An Integrated Diagnostic Architecture for Autonomous Robots

Kelvin Hamilton

Thesis submitted
for the
Degree of Doctor of Philosophy

Heriot-Watt University
Department of Computing and Electrical Engineering

May 18, 2002

Abstract

Autonomous robotic vehicles suffer from a lack of autonomy in that they are often unable to cope with the unforeseen circumstances common in the real-world. A key part of autonomy is being able to firstly recognise that something has gone wrong, then to be able to determine what the problem is, then to be able to cope with that problem. This research addresses these problems by proposing that autonomous robotic mission controllers should work in conjunction with a diagnostic system.

In order to improve diagnostic performance it is common to use more than one type of diagnostic system. This research proposes and evaluates an underlying architecture, RECOVERY, for integrating heterogeneous diagnostic systems and their associated diagnostic information. It also provides novel ways of using previously discarded design information such as dynamic models and circuit diagrams. Novel methods of diagnostic search space reduction and domain-independent diagnostics are also presented, together with supporting work on the types of information available on an autonomous robotic vehicle.

The RECOVERY architecture is implemented and evaluated using an Autonomous Underwater Vehicle (AUV), designed and constructed during the first year of this research. The AUV is called RAUVER. The experimental results show that RECOVERY successfully integrates the diagnostic tools and information.

This thesis is dedicated to my friends and family.

Kelvin Hamilton

Edinburgh, Scotland, UK

Spring 2002

Acknowledgements

Thanks go first to the United Kingdom Engineering and Physical Sciences Research Council (EPSRC) for funding this research with Standard Research Quota Nomination 98315473.

Many thanks to Professor David Lane for providing me with an incredible 'once in a lifetime' opportunity to work on my independent research. Dr. Nick Taylor helped me greatly with his kind support, particularly when I was at my lowest point and seriously thinking of giving up. Dr Keith Brown was kind enough to help me out with invaluable ideas and appraisals.

Len McLean deserves a special mention for all his support, hard work, patience and insight. Without him the RAUVER vehicle would still exist only as a pile of components. He also helped during the frequent RAUVER deployments during which this research was evaluated, often involving heavy lifting and clearing away goat droppings. Ron Lynch, Ian Chalmers and Dave Haldane provided invaluable help and advice in constructing and modifying RAUVER's mechanical components, often at short notice and under pressure.

All the people in the Ocean Systems Laboratory provided feedback, ideas and a great place to work. Particular thanks to Joe for large doses of cynicism and helping with the thesis.

The people at East of Scotland Water Plc provided great assistance in allowing unlimited, free access to their lochs and reservoirs.

The Unites States Office of Naval Research (USONR) funded travel costs for an invaluable visit to the Ocean Engineering Departments of Massachusetts Institute of Technology (MIT), Florida Atlantic University (FAU), the Woods Hole Oceanographic Institute (WHOI), the Autonomous Undersea Systems Institute (AUSI) and the 2000 Symposium on Underwater Vehicles. The Symposium organisers generously allowed me free entry in exchange for some assistance at the conference, work which I enjoyed and allowed me to meet some new and interesting people. The hospitality at all of these places was generous, friendly and helpful in the extreme.

Finally, my friends and family supported me and understood when I had to concentrate on work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Throughout history the idea of artificially constructed entities has fascinated mankind, from the Golems of Prague, through the robots of Fritz Lang's Metropolis, to the androids of Philip K. Dick's Blade Runner. Modern technology has finally enabled the construction of real robotic agents capable of functioning in the real world, although they are a long way behind their fictional forerunners.

Many modern robots are implemented as vehicles such as space-probes, aeroplanes, submarines, wheeled buggies or even humanoid walking robots. These robotic vehicles are increasingly being used to replace humans in harsh environments, with the eventual goal being to provide them with the ability to perform tasks and cope with problems without any human assistance. This ability, known as *autonomy*, is the subject of a large worldwide research effort.

This chapter introduces different types of robotic vehicles and discusses their advantages over humans for exploration purposes. A common weakness crucial to autonomy is then highlighted and the purpose of this research introduced. The chapter ends by detailing the thesis structure.

1.1 Types of Robotic Vehicles

Robotic vehicles are currently used in a wide variety of environments, many of which are hostile to humans. Currently, most robot craft are remotely controlled but there is a growing move towards fully autonomous vehicles. Examples follow.

Unmanned Airborne Vehicles (UAVs) are a rapidly maturing technology, with their main applications being military and police. They are increasingly being used as spotter aircraft for artillery or other weapon delivery systems. Recently a UAV flew autonomously, including takeoff and landing, from North America to Australia

where it is now being used for coastline surveillance.

Although some autonomous surface craft (boats) exist most sea-based robots operate underwater. Remotely Operated Vehicles (ROVs) are used extensively for sub-sea survey and intervention. Autonomous Underwater Vehicles (AUVs) are a rapidly maturing technology, increasingly used commercially to survey pipelines and other seabed features. Scientific surveys of the water column and other areas of interest are also conducted, whilst the military is developing their mine hunting capability.

Autonomous land vehicles are uncommon at present due to the great difficulty involved in navigating terrain, although some advances have been made into structured environments such as factories. Some of the most famous land robots have been used for exploring other planetary bodies in our solar system.

Space has a long history of planetary robotic exploration, dating back to the cold-war technology boom of the mid 1950's. The Americans spent a great deal of time and effort on transporting men to the Moon, where they could stay for only a few hours. The Soviets concentrated instead on robotic exploration and landed a remotely controlled wheeled buggy; *Lunokhod*. This robot stayed on the moon for months, not hours, until its circuitry was destroyed by radiation.

The most advanced land vehicle so far, *Mars Sojourner*, had a limited amount of autonomy in order to cope with the large communications time lag between Earth and Mars. This time lag makes remote operation difficult and so the *Mars Sojourner* could decide whether it was capable of negotiating objects in its path. This proved to be a successful approach.

1.2 Exploration

One of the most suitable applications for autonomous robotic vehicles is exploration, usually involving mapping, sampling and discovering interesting features such as the ancient sunken temples off the coast of Japan and India.

Exploration is an extremely risky operation as it involves, by nature, venturing into the unknown. There is usually very little backup which, coupled with the possibility of loss, leads to it being a very risky investment. In compensation, the

rewards can be very high.

1.2.1 Historical Solutions

Until the middle of the twentieth century all exploration was done in two ways: by going there, or by looking through a telescope. The disadvantages of using humans for exploration mainly concern logistics; humans require food, water, rest, shelter and a relatively narrow set of environmental conditions. Historically, the lives of the explorers were often lost. Recently, political conditions have changed so that the loss of modern explorers, such as the crew of the U.S. Space Shuttle *Challenger*, is no longer acceptable.

The major advantage of human exploration is the ability to cope with unforeseen problems, together with the political and social implications of 'having been there'.

1.2.2 Robotic Exploration

Currently there are two main areas of exploration: space and sub-sea. In these extremely hazardous environments manned exploration is the second wave. Robots push back the frontier and, in some cases, prepare the way for humans.

The greatest disparity between the achievements of human and robotic explorers is in space. The furthest man has ventured is the Moon but a robotic space probe, *Voyager*, has passed Pluto and is heading through the Oort Cloud towards interstellar space. Other robots have landed on Mars and Venus, parachuted into Jupiter and landed on an asteroid. Currently operating robots are circling the Sun and Mars, whilst others are exploring the outer Solar System. When compared with the human presence, currently limited to the International Space Station in Earth orbit, it is clear that robots are the future of exploration. Given the large time delays associated with sending a signal even to our closest planetary neighbour, Mars, remote operation is not feasible.

Launch costs also show huge robotic advantages, especially for long missions. Robotic probes, essentially small spacecraft, are far smaller and lighter than a manned spacecraft with their life-support requirements. Current launch costs are huge and so any reduction in mass is of great benefit.

Underwater exploration presents significant challenges, in some ways even more so than space. Communications are limited to low bandwidth, unreliable acoustic methods, ambient pressure can be thousands of times that of atmospheric pressure and the environment is highly corrosive. This presents a significant engineering challenge.

1.2.3 Present solutions

Current robot explorers rely heavily on two methods: remote operation and/or pre-programmed instruction sets known as scripts. Remote operation has the advantage of the human abilities of adaptability and problem solving, but it is only viable when communications delays are short. Longer delays necessitate increased robot autonomy as with the *Mars Sojourner* buggy. Remote operation also ties up human resources for what may well be a long, repetitive task, essentially more suited to robotics. A good example of this is sub-sea pipeline surveying, where a human operator must drive the Remotely Operated Vehicle (ROV) along many kilometres of pipeline at speeds of around 1 knot ($0.5 m.s^{-1}$).

Scripts are commonly used in space probes and underwater survey vehicles. The robot follows a set of instructions provided by the (human) mission programmer, such as 'go to x, survey area y, go to z'. A limited number of problems will have been foreseen and appropriate reactions included in the script. For example, underwater robots commonly have a line each mission script commanding them to surface in the event of a water leak.

Scripts provide a limited amount of autonomy, whilst remote operation demands some level of autonomy for all but the smallest communications delays.

1.2.4 Future trends

Current autonomous robots are very much geared towards surveying, mainly because this removes most of the need to interact with the environment. It is easier for a sub-sea robot to fly a grid pattern above the seabed than to descend to the rugged terrain of the ocean floor and dock. This lack of intervention ability severely limits their usefulness.

For instance, NASA is planning a manned mission to Mars. In order to increase the probability of success they plan to send ahead a robotic construction team to build a base, providing shelter, fuel and communications to the arriving humans. It would be nearly impossible to accomplish this task using scripts due to the large number of unforeseen problems that are bound to occur.

A similar approach is used underwater. Some oilfield installations have been created using ROVs without a single human entering the water. Although this is far safer than older methods it is still costly in terms of manpower and surface ships.

1.3 A Common Limitation

The underlying problem with autonomous robots is a lack of autonomy. Autonomy can be defined as 'the ability to function independently, without outside help'.

