the EpsSimulation instance results in the instantiation and initialization of the Primary Power Conditioner (ppcu), Energy Storage Unit (esu) and powerDist (a pseudo component) components. These three components in turn instantiate and initialize the next lower level of component. This

A.2 Description of the Electrical Power System Model

The Electrical Power System that is modeled in Smalltalk as a simulation that contains some 1548 components as seen in the circuit diagram, figure 2-1. While there is a multiplicity of components, there are only 9 unique component classes in the entire circuit. The circuit is constructed of repeatedly duplicating these 9 components to construct the resulting EPS. The following is a brief description of this simulation, how it is constructed and the merits of the object oriented approach.

A.2.1 Top Level Code Descriptions

The various simulation component definitions are all subclasses of the Smalltalk class called EpsComponent. Within the EpsComponent class several methods and instance variables common to most components are defined (voltageIn, voltageOut, etc.). In Smalltalk, all instance variables and methods are inherited to all subclasses. (Unused methods and instance variables however, do not result in unnecessary overhead.) Inheriting several of the common instance variables and instance variable access methods results in a considerable reduction in programming for the lower level component definitions.

The simulation is constructed and controlled by an instance of the Smalltalk class 'EpsSimulation'. Following instantiation, initialization of the EpsSimulation instance results in the instantiation and initialization of the Primary Power Conditioner (ppcu), Energy Storage Unit (esu) and powerDist (a pseudo component) components. These three components in turn instantiate and initialize the next lower level of component. This

process repeats until all components are instantiated and initialized.

The simulation can be manipulated in several ways. Normal operation is performed through the action of methods that control the relevant component's parameters. This includes: turning loads on and off, setting the solar panels solar state (sunlit or eclipse), setting the batteries charge level, etc. The individual component's models are not altered to perform these actions. To model certain faults will however require changing the individual component's model. The component models and their alteration to model faults is described in the next section.

A user interface is provided to the simulation in the form of a graphics window displaying the higher level components, popup windows displaying the cell back and tile level components and a text pane to list the simulation data. These are briefly described in section 5.0 shown in figures 5-3 through 5-6. The software description and source code document [Adamovits 1992] provides additional detail.

A.2.2 EPS Component Modelling

The electrical properties of the domain components are model as Smalltalk code blocks in the corresponding simulation components. In this section, the various types of models encoded in these code blocks are briefly described. The interaction of the combined components that comprise the simulation is described in the following section. Section A.3 provides a description of the individual EPS component models.

The software components of the simulation have one or more 'models' that describe the electrical properties and behaviour of the corresponding hardware element in the domain. Each component has one model called *componentModel* that describes the operational electrical characteristics of the component. The power distribution and load components (intermediate conditioners, transmit receive conditioners, auxiliary conditioners, and radar loads) additionally have a model called *demandModel*. The power source elements have a model called *availableModel* (solar panel, Ppcu).

These model definitions exist at the instance level of each software component. Each is thus fully independent of all others of the same class. Initially, the models are copies of the corresponding *DEFAULT* class definition of the model (*ComponentModel*, *AvailableModel* or *DemandModel*). As the reasoning proceeds, it may be required to modify the operational characteristics of a specific component. This is easily accomplished by reassigning the relevant instance variable (*componentModel*, *availableModel* or *demandModel*)⁷⁰ to the new model definition.

Individual component models are contained in Smalltalk code blocks. As noted previously, each component has an instance variable *componentModel* that contains the code block used by the instance to model the specific component's behaviour. As the behaviour of the component is located in an instance variable rather than in the methods of the component, the behaviour can be modified quite easily by replacing the code block. A simple code block for a *radarLoad* that is stuck in the off position is defined by the block:

⁷⁰ Note: Default class definitions are located in the class variable. These names start with a capital. The instance variables have the same name however these start with a lower case character.

```
[aLoad |
  aLoad power: 0.01.
  aLoad currentIn: 0.001].
```

What the code block does is quite evident. The power and current into the component is independent of the state (on or off) of the component and is some nominal value. For this component the default, normal operational model is:

```
[aLoad |
  efficiency := aLoad efficiency.
  voltage := aLoad voltageIn.
  rfPower := 0.0.          "default level: off"

  (aLoad state = 1.0)
    ifTrue: [power := 5.0]    "turned on power"
    ifFalse: [power := 0.01]. "stand by power"
    "fault mode modeling..."
  (voltage < 10.0)          "low voltageIn go to"
    ifTrue: [power := 0.01]. "stand by power level"
  (voltage = 0.0)
    ifTrue: [power := 0.0].  "dead source..."
    "power in this case is the nominal power..."
  rfPower := power * (aLoad rfEfficiency). "0 < rfEff <= 0.20"
  "power/eff results in actual electrical power"
  power := power / efficiency.    "0 < efficiency <= 1.0"
  (power = 0.0)
    ifTrue: [current := 0.0]
    ifFalse: [current := power / voltage]. "includes efficiency"

  aLoad currentIn: current;
  power: power;
  rfPower: rfPower.
].
```

To change the model of a component (say for example load4) from the existing model to a new code block is accomplished by the following typical statement:

```
load4 componentModel: [ ... a code block ... ]
```

As noted previously, all EpsComponent subclasses have default component models that are used at the time of initialization of the instance. These default models are located in the class's class variable called *ComponentModel*. To return to the default component model located in the class definition (again for a radarLoad instance called load4) the following statement is executed:

load4 componentModel: (RadarLoad componentModel).