A major limitation of current generation autonomous vehicles is frailty in the face of unexpected events, such as component failure or environmental interaction. Most autonomous robotic mission controllers are based on scripts using specifically coded routines to cope with such events, but for this to be effective the mission programmer must predict all such situations. This is clearly impractical. Embedded planners provide the capability to re-plan the mission but their effectiveness varies with the quality and scope of information provided to them: *"Incorrect information results in unsatisfied preconditions for actions and plans."*

1.3.1 Towards a Solution

In order to cope with unforeseen faults it is vital to have accurate information on what has gone wrong. Current generation mission controllers have no explicit fault diagnosis systems apart from simple rule bases, which can only recognise specific, foreseen faults. More powerful fault diagnosis systems provide the capability to diagnose unforeseen faults, although each system has its own weaknesses.

By providing explicit fault diagnosis to the autonomous vehicle's onboard mission controller the ability to cope with faults should be enhanced due to the extra information provided.

1.3.2 Benefits

The benefits of providing accurate fault information to an onboard mission planner include:

- Enhanced re-planning
- Knowing when a mission is still attainable (and when to quit)
- Ability to cope with unforeseen problems
- Increased mission robustness
- Increased autonomy

Any autonomous vehicle operator will gain from the benefits described above. Autonomous vehicles will have an increasing ability to cope with the myriad of problems, both internal and external, that plague any immature, high technology system that must interact with the real world.

1.4 Use of an AUV for System Evaluation

The vehicle chosen for evaluating this research is an Autonomous Underwater Vehicle, or AUV. Underwater robotics presents all the challenges that produce the need for autonomy but in a highly accessible environment.

AUVs have recently become a useful tool for organisations involved in sub-sea operations such as area survey, environmental monitoring or Antarctic ice sheet surveying. An increasing number of companies provide commercial AUVs for just such applications. It seems likely that in the near future AUVs will become one of the most used autonomous robotic tools.

An AUV also provides a cheap, realistic platform for real-world development without resorting to the unrealistic practice of 'corridor robots'. Land vehicles do use a more accessible environment but they are faced with the difficult problem of navigating terrain, which is outside the scope of this research.

Although this research is aimed at all autonomous robots it is felt that using an AUV for evaluation will provide a realistic research platform with immediate benefit to the AUV community.

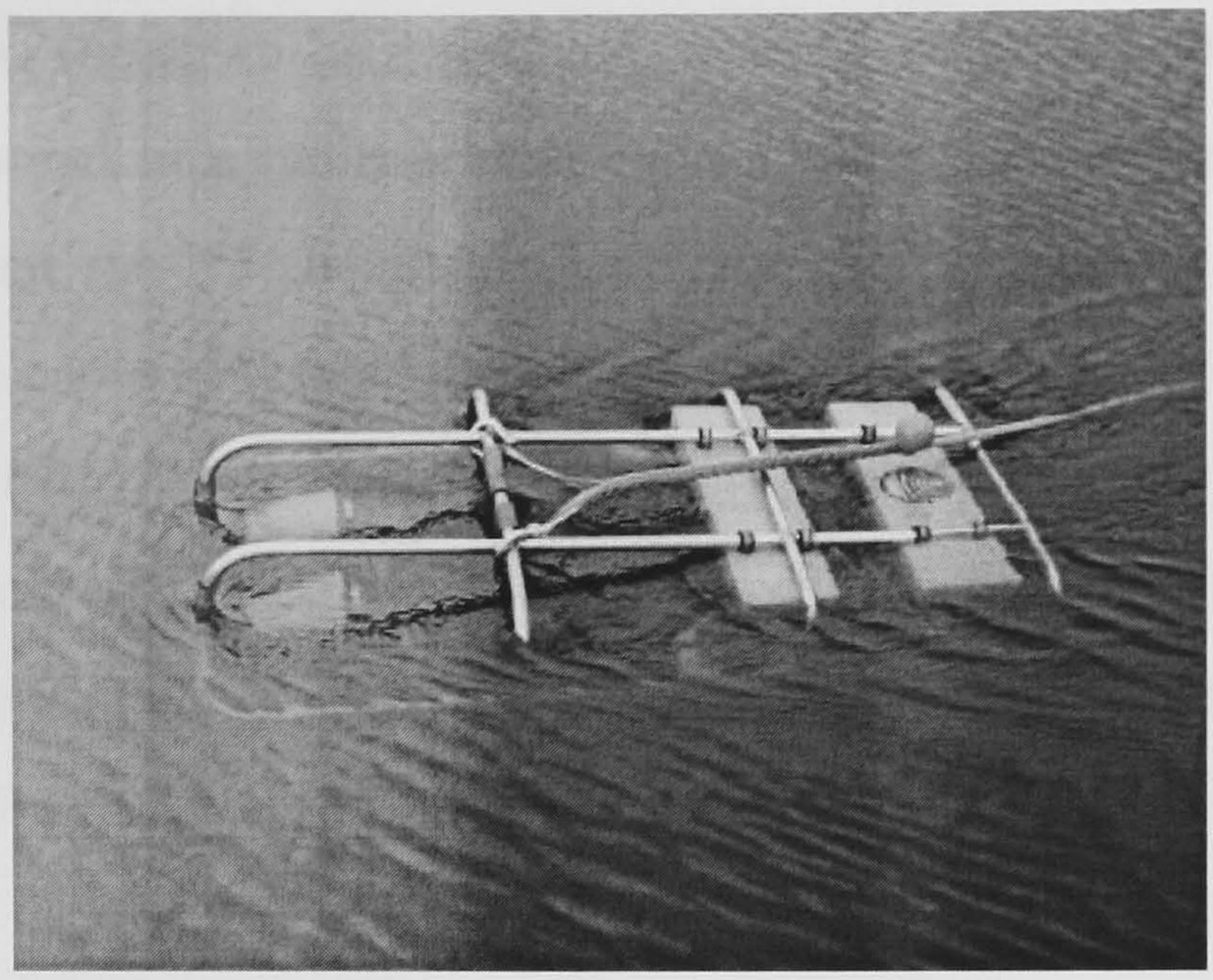


Figure 1.1: RAUVER, The Evaluation Vehicle

An autonomous underwater vehicle, also capable of remote and tethered autonomous operation, was designed by the author during the first year of this research. The vehicle is called RAUVER, an acronym for 'Remote / Autonomous Vehicle for Experimentation and Research'. During the first year the author also supervised RAUVER's construction.

1.5 Aims

The aims of this research are:

- To aid increased vehicle autonomy by enhancing the vehicle's ability to deal with unforeseen events.
- To provide detailed, accurate information on unforeseen faults to the onboard mission controller.
- To evaluate the above system on a real autonomous vehicle under real operating conditions.

To achieve these aims this research proposes the use of on-vehicle Integrated Diagnostics. The proposed architecture for Integrated Diagnostics, RECOVERY, provides integration and methods for the use of heterogeneous design knowledge, heterogeneous diagnostic systems and domain-independent diagnostic knowledge.

RECOVERY is shown to be a detailed, working and extended implementation of a concept proposed in a multi-organisational study. The organisations involved included NASA and the US Department of Defense.

Although the results of that study were not published until after the first year of this research the RECOVERY concept was already in place and found to be more detailed and wider ranging than the study proposal. RECOVERY's architecture and operation also already existed and the implementation was well under way, effectively making it the state of the art for Integrated Diagnostics. This is supported by the literature review.

1.6 Summary of Originality

The core of the RECOVERY system is a dynamic, relational construction model, loosely based on partitioned semantic networks, to tie together different types of diagnostic systems and design knowledge. Using this Relational Model in conjunction with a focus of suspicion method the diagnostic search space may be greatly reduced, particularly when both multidimensional and dynamic diagnostic techniques are required.

The use of the Relational Model allows the use of domain-independent diagnostic knowledge and methods, which enhance the ability to diagnose completely unforeseen faults. The use of domain-independent techniques is investigated. Work is also presented on the types of knowledge available on an autonomous robotic vehicle and how this knowledge may be used to aid diagnosis.

This is the first time that Integrated Diagnostics has been successfully focussed, implemented and evaluated on a real autonomous vehicle performing real missions.

Two papers arising from this research are:

- K. Hamilton, D.M. Lane, N.K. Taylor, and K.E. Brown. *Enhancing AUV fault-diagnosis capabilities with the RECOVERY system.* Accepted for Publication by the IEEE Journal of Oceanographic Engineering, 2002. (Reference [46])

- K. Hamilton, D.M. Lane, N.K. Taylor, and K.E. Brown. *Fault diagnosis on*

autonomous robotic vehicles with RECOVERY: An integrated heterogeneous-knowledge approach. IEEE International Conference on Robotics & Automation, Seoul, Korea, May 2001. Page(s): 3232 -3237 Volume 4. (Reference [45])

1.7 Thesis Structure

Chapter 2 provides some background on mission controllers, modularity and automated diagnostics. Chapter 3 surveys and critiques the available literature on individual, hybrid and integrated diagnostic systems. Chapter 4 presents the RECOVERY system, starting with detail on the originality and showing where this is evaluated, then moving through the concepts involved in RECOVERY. Chapter 5 details the operation of the overall RECOVERY system and its individual modules. Chapter 6 describes the evaluation platform (RAUVER) and the evaluation locations. Results for the novel Domain-Independent Diagnostic methods proposed in this research are then given and discussed. Chapter 7 gives and discusses results for the full RECOVERY system and its method for Search Space Reduction. Chapter 8 draws conclusions and details further work necessary to extend and improve the RECOVERY system.