Replacing the code blocks to change the components behaviour is applicable in all cases where code blocks are used. That includes (where relevant) the instance variables availableModel, demandModel and componentModel. Finally, it is important to note that changes to this component's model have no effect on the remaining components that are instances of the same class (unless the class definition is changed and the instances are reinitialized to the new class definition). From an AI perspective, the class definitions may be considered to be *default models* which will be revised and changed as additional more accurate data becomes available.

A.2.3 Simulation Update Cycle

The components and data in the simulation are controlled by an instance of the EpsSimulation class. This software element acts as an interface to all external commands and demand for data from the components of the simulation. Various interface methods are provided for this purpose.

The simulation performs the update cycle by initially finding the 'power available' (using the model availableModel) from the solar panel, finding the 'power demanded' (using demandModel) by the loads through the power distribution elements, and comparing the difference of these with the capacity available at the battery. When a power balance has been attained, the bus voltage is set (by the battery characteristics).

Once the bus voltage has been set the components attached to the bus are updated. The attached components (ppcu and intermediate conditioners) will apply their models (located in componentModel) to the to the voltage that now exists. This process continues through to lower level components (trConditioners, loads, solarPanel). At each step along the way, the individual component's model 'componentModel' is applied to set it's operational parameters.

In a fault free case, with an adequately charged battery, all components will operate when turned on. In the event of low voltage on the bus (or intermediate power distribution busses), certain elements may switch off or go to a power down mode. The low voltage will be due to a discharged battery or one or more component failures.

If components are unexpectedly set to an off state, the power balance performed initially will be incorrect. This will require the update cycle to be repeated a second time to resolve the discrepancy. For example, in terms of the powerDistribution pseudo component, the expected power required (as a result of applying the demandModel) does not match the power required (resulting from applying the componentModel). A new bus voltage is then set as a result of this new power level. Reapplying the powerDistribution componentModel will resolve the error.

A.3 Description of the Component Models

A high level description of the component models is provided in this section. For a more complete description of the models see the software description and the commented source code see the companion document to this work [Adamovits 1992].

The component models model both the normal operational state and selected failure modes. These failure modes are used to generate data by the Domain for analysis by the MFDS and as part of the MFDS to adapt the Model to the Domain. The operational modes of the power distribution components is shown in Table A-1, for the energy storage components in table A-2 and for the solar panel and ppcu in table A-3.

	intermediate conditioner	transmit receive conditioner	auxiliary conditioner	radar load
normal operation	x	x	x	x
open circuit	x	x	x	x
tripped circuit breaker	x	x	x	NA
loss of efficiency (electrical)	x	x	x	x
loss of efficiency (RF)	NA	NA	NA	x
always on	NA	NA	NA	x
always off	NA	NA	NA	x

NA - Not Applicable

Table A-1 Operational States of Power Distribution Components

	energy storage unit	cell pack	cell
normal operation (charge/discharge)	x	x	x
short	NA	NA	x
open	NA	NA	x
loss of charge	NA	NA	x

Table A-2 Operational States of Energy Storage Unit Components

	solar panel	primary power conditioner
normal operation	x	x
faults	NA	NA

Table A-3 Operational States for the Solar Panel and Primary Power Conditioner

Auxiliary Conditioner - The component model of the Auxiliary Conditioner is that of a constant power load of 10 watts. This power requirement exists whenever the auxCond has sufficient input power to operate. The behaviour of the AuxCond changes from this model when the efficiency is changed or the breaker is tripped. The sensor values available from the component model are noted figure A-1 and in the table A-4.

state	parameter	value
breaker tripped	voltageIn	set by IntCond
	power	0.0
	currentIn	0.0
breaker ok	voltageIn	set by IntCond
	power	if voltageIn > 24 power = 10.0 else power = 0.0
	currentIn	power / (voltageIn * efficiency)

Table A-4 Auxiliary Conditioner State/Parameter Relationships

Figure A-1 Auxiliary Conditioner Characteristic Curve

Energy Storage Unit - The ESU represents the total storage capacity of the string. Other storage devices (capacitors, thermal, chemical etc.) are not used. The ESU is comprised of 5 Cell Packs each of which contains 20 cells. The cells are assumed to be nickel cadmium and are connected in series. The ESU is connected directly to the primary bus and thus sets the voltage for the primary bus. The ESU voltage will vary from 96 volts to 160 volts. As noted, the primary bus will track this

voltage. There is 1 ESU per EPS string.

The ESU models (ESU, CellPack and Cell) are the most complex of the models used in the EPS. In spite of this, the accuracy of the models may prove inadequate as batteries are highly nonlinear devices.⁷¹ The model presented here includes a energy storage device with I^2R losses and a storage efficiency of 75%.

state	parameter	value
charge	voltage	sum of the cell pack voltages
	power	power available from the primary bus
	current	power / voltage
	time	charge time specified
	charge	sum of cell pack charge levels
discharge	voltage	sum of cell pack voltages
	power	power available to the bus
	current	-(power / voltage)
	time	discharge time specified
	charge	sum of cell pack charge levels

Table A-4 ESU State/Parameter Relationships

Note: The cell models account for the charge efficiency and I^2R losses.

The voltage-charge characteristics of the ESU are displayed in figure A-2.