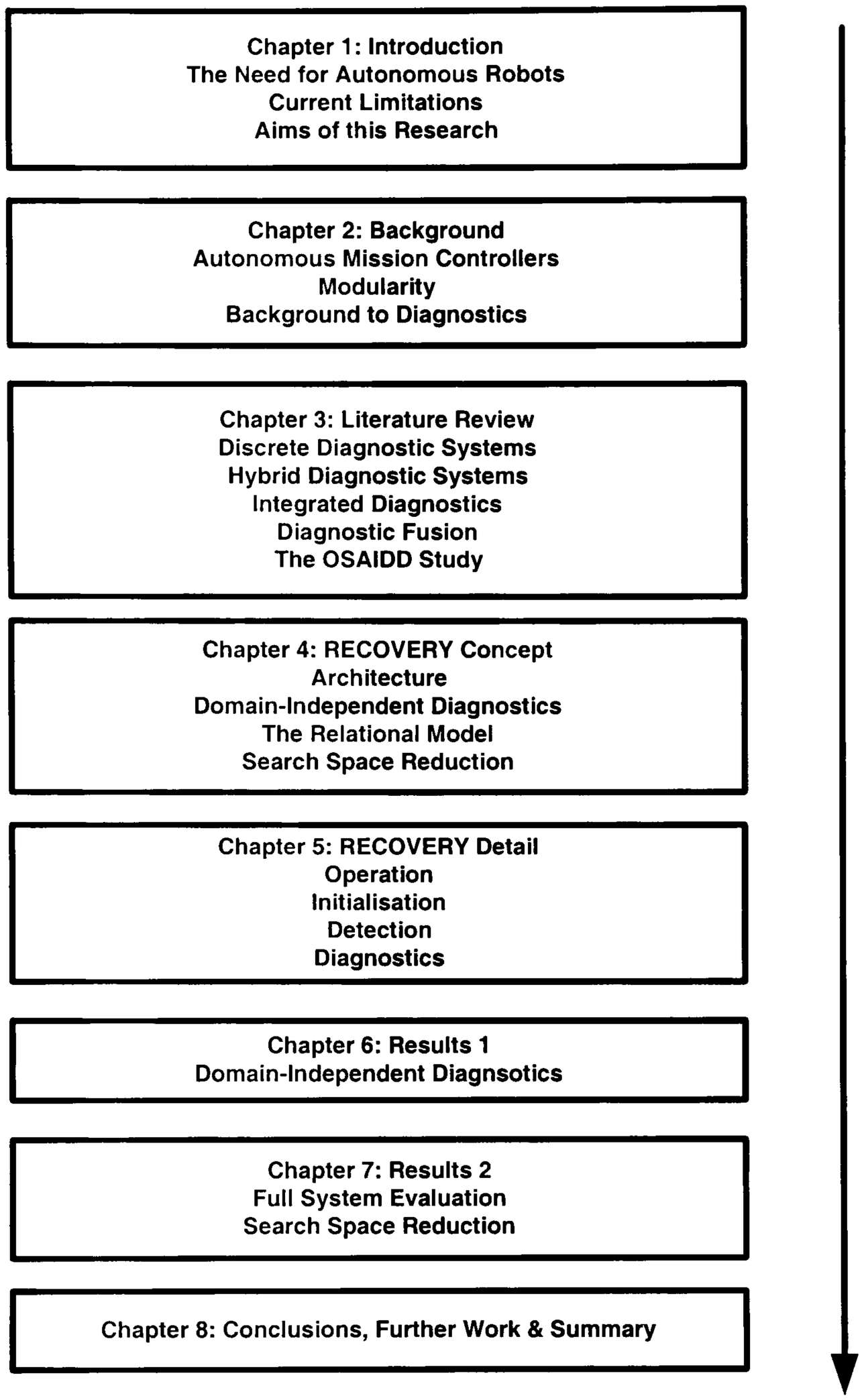


Figure 1.2: Thesis Flow Diagram

Chapter 2

Information on Background Issues of this Research

2.1 Aim of this Chapter

For a robotic vehicle to be able to autonomously perform a mission it must have some form of onboard mission control to direct the robot's actions. The purpose of this research is to provide fault information to this onboard mission controller to enhance the robotic vehicle's autonomy. In order to achieve this it is necessary to understand the different types of mission controllers and show how a diagnostic system could interact with them.

The concept of an architecture for autonomous robotic mission control is explained, popular examples of these architectures given and a common weakness highlighted.

The relationship between faults (what is actually wrong), observations (how the fault appears) and diagnoses (what is thought to be wrong)is discussed. The diagnostic process is then described, particularly detecting, diagnosing and accommodating (or recovering from) a fault.

This research is intended to be useful for a variety of robotic platforms, not just Autonomous Underwater Vehicles. To achieve this it is necessary for the architecture to be platform independent, flexible and versatile. A powerful method of providing these attributes is modularity which is discussed in this chapter.

2.2 Approaches to Mission Control Architectures

An architecture for controlling an autonomous robot *"... defines how the job of generating actions from percepts is organised"* (Russell & Norvig [90]). In other words, an architecture defines how to make the robot correctly respond to sensor

inputs in such as way as to do what you want it to do.

An architecture is needed as the task of making autonomous robots do what is required of them is generally too complicated for any individual system or module. Usually many different systems must work together, an architecture defines how they interact. The robot control architecture and the modules contained within it are called a mission controller. The ultimate aim of a mission controller is to get the robot to do what is required of it, even if that robot is faced with unforeseen and adverse events.

There are many types of mission controllers; three of the most popular types and some hybrids are discussed later in this chapter after a brief overview of some key concepts.

2.2.1 Goal Orientation

A natural way for humans to think is to break a mission into a set of goals, each of which must be achieved in order to complete a mission. It is common for these goals to be achieved in a temporal order, with conditions given for when a goal is completed. For instance, it is common for a military pilot to be given orders detailing a set of waypoints to follow to the target area, a designated target, an action to be performed on that target (bomb it) and a set of waypoints to follow home. Each goal must be achieved in turn to complete the mission.

Typical goals for AUVs may be: head to target, land, takeoff, get GPS fix, dive to depth, dive to altitude and other similar goals.

Goals can have different priorities. In the above example, going to a particular waypoint would probably have a lower priority than dropping the bomb on the target. A good mission controller should be able to determine that some goals can be sacrificed in order to complete the more important ones. The overall purpose of the pilot's mission is, after all, to bomb the target. Returning home is usually a lower priority.

Goals can also be constrained with a set of parameters that must not be violated. For instance, if an AUV was performing a sea-bed sample of an area it would need to land gently, with a goal similar to that shown in Figure 2.1. Constraints provide

more information for planners to take into account. Violated constraints may provide useful information to a diagnostic system.

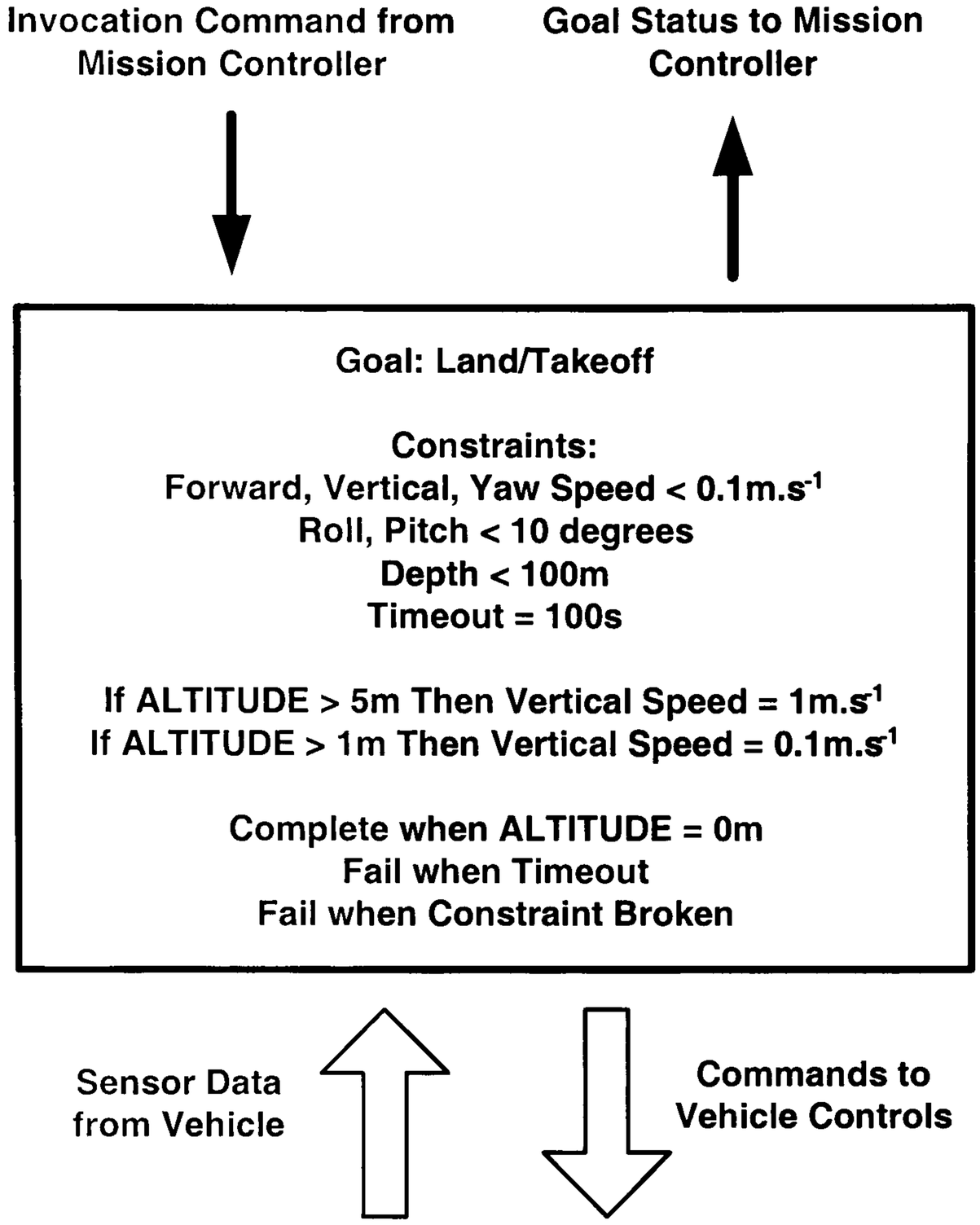


Figure 2.1: A Constrained 'Land Gently' Goal for an AUV

2.2.2 Robustness

A system is said to be robust if it has demonstrated an ability to recover gracefully from the whole range of exceptional inputs and situations in a given environment. This is clearly an important quality for systems that must operate on an autonomous robotic vehicle, particularly the mission controller.