⁷¹ Nickel cadmium cells have a *memory* of past charge and discharge rates that affects the ability to fully charge and fully discharge the cells respectively. The memory affect builds over several charge and discharge cycles and may be partially reduced through *reconditioning*. Reconditioning involves fully discharging the cell over an extended period of time then recharging the cell under closely controlled conditions (charge rate, cell temperature, initial and final charge, etc).

Figure A-2 Energy Storage Unit, Cell Pack and Cell Characteristic Curves

Cell - Two models are defined in this method, the ComponentModel and ShortModel model for the normal operational state and shorted cell states respectively. The componentModel of the battery is that of a linear device having a linear charge / voltage relationship, I squared R losses and a charge efficiency of 75%. This is based on table A-5 and illustrated in figure A-2. The charge model is extended to create the charge discharge model defined in table A-6.

charge level	available power (charge) Watt-hr ⁷²	voltage
fully charged	40.0	1.60 volts
discharged	0.0	0.96 volts

Table A-5 Cell Parameter Relationships

⁷² Charge is specified in Watt-Hours rather than the accepted standard Amp-Hours. This change was made as a modelling convenience only and does not represent a particular manufactures specification.

state	parameter	value
charge	current	specified by the ESU
	charge	$\text{charge} + (\text{power} * \text{efficiency} * \text{time} / 3000)$
	voltage	$0.96 + ((0.64 * \text{charge})/40.0) + (\text{current} * \text{resistance}) **$
	time	charge time specified by ESU
	power	voltage * current
discharge	current	specified by the ESU
	charge	$\text{charge} - (\text{power} * \text{efficiency} * \text{time} / 3000)$
	voltage	$0.96 + ((0.64 * \text{charge})/40.0) - (\text{current} * \text{resistance}) **$
	time	discharge time specified
	power	voltage * current

Notes: ** resistance is the effective resistance for I^2R calculations. (-I and -P) indicates discharge.

Table A-6 Cell State/Parameter Relationships

Note that the shorted model for the cell includes I^2R losses only. The cell is not capable of holding a charge. This is displayed in table A-7.

state	parameter	value
charge	current	specified by the ESU
	charge	0.0
	voltage	(current * resistance) **
	time	has no effect
	power	$-(I^2R)$
discharge	current	specified by the ESU
	charge	0.0
	voltage	- (current * resistance) **
	time	has no effect
	power	$-(I^2R)$

Notes: ** resistance is the effective resistance for I^2R calculations. (-I and -P) indicates discharge.

Table A-7 Cell State/Parameter Relationships: Shorted Cell Model

CellPack this is the model of the cellPack. It performs a summation of the individual cell instance variables and updates the instance variables in the calling instance directly. The characteristics of the cell pack are noted in table A-8 and illustrated in figure A-2.

parameter	value
voltageIn	sum of cell voltages
current	as specified by the ESU
state	as specified by ESU
time	as specified by ESU

Table A-8 Cell Pack Parameters

Intermediate Bus - The intermediate bus connects the intCond with its trConds and auxCond. This component exists for descriptive purposes only. This component is not modeled and the equivalent action is performed by the IntCond class.

Intermediate Conditioner - The intermediate conditioner regulates the primary bus voltage from the 96-160 volt range to a nominal 28 volt intermediate bus with a nominal efficiency of 80%. The IntCond output voltage will vary inversely with the power drawn by the attached loads. Each IntCond is connected to the primary bus on the input and to two transmit/receive conditioner and one auxiliary conditioner at the output. There are 120 Intermediate Conditioners on each string of the EPS.

Figure A-3 Intermediate Conditioner Characteristic Curves

The default model for the IntCond as a piece wise linear function as noted in table A-9 and illustrated in figure A-3. Note that if the power demanded exceeds 100 watts the trip breaker will trip and the breaker reset is a manual operation.

state	parameter	value
breaker tripped	voltageIn	set by ESU
	all others	0.0
breaker ok	voltageIn	set by ESU
	currentIn	power / voltageIn
	power	voltageOut * currentOut / efficiency
	voltageOut	$24.0 + (100.0 - \text{powerDemand}) * 4.0 / 100$
	currentOut	powerDemand / voltageOut

Table A-9 Intermediate Conditioner State/Parameter Relationships

Note: powerDemand is the sum of the demanded power by the loads. The characteristic curves of the Intermediate Conditioner are displayed in figure A-3.

PowerDist - Defines the default model for the Power Distribution pseudo component.

This model performs a summation of the power requirements of the down stream components as a simplification measure in the simulation. This (pseudo) component has no electrical properties as such. However, the instance variables for voltageIn, currentIn and power are representative of the summation of the Intermediate Conditioner and downstream component requirements.

parameter	value
voltageIn	as set by ESU
currentIn	summation of all intCond currentIn values
power	voltageIn * currentIn

Table A-10 Power Distribution Parameters

Primary Power Conditioning Unit - Each PPCU contains several voltage regulators in parallel a few of which are active at any time. In the event of voltage regulator failure, the transition from an active to backup voltage regulator is performed by hard wired logic within the Ppcu. At the time of writing, control of this transition is not available. The PPCU output is connected to the primary bus. There is one PPCU per EPS string.

The power needed to power the loads and charge the ESU is provided by the solar panel and ppcu. If the loads do not require the output available from the ppcu it will be available to charge the ESU. If the ESU also does not require the power then the excess will be dumped through heaters and thermal radiators (these are not modeled).

parameter	value
power	if (power demand > power available) (solar panel power) * (PPCU efficiency) else (power demand)
voltageOut	as set by ESU
voltageIn	as set by solar panel
currentOut	power / voltageOut
currentIn	as set by solar panel

Table A-11 Primary Power Conditioning Unit Parameters

The characteristic curves of the Primary Power Conditioning Unit are displayed in figure A-4.

Figure A-4 Primary Power Conditioning Unit Characteristic Curves

Primary bus - The primary bus connects the Ppcu output, the ESU and the PowerDist. In terms of the software simulation there is no primary bus class. The equivalent action of the primary bus is performed in the EpsSimulation class.

Radar Loads - The RadarLoads and the related processing elements are the primary payload of the spacecraft. The radar loads have a time averaged electrical load of

5 watts when in transmit and receive mode. At which time they emit 1 watt of RF power (for an RF conversion efficiency of 20%). The Radar Loads have a minimal power requirement while in standby mode (approx 0.001) watts. There are 4 RadarLoads per TRCond. This totals 960 RadarLoads per EPS string. The model below, notes the electrical and RF properties of the radar load:

state	parameter	value
transmit/ receive	power	5.0
	rfPower	1.0
	voltageIn	set by trCond
	currentIn	$\text{power} / (\text{voltageIn} * \text{efficiency})$
standby	power	0.01
	rfPower	0.0
	voltageIn	set by trCond
	currentIn	$\text{power} / (\text{voltageIn} * \text{efficiency})$

Table A-12 Radar Load State/Parameter Relationships

The characteristic curves of the Radar Load are displayed in figure A-5.

Figure A-5 Radar Load Characteristic Curve

Solar Panel - The solar panel is a series/parallel connection of photovoltaic cells. This device is not instrumented with the exception of total output voltage and current and temperature sensors. The solar panel can not be controlled or configured however, it has a certain degree of hardware based fault recovery in the form of sub-string fuses. There is one 4 Kw solar panel per EPS string (for the purposes of this system).

The model of the SolarPanel is that of a power source with a voltage - current characteristic curve defined by a piece wise linear equation based on the bounding values in the following table.

output voltage and current

170 volts at 0 amps

160 volts at 25 amps

0 volts at 30 amps

(based on a 4 kw solar panel)

This results in the following defining equations for the solar panel:⁷³

$$\text{voltageOut} = 170.0 - [(10.0/25.0) * \text{currentOut}].$$

$$\text{for } 0 < \text{currentOut} \leq 25.0$$

$$\text{voltageOut} = 160.0 - [(160.0/5.0) * (\text{currentOut} - 25.0)].$$

$$\text{for } 25.0 < \text{currentOut} < 30.0$$

This model does not model the knee of the voltage current curve accurately. As the actual PPCU will be optimized to operate at the peak power point (i.e. the knee of the curve) this model is insufficiently accurate for a final system. The characteristic curves of the Solar Panel are displayed in figure A-6.

Figure A-6 Solar Panel Characteristic Curves

Transmit Receive Bus - The transmit receive bus connects the trCond with its loads. The transmit receive bus exists for descriptive purposes only. This component is not modeled and the equivalent action is performed by the trCond model.

⁷³ These equations can be considered to be first order approximations only. The actual behaviour of the solar panel can be considered to be linear for most of the voltage limited ($I < 25$) and current limited ($I > 25$) regions of operation. The transition from voltage limited to current limited operation however is not as abrupt as is the case in this model.

Transmit Receive Conditioners - The TRCond regulate the voltage to the transmit receive elements to a high degree of precision. The nominal output of the TRCond is 12 volts with a maximum ripple of +/- 0.01 volts with a nominal efficiency of 95%. The input of the TRCond is the Intermediate conditioner output. The output of the TRCond is to the transmit receive elements. There are 4 transmit receive elements per TRCond and 2 TRCond per intCond resulting in 240 TRCond per EPS string.

The model for the transmit receive conditioner TRCond is as a piece wise linear function as noted by the following table:

If Power Demanded > 40 watts then trip breaker

(Note: Breaker reset is a Manual operation)

state	parameter	value
breaker tripped	voltageIn	set by IntCond
	power	0.0
	currentIn	0.0
breaker ok	voltageIn	set by IntCond
	voltageOut	if (voltageIn > 24) voltageOut = 12 else voltageOut = 0.0
	currentOut	if (voltageOut > 0.0) power demand / voltageOut else 0.0
	power	if voltageIn > 24 power = 10.0 else power = 0.0
	currentIn	power / voltageIn

Table A-13 Auxiliary Conditioner State/Parameter Relationships

The characteristic curves of the Transmit Receive Conditioner are displayed in figure A-7.

Figure A-7 Transmit Receive Conditioner Characteristic Curve

A.4 Sensor Data Expected to be Available in the Domain

In the hardware system presently under development a limited number of sensor points are available on the telemetry link. The following is a list of the sensor data that is expected to be available for the components that comprise the EPS:

Solar Panel - Output voltage and output current (measured at the bus connecting the solar panel and the PPCU), several surface temperature readings.

PPCU - output voltage, output current, status (ok, failure) (All measured internally to the PPCU).