2.2.3 Determinism

Determinism is the attribute of being totally predictable under all conditions. For instance, the environment in which the robot operates is said to be deterministic if

"... the next state of the environment is completely determined by the current state and the actions selected by the [autonomous robotic] agents..." (Finlay & Dix [37]).

2.2.4 World Models

A world model is a representation of the environment in which the autonomous robot operates. There are clear limitations to the extent of such a model due to processor and memory constraints. The detail and accuracy of such a model is limited by the level of technology available on autonomous vehicles.

2.2.5 Environmental Interaction

A major contributor to the difficulty of effective mission control is that the robot must interact with the environment. This has proved difficult within even a relatively structured environment, such as a factory. When the robot must operate in an unstructured environment such as the Martian surface, or in the sea, the task becomes far more difficult.

2.3 The Main Approaches to Mission Control

There are currently three main approaches to mission control: Scripts, Reactive Controllers and Planners. There are also a great many hybrids combining attributes of these, such as Reactive Planners, or Planners running above a fast Reactive layer. This section details the three main types of mission controller and highlights a common weakness.

2.3.1 Scripts, or Finite State Machines

A Script is a set of instructions that the robot must follow in order to complete the mission. Figure 2.2 shows an example script of a segment of a typical AUV survey mission.

A Script is essentially a Finite State Machine. A Finite State Machine is any system which has a limited number of states, and specific events that will make the system change from one state to another. Figure 2.2 shows states, such as 'Dive to

```
1. Transit to Coordinates Xa, Ya
     (If Coordinates = XaYa then Move to Next State)
2. Dive to Depth D
     (If Depth = D then Move to Next State)
3. Survey Box ABCD
4. Transit to Coordinates Xb, Yb
5. Surface
6. Standby

If (Leak_Detected) Then (Emergency Surface)
```

Figure 2.2: A Mission Script for a Survey AUV

Depth D' and transition events such as 'If Depth = D then Move to Next State' which cause the robot to change to the next state, in this case 'Survey Box ABCD'.

There are also some fault states. If a water leak is detected then the robot immediately changes to the 'emergency surface' state. This is the only provision for faults within Scripts, all faults must be foreseen and appropriate reactions planned before the mission begins. Because of this, Script Mission Controllers are not robust, as they cannot cope with unforeseen circumstances.

The main advantage of Scripts is that they are very easy to develop and use minimal computing resources. They are a cheap, practical solution to mission control if the environment is reasonably predictable. For instance, most AUVs operating today use Scripts because most of them conduct surveys well above the seabed in clear water, where there are a minimum of obstacles.

2.3.2 Behavioural Controllers

Behavioural (sometimes known as Reactive) controllers are based on the Subsumption architecture developed in 1986 by Rodney Brooks of the Massachusetts Institute of Technology (M.I.T.). It is essentially a method of joining Finite State Machines in such a way that the robot can react to changes in its environment (Brooks [16]).

Subsumption uses a hierarchy of Finite State Machines. Each Finite State Machine is called a 'behaviour', with higher level behaviour being able to override, or subsume, lower level behaviours.

The classic example is that of a wall following 'corridor robot'. A robot has a low level behaviour enabling it to move forwards, called 'Wander'. A higher level behaviour, activated using a simple sensor and commanding the robot to turn 90 degrees when it encounters an obstacle, is called 'Follow Wall'. When the robot is activated the only active behaviour is 'Wander', so it wanders forwards until it meets a wall. At this point the higher level 'Follow Wall' is activated by the simple sensor. 'Follow Wall' overrides, or subsumes, the 'Wander' output and so the robot turns 90 degrees, at which point there is no longer an obstacle and so 'Follow Wall' is deactivated. 'Wander' is again the only active behaviour and so the robot moves forwards.

In this way, using extremely simple behaviours and a simple sensor, the robot can navigate a corridor quite successfully without large amounts of computing resources. As the behaviours are simple finite state machines the computer can even be replaced with small, hardwired logic circuits consisting of only a few gates, leading to extremely fast reaction times.

Note that there is no representation of the environment, just reactions. Brooks states that *'... the world is its own best model'* in Brooks xx [16]. Subsumption works well for simple tasks but struggles with complex missions. As there is no overall mission plan individual behaviours must be 'tuned' to try to get the robot to react correctly. The robot is limited to reacting to its environment, dealing with real-world complexity by ignoring it.

Behavioural controllers are useful for simple missions, or as a low-level reaction stage, but their usefulness for complex, real-world missions is limited. The emergent properties of their behaviour do not allow for a high level of predictability. This greatly limits their usefulness in current robotic applications where a single robot often has a complicated mission to follow. Current commercial applications, such as operating in areas close to sub-sea structures, require an extremely high level of determinism to ensure that they do not collide with structures or stray into 'no-go' areas.

They are more suited to applications where the robot does not have a particular goal but instead has a more general purpose such as 'wander the corridor without colliding with objects'.

2.3.3 Embedded Planners

Planners use their perception of the environment together with knowledge of how their actions affect that environment. They use this knowledge to try to generate a plan of actions that lead from the present to the desired state. The robot can then follow this plan of actions until the desired state is achieved.

Planners are powerful. If an unforeseen problem occurs then the planner can replan to take into account the problem and complete the mission, making them the most robust of the main mission controllers. They are also goal orientated, with each step of the plan being a goal.

Planners rely heavily on world models to calculate what effect their actions will have, and whether these actions will lead towards the eventual achievement of the desired state. This leads immediately to problems with the complexity of the real world and the difficulty of sensing and modelling it accurately. When taken together with the iterative nature of planners the end result is extremely high processor overheads and long run times.

One of the best known mission controllers to use embedded planners is the NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM) by Albus, Quintero and Lumia [1]. Although initially developed for use with a telerobotic arm on a space station it is a domain-independent architecture suitable for autonomous vehicles.

2.3.4 Hybrids

There are many hybrids between the three paradigms described above, such as a low-level reactive layer providing 'dodge' survival capability below a classical planner slowly providing high level plans. Another method is the reactive planner, where plans are generated in advance and triggered by a reactive layer in response to external stimuli. NASA has had some success with its 3Tier architecture, as described

in Schreckenghost, Bonasso, Kortenkamp and Ryan [92].

2.3.5 Summary of Mission Controller Abilities

Table 2.1 shows a comparison of the abilities of the main types of mission controllers.

Ability	*Script*	*Reactive*	*Planner*	*ReactivePlanner*
Replanning	No	No	Yes	No
Fast Reaction	Yes	Yes	No	Yes
Low Processor Overhead	Yes	Yes	No	Yes
Cope with Unforeseen Events	No	Yes	Yes	Yes
Robust	No	Yes	Yes	No
Goal Orientated	Yes	No	Yes	Yes
Easy to Specify Mission	Yes	No	Yes	Yes
Easy to Develop	Yes	Yes	No	Medium
Explicit Fault Diagnosis	No	No	No	No

Table 2.1: Mission Controller Abilities

2.3.6 A Common Weakness

None of the above mission controllers feature explicit fault diagnosis and so their ability to cope with unforeseen faults is constrained. They rely on 'under informed' or non-existent fault diagnosis strategies.

Scripts rely on having enough states and events to cope with problems. By using explicit fault diagnosis the number of events can be increased, improving the 'coverage' of possible problems. Scripts still require the states to be predicted in advance but the use of explicit diagnosis means that these states may be reached without predicting the events.

For example, by using explicit diagnosis a fault state and matching action for each component can be generated, such as 'if port-stabiliser-fin faulty (fault state) then reduce speed (action)'. Without explicit diagnosis a matching event must also be foreseen to trigger the change to that state; 'if vehicle rolls to port (trigger event)

then port-stabiliser-fin faulty (fault state)'. By using explicit fault diagnosis the trigger event can be generated independently without having to foresee it.

Reactive controllers also rely on events, although these events are usually environmental. The use of explicit diagnosis can increase the 'reaction range' of the robot by providing more events.

Embedded planners provide the capability to re-plan the mission but their effectiveness varies with the quality and scope of information provided to them: *"Incorrect information results in unsatisfied preconditions for actions and plans."* (Russell & Norvig [90]). It is therefore vital that an embedded planner is provided with accurate and detailed information on the problem and the vehicle's degraded capabilities. By providing detailed fault information replanning capability may be greatly enhanced, increasing the robot's capability to cope with unforeseen events.

2.4 Diagnostic System Basics

Diagnostic systems try to identify the causes of faults. In robotic applications the fault is usually a faulty component such as a transistor or a thruster.

Diagnostic systems are commonly used in industrial situations such as chemical plant, reactors etc and military or aerospace applications such as helicopters. They are also starting to appear on domestic cars.

2.4.1 Fault, Observation and Diagnosis Spaces

There are three inter-related spaces involved in fault diagnosis: fault space, observation space and diagnosis space. Each space contains the totality of all possibilities in their respective areas. Fault space contains all faults that can occur on the vehicle, observation space contains all observations from the point of view of the vehicle and diagnosis space contains all possible diagnoses available by using the observation space. Figure 2.3 shows the relationship between these spaces, where the fault space is linked to the diagnosis space through the observation space.