ESU - current and voltage (measured internally), status (charging, discharging)

CellPack - Current, voltage, temperature (all measured internally), status

Cell - Voltage, current (measured at the cell)

IntCond - Voltage and current output (measured internally), status (breaker ok, tripped)

AuxCond - Voltage and current output (measured internally), status (breaker ok, tripped)

TRCond - Voltage and current output (measured internally), status (breaker ok, tripped)

**RadarLoads - none (RF power will be estimated from radar signal power output;
Commanded state will be obtained from the command stream. The commanded
state is not a sensor but represents the expected state.)**

A.5 Sensor Points Available on the EPS Model

Considerably more sensor data is available on the model than from the telemetry stream. All data available from the model is noted in the following. Data that is only available on the telemetry stream is noted by (tm).

Solar Panel

Available data:

voltageOut - the output voltage (tm),

currentOut - the output current (tm),

nominalPower - solarPanel power output when sunlit,

power - actual power output,

state - status: 0 - eclipse, 1 sunlit.

Missing data:

temperature - temperature sensors are not model.

PPCU

Available data:

voltageOut - the output voltage (primary bus voltage) (tm),

voltageIn - the input voltage (solarPanel voltageOut) (tm),

currentOut - the output current (tm),

currentIn - the input current (solarPanel output),

power - actual power output to the bus,

efficiency - efficiency of the device (power out/power in)

Missing data:

temperature - temperature sensors are not model.

status - status of the power conditioners

ESU

Available data:

currentIn - (+) indicates current into the ESU, (-) indicated current out (tm),

voltageIn - sum of the cellPack voltages (tm),

power - (+) indicates power into the ESU, (-) indicated power out,

status - charging/discharging (derived from sign of currentIn & power),

charge - charge/discharge (derived from individual cell charge values)

Missing data:

temperature - temperature sensors are not modeled

CellPack

Available data:

currentIn - (+) indicates current into the cellPack, (-) indicated current out (tm),
voltageIn - sum of the cell voltages (tm),
power - (+) indicates power into the cellPack, (-) indicated power out,
status - charging/discharging (derived from sign of currentIn & power)
charge - charge/discharge (derived from individual cell charge values)

Missing data:

temperature - temperature sensors are not modeled

Cell

Available data:

currentIn - (+) indicates current into the cell, (-) indicated current out (tm),
voltageIn - cell voltage (tm),
power - (+) indicates power into the cell, (-) indicated power out,
status - charging/discharging (derived from sign of currentIn & power)
charge - charge on the individual cell

Missing data:

temperature - temperature sensors are not modeled

IntCond

Available data:

voltageOut - the output voltage (intCond, auxCond voltageIn) (tm),
voltageIn - the input voltage (primary bus voltage),
currentOut - the output current (tm),

currentIn - the input current,
power - actual power drawn from the primary bus,
efficiency - efficiency of the device (power out/power in)
state - circuit breaker status (tm)

Missing data:

temperature - temperature sensors are not model.

AuxCond

Available data:

voltageIn - the input voltage (IntCond voltageOut),
currentIn - the input current,
power - actual power drawn from the intermediate conditioner (tm),
state - circuit breaker status (tm)

Missing data:

temperature - temperature sensors are not modeled,
voltageOut - secondary loads not modeled,
currentOut - secondary loads not modeled.

TRCond

Available data:

voltageOut - the output voltage (tm),
voltageIn - the input voltage (IntCond voltageOut),
currentOut - the output current (tm),
currentIn - the input current,
power - actual power drawn from the intermediate bus,
efficiency - efficiency of the device (power out/power in)

state - circuit breaker status (tm)

Missing data:

temperature - temperature sensors are not modeled.

RadarLoads

Available data:

voltageIn - voltage at input (TRCond voltageOut)

currentIn - current at input,

power - average RF power transmitted (tm)

state - commanded state (transmit/receive or stand-by) (tm)

efficiency - electrical conversion efficiency⁷⁴

rfEfficiency - RF conversion efficiency⁷⁵

Missing data:

power - electrical power drawn from the trCond module.

⁷⁴ Note: Small changes in electrical efficiency do not affect RF output. A very low electrical efficiency will cause the radar load to stop functioning however.

⁷⁵ Note: The RF conversion efficiency is deemed to have no impact on the electrical power required for the device. Nominally, the RF efficiency is 20%.

Appendix B Tests Performed on the Multiple Fault Diagnosis System

B.1 Overview of Tests Performed	161
B.2 Electrical Power System Simulation Testing	161
B.3 Multiple Fault Diagnostic System Testing	166
B.4 EPS Simulation Browser and MFDS Browser Testing	173

B.1 Overview of Tests Performed

During the design and development of the MFDS several tests were performed to evaluate the MFDS and its subelements. Appendix B summarizes these tests and presents but does not analyze the resulting data. For a discussion of the tests performed and an analysis of the data see sections 6 and 7 respectively.

The tests performed on the completed MFDS fall into three groups. The following list notes the groups and the sections describing the tests performed:

- i tests of the EPS simulation (used for the model and domain) to generate both normal and failure mode data (Appendix B.2),
- ii tests of the functionality and completeness of the knowledge bases (QualRules, QuanRules, DiagRules etc.) (Appendix B.3),
- iii tests of the user interfaces (Appendix B.4).

B.2 Electrical Power System Simulation Testing

The Electrical Power System simulator used for both the model and the domain was designed to meet the overall requirements of the EPS defined in section 2 and Appendix A. The high level requirements for the EPS defined in section 2 defined the requirements for the components that constitute the EPS. The component requirements are defined in Appendix A.3 and displayed in figures A-1 through A-7.

The EPS component models and the resulting EPS simulation were tested against and found to comply with the requirements as defined in Appendix A.3. In addition to normal

operational requirements, the components comply with the faults as defined in tables A-1, A-2 and A-3 in Appendix A.3. The reader is referred to these tables for information on the functionality modeled for each electrical component of the EPS.

The components and the resulting EPS simulation were tested extensively during the software development stage. Tests were performed on individual components, sub-assemblies and the resulting EPS. These preliminary tests were completed largely through the use of the debugger and are not documented here.

Following the development of the *EPS Browser* user interface, the EPS simulation and the EPS components were again tested. These later tests identified several errors in the implementation which were corrected. A sample component test and evaluation is presented in the following:

The requirement on the transmit receive conditioner (TRCond) is that it provide a regulated 12 volt output regardless of the load on/off state. Changes in the combine load power result in changes in the conditioner current output only. Voltage input to the transmit receive conditioner can change slightly as the demands on the intermediate conditioner change. This is not to affect the transmit receive conditioner output voltage.

To evaluate the component model the following tests were performed:

- 1 TRCond voltage and current values for input and output were observed for the attached loads in the on state.
- 2 TRCond voltage and current values for input and output were observed for the

attached loads in the off state.

- 3 The input voltage to the TRCond was changed by varying the load on the intermediate conditioner and the ability of the TRCond to regulated the output voltage was observed.
- 4 An open circuit fault was injected into the transmit receive conditioner and the voltage and current were observed.
- 5 A tripped breaker fault was injected into the transmit receive conditioner and the voltage and current were observed.
- 6 The faults were removed from the TRCond and tests 1 to 3 above were repeated to verify the component has returned to the operational state.

The data for the tests above was observed on the *EPS Browser Data Window* text output. A sample of data listings from the EPS Browser Window is shown in figure B-1. The figure shows the electrical parameters for the transmit receive conditioner when all the loads are turned on, a sample load,⁷⁶ and the intermediate conditioner to which this transmit receive conditioner is attached.

The remaining tests (tests 2 to 6 above) for the intermediate conditioner were completed and the results were also observed on the EPS Browser Data Window. In all cases, the data was examined to ensure that the correct voltage, current, status and power values were observed. This was accomplished through manual calculations using the model definitions in Appendix A.3. All tests were observed to have satisfactory results.

⁷⁶ The other 3 loads attached to this transmit receive conditioner are omitted for clarity. These other loads were however examined and were observed to have the same electrical parameters to the load listed.

EPS Simulation Data		
TRCond1	IC9	
	voltageIn	25.915789
	currentIn	8.1234768e-1
	voltageOut	12.0
	currentOut	1.6666666
	power	21.052631
	state	1.0
	efficiency	0.95
Load1	TRCond1	IC9
	voltageIn	12.0
	currentIn	4.1666666e-1
	power	5.0
	state	1.0
	efficiency	1.0
	rfEfficiency	0.2
	rfPower	1.0
TRCond1	IC9	
all loads attached to this trCond turned off		
*** Simulation updated ***		
TRCond1	IC9	
	voltageIn	26.755210
	currentIn	1.5736631e-3
	voltageOut	12.0
	currentOut	3.3333333e-3
	power	4.2105263e-2
	state	1.0
	efficiency	0.95 Y
Load1	TRCond1	IC9
	voltageIn	12.0

Figure B-1 Transmit Receive Conditioner Testing

The remaining components of the EPS were tested and evaluated using the same technique as noted above for the intermediate conditioner. All components were demonstrated to have the behaviours as noted in tables A-1, A-2 and A-3 of Appendix A.3. The test of the EPS components was thus complete.

Following the test of the individual component models, the integrated EPS simulation was tested. For these tests, the primary concern was the correct operation of the simulation update cycle, as defined in appendix A.2.3. During simulation update, the power available (from the sources) and demanded (by the power distribution and loads) all component models is examined and then updated (by the EpsSimulation instance). The components were closely examined (manually through the browser and text interface) to ensure that the appropriate voltage and current levels were recorded and that Kirchhoff's Laws were not violated. Tests performed on the simulation included:

- Sunlit and eclipse operation
- Charge and discharge of the batteries
- Changing load levels and observing the behaviour of the various components interacting.

Additionally, several tests were performed to examine the behaviour of the simulation in the presence of component failures. All tests performed on a component basis as identified above (and found in tables A-1 through A-3) were repeated on the complete simulation. Errors that were detected in either the component or simulation behaviour were corrected and retested. It is believed that the simulation is a good representation of the domain.

B.3 Multiple Fault Diagnostic System Testing

Several tests were performed during the development and integration stages of the MFDS components and on the resulting Multiple Fault Diagnostic System itself. Tests performed during the early stages of the system development are not documented nor are they reported on here. Following the integration of the MFDS components forming the MFDS, the functionality of the MFDS was tested as a unit. The functionality of each reasoning system was tested to the extent that the knowledge is defined.⁷⁷ The tests performed are summarized in the table B-1a and B-1b on the following two pages.

The table is interpreted as follows:

Column 1 is a key to the *test case* information that follows the tables describing the actual tests performed.

Column 2 notes the component (in **bold type**) or the functionality demonstrated.