The size of the fault space, or the range of possible faults that can occur on an AUV, when taken together with the unpredictable nature of the environment,

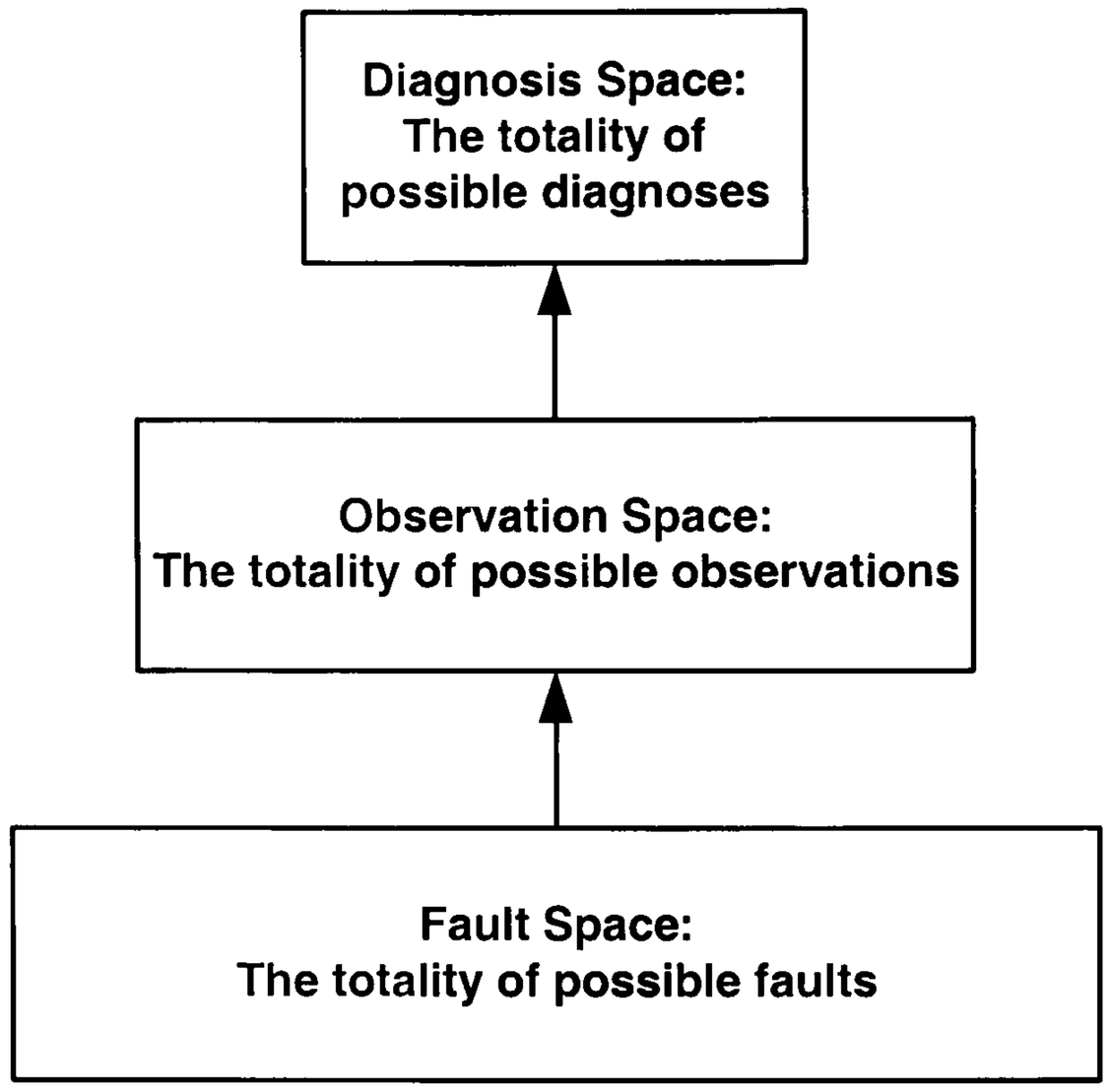


Figure 2.3: Fault Spaces and their Relationships

is extremely large. It is practically impossible to predict (and plan for) all possible faults. This is borne out by practical experience of AUV operations (and almost every other field of engineering). Further, the range of events that can be planned for is limited by the scope of observation, where an observation may be gained from a sensor such as a temperature or attitude sensor.

Whilst fault space may tend towards infinity the observation space can be directly controlled by the vehicle designer. The initial size of the observation space is determined by the number of sensors embedded in the vehicle but it can be expanded by combining observations in a similar way to the concept/percept paradigm. The initial size of the diagnosis space is also determined by the number of sensors. In a typical case the ratio of observation space to diagnosis space is 1:1, that is, each sensor is responsible for a specific diagnosis.

An example of this is a thermostat placed inside a battery; typically, the thermostat tripping will set a flag somewhere in the mission control software saying battery overheating. Often the ratio will be less than 1:1 because some vehicle sensors do not contribute to fault diagnosis, i.e. vehicle attitude sensors will flag that the vehicle is outside its attitude limits but will not be tied to any components. This poor ratio between observation and diagnosis spaces is a feature of current mission-control technologies.

2.4.2 The Component Concept

A component is usually defined as the lowest replaceable unit when using diagnostic systems aimed at providing a solution to a human operator (who must then fix the fault). For instance, to a field engineer the smallest diagnosable component of a piece of equipment may be a circuit board, but at the factory it may be an integrated circuit, or chip. The situation is different for most autonomous vehicles as they are far from human help and cannot fix themselves - the lowest replaceable unit is the entire vehicle. The emphasis must be on finding out what has gone wrong and trying to work around the problem.

For diagnostics on autonomous robotic vehicles a component is not the lowest replaceable unit but the lowest *diagnosable* unit, the level of which is dependent on the size of the observation space (i.e. the number and location of sensors). This research uses a further classification, that of a sub-component, which may only be diagnosed using component-specific, rather than vehicle-specific information

2.4.3 Redundancy

A major part of fault detection and diagnosis is the use of redundancy. There are two main types of redundancy: hardware and analytical. Hardware redundancy utilises different sensors measuring the same data, a difference between the two signals (known as a residual) usually indicates a fault. For instance, if two sensors measure magnetic heading then the readings should be identical unless one of the sensors fails, in which case the signals will differ.

An increasingly popular approach is to use analytical redundancy, where an algorithmic model is used to provide the second signal. Although this requires the use of a microprocessor these are extremely common, fast and light in modern systems. This saves the extra cost and weight of additional sensors and is an extremely versatile approach.

Analytical redundancy can be broken down into two forms: direct and temporal (Magrabi and Gibbens [67]), which are equivalent to the static and dynamic terms used throughout this thesis. Static (or direct) redundancy is when the state measured by one sensor can be determined using the instantaneous outputs of other

sensors. Dynamic (or temporal) redundancy is when the state can be determined only by using the historical data from other sensors.

2.5 The Diagnostic Cycle

The diagnostic cycle is the process of moving from detecting a fault to accommodating (or recovering from) the fault. There are four stages to the diagnostic cycle: failure, detection, diagnosis and recovery (FDDR). Most current-generation diagnostic systems do not provide fault-recovery, these are known as FDI systems (Failure, Detection and Isolation).

The diagnostic cycle can be thought of as the process of answering three questions:

- Is something wrong? (Failure Detection)
- What is wrong? (Diagnosis)
- What can I still do? (Recovery)

2.5.1 Failure

A fault is defined as *"... an unexpected change that leads to the corruption of the overall performance of the system."* (Magrabi and Gibbens [67]). Faults largely break down into three types:

Single/Multiple Faults

A single fault is a discrete fault, caused by one component. Multiple faults, where many components fail, are far harder to diagnose as the symptoms may overlap or interact. This may produce a different, larger set of symptoms than the sum of the individual faults. Some systems assume that there may be multiple-single faults, where many single faults with isolated symptoms may appear.

Structural/Non-Structural Faults

A structural fault is caused by a re-arrangement of the system's physical topology, for instance a transformer breaking loose and shorting against another circuit. These are extremely difficult to diagnose, generally requiring an explicit topological model. Most diagnostic systems regard the topology as fixed in order to ease the diagnostic burden.

Static/Dynamic Faults

A static fault is one that may be diagnosed using information gathered from one point in time, a 'snapshot' of the system. A dynamic fault may only be diagnosed by observing the behaviour of the system over time, a 'movie' of the system. Examples of dynamic faults include intermittent faults, thermal drift and transient problems.

2.5.2 Fault Detection

Once a fault has occurred it must be detected. Some level of detection is usually designed into the system in order to prevent catastrophic failure. For instance, thermostats on power-conversion components signal when they are too hot for continued operation. Another common practice is to watch parameters using software and set a fault flag when a parameter exceeds some preset value. A mission controller may also provide a form of detection by signalling when goal constraints are violated.

2.5.3 Fault Diagnosis

The diagnostic system now takes responsibility for determining what the problem is. This usually takes the form of a component, where a component is the smallest isolated item of the system that can be diagnosed (as discussed in Section 2.4.2.

Diagnostic systems can generally determine what has gone wrong (the circuit board), but not usually why it has gone wrong (a power spike).

2.5.4 Fault Recovery

Once the fault has been diagnosed an autonomous vehicle must then be able to decide how to cope with the fault, that is, how to continue the mission using whatever capabilities remain. Although this stage is perhaps best performed by the mission controller the diagnostic system can contribute. New operational models must be generated so that the planner can correctly calculate whether the mission is still attainable, and if so what actions must be taken to complete it.

2.6 Modularity

An increasing feature of modern design is modularity. The move towards modularity is particularly true of Autonomous Underwater Vehicles due to the wide range of mission types that they are required to perform; a modular strategy is vital in order to achieve efficient flexibility. An information based fault diagnosis system must encompass modularity or its usefulness will decline as the vehicle changes from its initial state.

Two popular methods of implementing modularity are the distributed and centralised approaches, of which the latter is the most common.

2.6.1 Distributed Systems

A distributed diagnosis system would have each major component capable of diagnosing itself and providing this information to the overall diagnosis system. This would be a powerful approach, as it could provide graceful degradation and maximum survivability in case of damage. Of course, it is in case of damage that the diagnosis system is most needed. Unfortunately the need for processing power attached to every component adds a certain overhead in terms of power, weight and space. Although there is a move towards such 'smart' components, it is not yet common enough to merit developing fully distributed diagnosis systems.

2.6.2 Centralised Systems

With the centralised approach various modules can be 'plugged' into a central core. Most current generation AUVs favour a centralised approach with vehicles consisting of a core onto which other sensors or other devices may be added. At present, the centralised modular approach is the most practical solution, although it can lead to the loss of graceful degradation. The long-term aim of any diagnostic system must be survivability in order to provide diagnostic information no matter what the fault.