Column 3 (FD QL) notes the qualitative fault detection results.

Column 4 (FD QN) notes the quantitative fault detection results.

Column 5 (diag) notes the diagnosis and hypothesis generation results.

Column 6 (prun) notes the hypothesis space pruning system results.

Column 7 (val) notes the validation system results.

Column 8 (test) notes the test case selection system results.

Column 9 (ver) notes the verification system results.

An x in the table indicates successful completion of the test. No indicator indicates that the test is not relevant to this software element. Items of interest (noted by i, ii etc.) are

⁷⁷ The reasoning systems were not tested for missing knowledge. It is recognized that at this time, the knowledge bases are incomplete. This is an indication of the development time available however, a limitation in the approach.

described on the page immediately following table B-1b. Information describing the individual test cases follows the notes identified above.

In most cases, the tests were performed in one cycle. The breaker reset fault test (for the intermediate conditioner, transmit receive conditioner and auxiliary conditioner) requires two cycles through the reasoning system to complete the analysis. The sequence of events to accomplish this test is as follows:

- Set breaker to tripped and the component to open circuit also.
- run MFDS
- MFDS identifies the breaker as tripped (correctly, given the data available) and verifies this with the model
- The validation system then attempts to validate the diagnosis by performing a breaker reset test in the domain and is unsuccessful
- The MFDS reasoning process then changes its response to breaker reset fault and additionally identifies the component as open circuit based on this new information.

The concluding diagnosis was thus changed from the initial value as more information was made available.

test case	component/test	FD QL	FD QN	diag	prun	val	test case	ver
	auxCond							
1	open circuit	x		x	x	x	x	x
2	breaker tripped	x		x	x	x	x	x
3	breaker reset fault	x		x	x	v		
4	breaker status error		x					ii
5	power error		x					ii
6	efficiency			x		vi		iii
	intCond							
7	open circuit	x		x	x	x	x	x
8	breaker tripped	x		x	x	x	x	x
9	breaker reset fault	x		x	x	v		
10	loss of efficiency: unknown conditioner	x						ii
11	open circuit: unknown conditioner	x						ii
12	current error		x					ii
13	voltage error		x					ii
14	breaker status error		x					ii
15	low output current			x				ii
16	low cond power demand			x				ii
17	high cond power demand			x				ii
18	low output voltage			x				ii

Table B-1a Multiple Fault Diagnostic System Tests

test case	Component/test	FD QL	FD QL	diag	prun	val	test case	ver
	Radar Load							
19	turn on fault	x						iv
20	turn off fault	x						iv
21	loss of efficiency	x						iii
22	power error		x					ii, iii
23	command status error		x					iii
24	loss of RF efficiency		x		x	vi		iii
25	loss of efficiency: electrical				x	vi		iii
26	open circuit	x		x	x	x	x	x
	trCond							
27	open circuit	x		x	x	x	x	x
28	breaker tripped	x		x	x	x	x	x
29	breaker reset fault	x		x		v		
30	voltage regulation fault	x		x	x			i
31	loss of efficiency: unknown load	x						ii
32	current error		x					ii
33	voltage error		x					ii
34	breaker status error		x					ii
35	high load power demand			x				ii
36	low load power demand			x				ii

Table B-1b Multiple Fault Diagnostic System Tests (cont'd)

Notes on Test Results

- x Successfully demonstrated behaviour.
- ' ' (blank). Test is not relevant to this system or is tested by other means.
- i This test case does not presently exist. Fault can not be corrected. Definitive data supporting this fault hypothesis is not available.
- ii Covered by other tests for this or related components.
- iii Can not be resolved using available data. See discussion in section 7.
- iv Not presently tested, should be possible with additions to diagnosis and pruning system's rule bases.
- v Validation failed (correctly) as more information became available, diagnosis changed to open circuit (correctly) see test case 3.
- vi Software to support the calculation the anticipated efficiency from the observed data has not been written. This is not a failing of the reasoning system but the ability to modify the model appropriately to reflect changed behaviour.

Description of the Test Cases

Auxiliary Conditioner (Tests 1-6)

- 1 Open circuit: Auxiliary conditioner at intermediate conditioner 2 set to open circuit.
- 2 Breaker tripped: Auxiliary conditioner at intermediate conditioner 6 set with breaker in tripped state.
- 3 Breaker reset fault: Auxiliary conditioner at intermediate conditioner 10 set to have both open circuit and breaker tripped faults. MFDS run. Noted breaker tripped. Verification confirmed breaker tripped as a plausible fault. Test case selected. Verification of tripped breaker failed to remove breaker tripped fault. Component now diagnosed as breaker reset fault and open circuit.
- 4 Breaker status error: Observed in tests 2 and first part of test 3.
- 5 Power error: Observed in tests: 1, 2, 3 and 6.
- 6 Efficiency: Auxiliary conditioner at intermediate conditioner 14 set to have an efficiency of 0.5 (normal efficiency is 1.0). Note: Validation system does not yet permit modifications to the model to validate this fault.

Intermediate Conditioner (Tests 7-18)

- 7 Open circuit: Intermediate conditioner 2 set to open circuit.
- 8 Breaker tripped: Intermediate conditioner 6 set with breaker in tripped state.
- 9 Breaker reset fault: Apply sequence of test 3 to intermediate conditioner 10 and observe the results.

- 10 Loss of efficiency for unknown conditioner: The transmit receive conditioner 1 attached to intermediate conditioner 14 is set to have an efficiency of 0.4 (normal efficiency is 0.95). The rules associated with the intermediate conditioner noticed this change.
- 11 Open circuit for unknown conditioner: The transmit receive conditioner 1 attached to intermediate conditioner 18 is set to be open circuit. The rules associated with the intermediate conditioner noticed this change.
- 12 Current error: Observed in tests: 7, 8, 9, 10, 11 and 17.
- 13 Voltage error: Observed in tests: 7, 8, 9, 10, 11, 17 and 18.
- 14 Breaker status error: Observed in test 8 and the first part of 9.
- 15 Low output current: Observed in test: 7, 8 and 9.
- 16 Low conditioner power demand (unknown conditioner): Observed in tests: 7, 8, 9 and 11.
- 17 High conditioner power demand (unknown conditioner): The three conditioners attached to intermediate conditioner 22 are set to have an efficiency of 0.2. Also observed in test 10 results.
- 18 Low output voltage: Should be observed from test 17 however an error in the intermediate conditioner model when under excessively high loads prevents this from occurring.

Radar Load (Tests 19-26)

- 19 Turn on fault: All loads attached to intermediate conditioner 2, transmit receive conditioner 1 were turned off. The domain simulation was updated. Load 1 was set to have an always off fault and then turned on. The simulation was updated and the MFDS was run observing the results.
- 20 Turn off fault: All loads attached to intermediate conditioner 6, transmit receive conditioner 1 were turned on. The domain simulation was updated. Load 3 was set to have an always on fault and then turned off. The simulation was updated and the MFDS was run observing the results.
- 21 Loss of efficiency (electrical): Load 4 attached to transmit receive conditioner 1 which is in turn attached to intermediate conditioner 10 was set to have an electrical efficiency of 0.3.
- 22 Power error: Observed in tests: 19, 20, 24 and 26.
- 23 Command status error: Observed in tests: 19 and 20.
- 24 Loss of RF efficiency: Load 1 attached to transmit receive conditioner 2 which is in turn attached to intermediate conditioner 14 was set to have a RF efficiency of 0.05 (normally 0.2).
- 25 Loss of efficiency: See test 21.
- 26 Open circuit: Intermediate conditioner 18, transmit receive conditioner 2 load number 3 was set to open circuit.

Transmit Receive Conditioner (Tests 27-36)

- 27 Open circuit: Transmit receive conditioner 1 attached to intermediate conditioner 2 was set to open circuit.
- 28 Breaker tripped: The breaker for transmit receive conditioner 2 attached to intermediate conditioner 6 was set to tripped.
- 29 Breaker reset fault: Apply sequence of test 3 to intermediate conditioner 10, transmit receive conditioner 1 and observe the results.
- 30 Voltage regulation fault: The loads attached to transmit receive conditioner 1 for intermediate conditioner 18 were each set to have an electrical efficiency of 0.05 (normally 0.2). This caused the load on the transmit receive conditioner to be increased excessively causing a loss of regulation capability.
- 31 Loss of load efficiency (unknown load): Load 2 attached to transmit receive conditioner 1 of intermediate conditioner 18 was set to have an electrical efficiency of 0.05 (normally 0.2).
- 32 Current error: Observed in tests: 27, 28, 29, 30 and 31.
- 33 Voltage error: Observed in tests: 27, 28, 29 and 30.
- 34 Breaker status error: Observed in tests: 28 and 29.
- 35 High load power demand: Observed in tests: 30 and 31.
- 36 Low load power demand: Observed in tests: 27 and 29.

Further notes on Testing the MFDS

The object oriented (Smalltalk) based modules of the MFDS (browsers, interface code, procedural processing, etc.) were tested extensively prior to integration into the MFDS and separately from the reasoning elements. Following integration, the MFDS was tested primarily on a major software component basis (fault detection system, diagnosis and hypothesis generation system, etc.). Although specific tests were not performed to ensure correct operation of Smalltalk portions following integration, the functionality was tested through the tests performed on the associated reasoning systems.

B.4 EPS Simulation Browser and MFDS Browser Testing

Formal requirements for the user interface were not specified for this system. As this is the case, a user interface was designed to suit the *design and development issues* of the MFDS only. Its design evolved to suit the needs of the MFDS development. Considerable improvement is possible in the form of a better textual output system and additional graphical support.

Tests were performed to evaluate the functionality of the two primary browsers and the auxiliary text and popup windows used in this system. Several tests were performed during the development and implementation of the software system. These initial tests were primarily conducted to ensure that the facilities for mouse selected graphical and menu selection input and textual output were operational. These initial tests are not documented.

The capability of the user interfaces was extended and evolved to suit the needs of the evolving reasoning system. Formal tests were not conducted on the user interface. The successful operation of the user interface continued to be evaluated as part of other tests being conducted (as noted in previous sections).

The functionality offered by the user interface is noted briefly in section 5.0 and in greater detail in [Adamovits 1992]. Figures displaying the user interface graphical and textual windows are:

Figure 5-2 Multiple Fault Diagnosis System Browser

Figure 5-3 EPS Domain Model Browser

Figure 5-4 EPS Domain Model Browser Text Window

Figure 5-5 Intermediate Conditioner Popup Window

Figure 5-6 Cell Pack Popup Window

END

24 05 93

FIN