2.7 Chapter Summary

This chapter showed the two main paradigms used for autonomous mission control; functional and behavioural. Behavioural approaches were shown to be unsuitable for current applications due to their emergent properties that severely limit predictability and make it difficult to specify a complex mission. The functional approach to mission control is more useful for today's applications, although mission controllers that combine both approaches, particularly NASA's NASREM architecture, have proved successful.

Different types of mission controllers implemented on autonomous robotic vehicles were then discussed and a common weakness highlighted: the lack of explicit diagnosis.

Background information was presented on the necessity for a modular approach to the design of a diagnostic system and the two main methods discussed; centralised and distributed. Although distributed systems have advantages, such as graceful degradation, this approach is not yet common enough in current generation autonomous robots to merit focussing on a fully distributed diagnostic system.

Background concepts of automated diagnosis were then shown. The relationship between faults (what is wrong), observations (how the fault appears) and diagnoses (what is thought to be wrong) was discussed and the size of the observation space shown to be crucial.

The component concept, usually based on the level of Least Replaceable Units was discussed and it was shown that on autonomous robots during a mission the

least replaceable unit is the vehicle itself as, at present, the robot cannot fix itself. This leads to the focus being on finding what has gone wrong and working around, not fixing, the problem.

Redundancy was shown to have two forms; hardware and analytical. Hardware redundancy is where two separate components (in this case sensors) perform identical or overlapping tasks, so that if one of them fails the residual difference in signals may be used to detect the fault. This is a powerful approach but it has a size, weight and power overhead. Analytical redundancy uses models to provide the second sensor signal, this is less costly in terms of size and weight but it does require more processing power.

The diagnostic cycle of failure (something goes wrong), detection (realising something is wrong), diagnosis (finding out what has gone wrong) and accommodation/recovery (working around the fault) was then described.

The next chapter details different approaches to diagnostics, ranging from individual system-specific approaches through to Integrated Diagnostic frameworks. The multi-organisational study; Open Systems Approach to Integrated Diagnostics Demonstration (OSAIDD) is discussed and its strengths and weaknesses highlighted.

Chapter 3

State of the Art of Autonomous Diagnostic Systems

3.1 Aim of this Chapter

This chapter aims to show that there are many diagnostic paradigms and systems but little attempt to provide a framework for Integrated Diagnostics within which they can all operate.

The first part of this chapter provides a brief overview of individual diagnostic paradigms, particularly popular approaches such as Neural Networks, Rulebases and Model Based Diagnosis. Examples of specific systems utilising these approaches are given.

Hybrid diagnostic systems utilising two or more paradigms are then detailed. These hybrid systems usually try to combine the complementary strengths of different paradigms but are not an attempt to provide a generalised framework for integrating different diagnostic paradigms.

Approaches to Integrated Diagnostics are then detailed. Limitations of the existing attempts, particularly the United States Department of Defense's Open Systems Approach to Integrated Diagnostics Demonstration (OSAIDD) by Freschi *et al* [40] are shown.

3.2 Individual Diagnostic Systems

This section presents a description and examples of a representative sample of the individual diagnostic paradigms. Although there are many different forms of diagnostic system, many of which are tailored to specifically suit the developmental system, they broadly fall into model-free and model-based techniques (Alessandri

[3]). Model-free methods do not rely on using models of the system to be diagnosed, an example of this is a Rulebase. Model-based methods use mathematical models of the system to achieve diagnosis, such as a typical Model-Based Diagnosis implementation.

3.3 Model-Free Diagnostic Methods

This section details various types of model-free diagnostic systems.

3.3.1 Rule Based Diagnostic Systems

The Rulebase is the simplest form of detection and diagnosis available today, despite which it is still the subject of research, particularly by NASA. Rulebases are a common, first attempt solution to providing fault detection and diagnosis for an autonomous vehicle.

Rulebases are said to be first generation knowledge-based systems, where a 'knowledge engineer' takes the knowledge of a human expert and converts this knowledge in to a form understandable by a machine. This usually consists of rules, in the form of if - then statements, and commonly embraces fuzzy logic to give some degree of generalisation. These rules use knowledge gained by outside observation of the system rather than a representation of the internal mechanisms, and as such are termed 'shallow' knowledge.

Rulebases are often used for evaluating systems that must interact with diagnostic systems, as they are so easy to develop. For instance, Yan, Nakamura, Arai and Kuwahara [106] developed a system for remote diagnostics using standardised message protocols. To evaluate it they used a simple Rulebase for proof-of-concept work rather than spending crucial time developing a powerful diagnostic system, such as a neural network.

Tunstel, Howard and Seraji [100] have recently developed a fuzzy Rulebase system for reasoning about the safety of a planetary rover on challenging terrain. Dai and Sugisaka [28] have developed a probabilistic fuzzy Rulebase for diagnosing a power plant, with the stated goal of 'quick diagnosis'. Huang, Yang and Huang [52]

have developed a fuzzy Rulebase which is then modified using genetic algorithms to increase diagnostic performance.

NASA has also developed an advanced Rulebase system known as SHINE (Spacecraft Health INference Engine), which is detailed later in this chapter.

The continuing research in Rulebases shows that, despite their simplicity, they are a valid and useful diagnostic tool.

3.3.2 Case Based Reasoning

Case based reasoning uses the concept of analogy in a similar way to legal trials. All the cases (examples) are stored in a database, called a case base. New situations, such as faults, are compared with the examples in the case base and the best match is found. If an exact match is found then the example may be simply repeated, if not then the reasoning system is used to derive a response.Case based reasoning is useful for fault diagnosis when observations, or attributes, are missing, as a simple scoring method may be used for matching.

Diagnostic systems utilising case based reasoning include Penido, Nogeuira and Machado [81] and Lewis [66].

3.3.3 Vibrational

A popular method of fault detection and diagnosis with physical structures is vibrational analysis. In some cases the structure is excited with a specific signal, often a 'lamb' wave, which is distorted by a structural defect. The returned signal is analysed and the presence of the defect deduced. For instance, Cattarius and Inman [20] use a reference signal combined with a vibrational analysis algorithm on a helicopter rotor blade to recognise individual defects. They find that different defects produce individual signatures, which is desirable for fault detection and diagnosis.

Another approach is to identify vibration signatures while the structure is in use. When mechanical structures wear they often produce a different noise to when in good condition. For instance, a car wheel bearing is near silent when healthy, but if it is worn then it will emit a grinding noise which gets worse as the bearings deteriorate. Yen and Lin [110] use wavelet packet transforms for analysis rather

than the conventional Fourier transform. Chen, Sasaki, Nakayama and Toyota [24] use a sound sensor mounted on a robot arm together with genetic algorithms that control the position of the arm. In this way they steer the sound sensor towards the faulty part, in their evaluation system this is a particular bearing.

Whilst this is a useful technique for analysing physical structures, its usefulness in other areas is limited.

3.3.4 Probabilistic Diagnostic Techniques

Probabilistic techniques are very useful for dealing with incomplete knowledge, which with diagnostics often relates to incomplete observations of the fault.

Bayes' rule provides a method for estimating the probability (degree of belief) of a state given the probabilities of three other related states, one of which is conditional and the other two unconditional (Russell & Norvig [90]). In short, Bayes' rule allows the computation of unknown probabilities from known, stable ones.

When there are a number of states it is possible to compute the probability assignments to all states, this is known as the joint probability distribution. This completely specifies the degree of belief in all states in the domain given the set of observations.

If there is more than one conditional probability, derived perhaps from different sensor readings, it is possible to exploit a simplified form of Bayes' rule that uses the concept of conditional independence. This greatly reduces the number of conditional probabilities that must be specified when working with multiple observations.

3.3.5 Network Approach

Yan, Nakamura, Arai and Kuwahara [106] propose an approach based on the Simple Network Management Protocol. They concentrate on specifying a global communications protocol for autonomous robotic diagnostics, showing how this could enable 'remote diagnosis' of problems. They evaluate their ideas using a Rulebase combined with active testing.

3.4 Model-Based Diagnosis Systems

Model based diagnosis systems are said to be second generation knowledge based systems, where a model constructed from detailed in-depth knowledge, preferably from first principles, of the system is used. This is known as 'deep' knowledge, said to be gained from observing the system from the inside.

There are many types of models available for use in diagnosis. Generally, an explicit model of the device is created from knowledge of the system's structure and the physical (or other) laws that govern its behaviour. The movement models discussed above tend to be equational, with the level of detail determined by the amount of design effort applied to the modelling process.

3.4.1 Operation

By using models the behaviour of a system can be predicted and then compared to the actual, measured behaviour. Any discrepancies are likely to indicate a fault, which the diagnosis system can then try to replicate by changing model parameters. If the model can match, or at least approximate, the faulty system's behaviour then the altered variables, which are likely to represent components or subsystems, are candidates for the fault.

3.4.2 Model Generation and Selection

Ideally, it would be possible to create an effectively perfect model of the vehicle, correct in every detail, so that any possible faults could be tracked down by operating on the model. This level of modelling is outside current capabilities, even if such a model could be constructed the amount of time needed to operate with such a large number of variables would be prohibitive.

An alternative approach to perfect model building is realistic model building. The most common model available to AUV engineers is a vehicle-movement model, often generated at the prototype stage by using towed scale-model techniques.

There are many types of models available for use in diagnosis. The movement models discussed above tend to be equational, with the level of detail determined by

the amount of design effort applied to the modelling process. Other types of model include functional, abstract and black boxes; for a guide to the selection of suitable model based diagnosis techniques see Chantler et al [23].

Available on-vehicle computing platforms require limiting the depth of the model and managing the time available for diagnosis, as in the hierarchical approach of Aldea [2]. In this case the simplistic model would probably have a term that lumped together all the hydrodynamic effects of one side of the vehicle, thus reducing the number of variables and the run time. The more detailed model would have individual terms for every component on that, which would mean more variables and a higher run-time.

Caccio, Indiveri and Veruggio [19] have performed valuable work on the generation of models using the vehicle's own sensor suite, which substantially reduces the cost of generating the models available for diagnosis. Chantler et al [23] provide a thorough examination of the suitability of various types of models to differing applications.

3.4.3 Multidimensionality

A Model based diagnosis capability may be extended by using multidimensional models, where a dimension is a point of view from which a fault is apparent. For instance, a vehicle that is diving when it should be surfacing will show a fault in the movement dimension. Most robotic vehicles possess a variety of onboard sensors using which it is possible to monitor a number of dimensions apart from movement. For instance, temperature sensors can monitor the thermal dimension and electric-current sensors can monitor the power dimension.

A model that covers just one aspect of a vehicle, such as case movement, is known as a monodimensional model. By modelling additional dimensions, such as thermal and power, and using these multiple viewpoints it is possible to greatly enhance the diagnosis capability.

For instance, as well as looking at vehicle movement, the power consumed may also be monitored with this additional information used to aid diagnosis. By extending this information through time, it is possible to access the temporal dimension,

enabling the diagnosis of dynamic faults using a system such as Shen [93].

It is possible not only to generate, but also to monitor and use a heterogeneous, multi-dimensional model of a vehicle for fault diagnosis.

3.4.4 Using World Models

Current generation AUVs have an emphasis on environmental sensors as the primary purpose of AUVs is to navigate, survey and intervene. As world models mature and processing speed increases these environmental sensors will become increasingly useful for fault diagnosis as environmental effects will become diagnosable.

For instance, modelling environmental effects such as water currents flowing around seabed topology consumes a great deal of processing power and is currently of little use to a model based diagnostic system. If an AUV were caught up in such currents, the diagnosis system would be likely to diagnose a fault with the thrusters or attitude fins leading to unnecessary mission termination. But if the model based diagnosis system could use the world model for diagnosis then diagnostic capability would be increased, and the likelihood of premature mission termination greatly reduced.

3.4.5 Indirect Sensing

Using models it is possible to predict values for measurements not directly sensed by any sensor. For instance, an AUV may not have a hull-breach sensor to detect when the vehicle is flooding, but by using a model of the vehicle's hydrodynamics the added weight may be detected and so, indirectly, the presence of water.

3.4.6 Examples of Model-Based Diagnosis Systems

The General Diagnostic Engine (GDE) by de Kleer & Williams [30] was one of the first model based diagnosis engines. It used models of the correct behaviour of components, which limits the amount of information available for diagnosis

Shen and Leitch [93] concentrate on diagnosing dynamic systems using dynamic models coupled with a synchronous tracking mechanism. System behaviour is observed over time and compared to the modelled behaviour. This is a lengthy process

if used with multiple models and they recognise the need for a refinement of search techniques.

Mosterman and Biswas [75] use bond graphs to constrain dynamic models as much as possible and so limit the amount of computation involved. This diagnosis system depends on analysing the transients generated by faults and so cannot analyse steady state, or static, faults.

Perrault and Nahon [82] have adopted a similar approach using an algorithmic model of the vehicle's dynamics for control. This allows them to use a synthesis of actuated and non-actuated degrees of freedom to allow continued vehicle operation with faulty thrusters.

Other approaches to Model Based Diagnosis may be found in Alessandri, Caccia and Veruggio [3], Caccia, Indiveri and Veruggio [19], Giovanni Indiveri's Ph.D. Thesis [57], Vukic, Ozbolt and Pavlekovic [103], Boppana and Fujita [10], Misra *et al* [73], Huang and Cheng [54], Huang [53], Larsson [63], Lavo, Chess, Larrabee and Hortanto [64], Zhang and Li [111], Perrault and Nahon [82], Portinale [83], Jackson [58], Rae and Dunn [88], Kolcio, Hanson and Fesq [61], Kurien and Clancy [62], Bos, Van Gemund and Witteveen [11].

3.4.7 Hierarchical Model-Based Diagnosis

Aldea [2] uses hierarchies of models to provide 'anytime diagnosis', where the level of diagnosis performed is matched to system deadlines using a tree of models of varying degrees of detail. This is a powerful approach, well suited to deadline orientated real-time systems as found in most AUVs. This system currently concentrates on the diagnosis of static faults.

3.4.8 Domain-Independent Diagnosis

Ng and Chow [76] propose an expert system designed to work closely with and advise a human test engineer working on a faulty circuit. They use a structural model of the circuit, based on a semantic network. The diagnostic system proposes points on the circuit to be measured, the engineer measures them and states whether they match the normal readings. The diagnostic system then uses domain-independent

trouble-shooting heuristics and the structural model to guide further measurements until the fault is isolated.

Ng and Chow have developed a Rulebase containing 74 domain-independent trouble-shooting rules/heuristics. They find that when evaluated on a hypothetical circuit the human working with the diagnostic system often performed better than a human working conventionally with the circuit diagram. When evaluated on a real circuit the conventionally working human was faster as humans bring other knowledge and senses to diagnosing a real circuit, such as audible noise, smell, hot circuits etc.

3.4.9 NASA Diagnostics

Perhaps the most powerful vindication of the use of diagnostics is the amount of research that NASA is conducting in this area. They have developed SHINE, the Spacecraft Health Inference Engine, (James and Dubon [59]) which is an advanced Rulebase and inference engine focussed on deployment on systems with limited time and computing resources, such as a space probe. There is also BEAM, Beacon-based Exception Analysis for Multi-missions, which is a highly mathematical approach "*... grounded in a radical approach to complex systems analysis, combining advances in adaptive wavelet theory, nonlinear information filtering, neuro-fuzzy system identification and stochastic modelling.*" (James and Dubon [60]). SHINE and BEAM are also being used by Unmanned Aerial Vehicle developers such as Lockheed Martin and JPL (Schaefer *et al* [91]).

Onboard diagnostic systems are also thought to be crucial to manned space missions, such as the manned mission to Mars programme; "*During a [manned] Mars expedition, autonomous plant operations [life support, transport etc] would allow unmanned systems to prepare for human arrival, protect crew and resources by rapidly responding to critical failures, and free humans from routine operations, allowing greater exploration.*" (Kurien and Clancy [62]).

Livingstone is a diagnostic system that implements Model Based Diagnosis by modelling each component as a Finite State Machine, and the whole spacecraft as a set of concurrent, synchronous state machines. This allows Livingstone to track

concurrent state changes caused by component failure. Livingstone is currently deployed in the Remote Agent Experiment.

3.4.10 NASA's Remote Agent Experiment

Advanced diagnostic methods, such as the Livingstone diagnostic module, have been evaluated on NASA's Remote Agent Experiment, as detailed by Bernard *et al* [8]. The Remote Agent Experiment is deployed on the Deep Space 1 probe. The spacecraft has a goal orientated mission controller (rather than a list of actions) which incorporates a model-based fault diagnosis module. Under fault conditions the diagnosis module is responsible for presenting detailed fault information to the mission controller. With the aid of this fault data the mission controller has an enhanced ability to plan around the fault.

An interesting aspect of the Remote Agent approach is the focus on integration of mission control and explicit diagnosis. The model-based diagnosis system is conventional, static and monodimensional, but when combined with the onboard planner the ability of the spacecraft to cope with unforeseen faults was increased (Bernard *et al* [8]). The melding of mission controllers and explicit diagnosis is sure to have a significant effect on the autonomy of robotic vehicles.

A case study on the Remote Agent Experiment found: "*The approach [on Deep Space 1/Remote Agent]... is to separate domain-specific monitoring specification from the [diagnostic] architecture chosen. This ... leads to reducing the time and cost of design and test through the reliable reuse of monitor software*" (Freschi [39])

3.4.11 Finite State Machines

Diagnosis may also be performed using finite state automaton models, which partition the state-space into finite regions and model the system's trajectory as it passes through these regions. This effectively breaks the state space into localised sets of if-then rules, represented as Finite State Automata Tables, which are then operated upon by a fault-detection and isolation algorithm.

Ramkumar *et al* [89] provide methods by which the localised tables may be produced automatically from the system's dynamic equations, such as standard

differential equations. This work is then extended by Xi *et al* [105], in which they highlight non-diagnosable circumstances.

A similar approach is adopted by Belhassine-Cherif and Ghedamsi [7] for distributed systems where the system specification is in the form of communicating, non-deterministic finite state machines.

3.4.12 Thruster Control Matrix

Some AUVs use matrices of thruster forces to detect, diagnose and accommodate faults with vehicles that have redundant control of degrees of freedom, such as having more than one combination of thrusters to move forward. The matrices are used for calculating required thruster forces for movement in a given direction, in the event of a thruster being diagnosed as faulty the matrix is reconfigured to use a different thruster solution to attain the desired motion in spite of the faulty thruster.

Yang, Yuh, Suk and Choi [107] [108] have tested one of these systems on the ODIN (Omni Directional Intelligent Navigator) AUV. Fault detection is performed by using thruster speed feedback and a simple model of thruster performance, any discrepancy indicates a faulty thruster. A Rulebase identifies the faulty thruster.

This approach to accommodation is restricted to vehicles with redundant control over their degrees of freedom.

Bayesian Belief Networks

A Bayesian Belief Network is used to represent the dependence between variables and to provide a precise specification of the joint probability distribution. The network is made up of nodes and links, where the nodes are variables and the links represent the influence of one node on another. A Bayesian Belief Network is both a representation of the joint probability distribution and a set of conditional independence methods. It is possible to update observations and use inference algorithms to calculate the probability of a given state, such as 'power surge', given a set of observations.

Bayesian Belief Networks are a powerful diagnostic technique, examples include Pryztula and Thompson [87], Molnar [74], Hunt, von Konsky, Venkatesh and Petros [55] and Vieira and Theys [102].

Kalman Filters

A Kalman filter allows prediction and estimation of large numbers of states to be made using simple matrix techniques, as long as certain assumptions are made. These are that:

- Each state variable is real-valued
- Each state variable is Gaussian distributed
- Each sensor is subject to unbiased Gaussian noise
- Each action can be described as a vector of real values, one for each state variable
- The new state is a linear function of the previous state and the action

Alessandri, Caccia and Veruggio [3] use a dynamic model of an underwater vehicle combined with a bank of Extended Kalman Filter Estimators for fault isolation. The emphasis is on accurate fault detection and diagnosis of thruster failure. Also see Magrabi and Gibbens [67].

Markov Models

A Markov model is a way of describing a process that moves through a series of states, with the model describing all possible paths through the state space with a probability value for each. The Markov property is that the probability of moving from the current state to another depends only on the current state, not any prior part of the transition path.

Washington [104] combines a Kalman filter with a Markov model for diagnosis, described more fully in the Hybrids section later in this chapter.

3.4.13 Failure Modes and Effects Analysis (FMEA)

Price and Taylor [85] propose the use of models generated at design time using the Failure Modes and Effects Analysis (FMEA) design discipline. The FMEA models are used to generate sets of possible faults at design time (offline generation). This

combines some of the strengths of model based diagnosis with reduced online run time, but sacrifices the ability to diagnose unforeseen faults.

3.4.14 Unified Geometric Approach

Dunia and Qin [36] use a geometric approach. A model is created of the normal operation of the system, under fault conditions a vector is defined from faulty to normal operation. This system is limited to single dimensional, steady state faults. Dunia and Qin then extend their geometric approach to multidimensional faults [35].

3.4.15 Neural Network

Neural networks are a popular diagnosis technique which are increasingly being combined with other approaches to form hybrid systems. A neural network consists of individual nodes called neurons, or units. Units are joined together by links, which typically have a weight associated with them. Weights are the primary means of long-term storage, with learning effected by adjusting the weight values. Each unit has a set of input links from other units, a set of output links, an activation level and a method of computing the activation level at the next time period given the current inputs and weights. The network effectively acts as a parallel computer, with each unit performing local calculations based on local inputs without any global control over the system. Much research has been performed on neural networks and there are many variations.

Neural networks are very good for classifying patterns (pattern recognition) and have the ability to learn from data sets, either with supervised or unsupervised learning. When correctly trained they are capable of a level of generalisation, for instance being able to recognise individual handwriting after being trained on a variety of handwriting styles.

Unfortunately, their parallel nature does not lend itself to the current generation of computers, which are serial based. Typically, a serial computer may take several hundred iterations to calculate the state of a simple network, whereas a parallel implemented neural network will only take one iteration. Neural networks are also

difficult to analyse mathematically and so are often referred to as 'black boxes'. If a neural network is used for diagnosis a common problem is that a diagnosis is provided, but it is very difficult to discover the reason for the diagnosis.

Takai and Ura [97] concentrate on actuator failure. A neural network is used as a dynamic model of the vehicle's movement. Diagnosis is achieved with pre-set 'diagnosis motion sequences' during which the vehicle performs a standard series of manoeuvres. A look-up table is then used to relate defective movement to defective components. This system has the ability to use the neural dynamic model for dead-reckoning control of the vehicle if a sensor fault is diagnosed.

Other neural network approaches to diagnosis include Spina and Upadhyaya [94], da Silva, Insfran, da Silveira and Lambert-Torres [27], Vemuri, Polycarpou, Sotiris and Diakourtis [101], Trunov and Polycarpou [99], Tarrassenko, Nairac, Townsend, Buxton and Cowley [98] and Butler and Momoh [18].

3.5 Hybrid Diagnostic Systems

Some researchers have used a hybrid of different diagnostic systems in order to increase overall diagnostic ability. Some examples follow.

3.5.1 Hybrid Rulebase/Model/Neural Network

Focussing on thruster failure, Deuker, Perrier and Amy [33] use a hybrid Rulebase/model/neural network system. Fault detection is achieved with a movement model to detect discrepancies between modelled and sensed movement. A set of foreseen faults is generated before the mission using the Rulebase, which is then compiled into a neural network. This limits the system to the diagnosis of foreseen faults only but with an increased level of generalisation due to the neural network.

3.5.2 Hybrid Bayesian Belief Network and Neuro-Symbolic

Hornfeld and Frenzel [51] propose using a Bayesian Belief Network in conjuction with incremental neuro-symbolic methods, calling their system INDOS. They hope that "... *the respective advantages of symbolic and connectionist processing can be*

combined while the limitations of each are reduced in their effects" (Hornfeld & Frezel [51]). Fault detection is achieved with conventional methods such as model based discrepancy detection.

3.5.3 Neurofuzzy

Patton, Chen and Lopez-Toribio [80] propose the use of a neurofuzzy model-based 'observer' for detecting incipient (small and slowly developing) faults in non-linear systems. They find that the use of fuzzy logic by itself is not an efficient way to detect incipient faults. Neural networks are much better, but their robustness and sensitivity is difficult to analyse mathematically.

They evaluate their fuzzy observers by applying them to the detection and isolation of intermittent faults in a railway traction system's induction motor. They conclude that *"...the fuzzy observer is an effective tool to generate residual signals for non-linear dynamic system fault diagnosis."* Patton *et al* [80]

Other neurofuzzy approaches include Yen [109] and Patton [79].

3.5.4 Neurofuzzy Models with Bayesian Estimators

Bossley, Brown and Harris [12] propose a system using neurofuzzy models in conjunction with Bayesian estimators. The neurofuzzy models have the advantage of combining expert knowledge with empirical data, with the neural weight being translatable into fuzzy rule confidences without loss of the encoded data. For system identification this means that qualitative expert knowledge and quantitative empirical data can be used in the same system identification cycle. Bayesian estimators are used to train the neurofuzzy models to overcome inadequacies in the data and model construction algorithms.

This system showed good results when tested on a high-quality data set gathered from Florida Atlantic University's Ocean Explorer AUV.

3.5.5 Kalman Filter with Markov Model

Washington, of NASA AMES Research Center [104] has developed a system using a combination of Kalman filters and Markov models.

A Kalman filter is an excellent tool for estimating the state of a process given observations (for a detailed description see Section 3.4.12), but it can only work in a continuous space. If more than one state is required then another approach is needed. Markov models are excellent at estimating which state the system is in from partial observations, as they inherently provide a probabilistic distribution over the set of states.

In this system the discrete states correspond to qualitatively different modes of rover operation, such as 'driving normally', 'stuck wheel', etc. To each of these discrete states is attached a model of operation represented by a Kalman filter.

The system has been tested on data gathered from a Marsokhod planetary rover, giving a promising performance.

3.6 Comparison of Model-Free and Model-Based Diagnosis Methods

Two of the most popular diagnostic paradigms are model-based and model-free. This section compares and contrasts representatives of both approaches, respectively using a conventional Model-Based Diagnostic system and a classic Rulebase.

3.6.1 Rulebase Advantages

Rulebases are extremely easy to develop as they essentially consist of a simple pattern matcher, making the amount of code that must be written relatively small. The rules are easy to encode as they are logical statements, well supported by programming languages. The operating methods are also simple as a classic Rulebase simply scans each rule in turn, highlighting rules that have matched the observed situation.

The simplicity of Rulebases minimises the overhead required of the host computer, and so minimal processor cycles are needed to scan each rule. This means that Rulebased diagnostics is usually far faster than other methods, assuming that the Rulebase is not overly large, and that a matching rule is contained.

As rules are simple logic statements they are easy to modify, and as they are

generally stored in a list (the Knowledge Base) it is easy to add or remove rules as necessary.

3.6.2 Rulebase Disadvantages

Perhaps the biggest single disadvantage of Rulebases is that they can only diagnose foreseen faults - faults that have been specifically foreseen and encoded by the developer. To predict all (or even most) possible faults in a system is a near impossible task, particularly if that system must operate in an unstructured, real-world environment.

Each rule is specifically focussed on a particular fault and situation. For a rule to 'match' the observed situation each parameter must match those specified by the developer. If the same fault was to occur in a different situation, or if parameters differ for some other reason, then the rule will not match even though the foreseen fault is occurring. Rulebases do not inherently have a high level of generality.

Methods do exist to increase the generality of Rulebases, such as Case Based Reasoning (CBR) and the popular Fuzzy Logic approach. Although these do somewhat increase the generality available they reduce several of the advantages, particularly the low overhead and speed of operation.

Rulebases operate on information about a system that is deemed 'shallow knowledge', as the rules simply state what the operation of the system will be under defined, discrete sets of conditions. There is no knowledge about why something happens, but only a relationship between symptoms and failures. For this reason a rule based diagnosis system cannot provide a coherent explanation of the failure, it can only show what rules were fired.

Rulebases also have problems with multiple faults, because "...multiple faults are not easy to isolate ... because when two faults appear, it is not always the case that the union of the symptoms of the two individual faults is equal to the symptoms of the device." (Hansen [47])

3.6.3 Model-Based Advantages

Model-Based diagnostics have the useful ability to diagnose faults that were not foreseen by the developer. This is a key advantage for systems that must operate in unstructured, real-world environments.

A specific fault may also be repeatedly diagnosed even though the observed parameters may differ each time. Because of this Model-Based diagnostic systems inherently have a high level of generality.

Model-Based diagnostic systems are also able to diagnose multiple faults in some cases, as they can perturb more than one system variable during their diagnostic operation.

Indirect sensing is also possible, where the system model is used to predict parameters that are not directly measured. This reduces the need for hardware sensors, so reducing weight and cost.

The system model may be used to generate virtual sensor readings, useful for fault detection, but also useful for system control in the event of sensor failure. Systems exist that switch over from faulty sensors to the 'virtual sensor' provided by the system model.

The system model may also be used for control, particularly constrained control. For instance, a robotic vehicle's mission controller may have to dive at a particular rate whilst keeping within certain attitude constraints. The conventional PID method would mean that the vehicle would start to dive, and limit its rate of dive when the attitude constraint is close to violation. This may mean that the goal could not be accomplished in time. The use of the system model for control means that the mission controller could predict the maximum allowable rate of descent and therefore determine whether a goal was achievable *before* starting the goal.

This ability is even more useful in cases where the robot's operation has been degraded by a fault. Assuming that the system model has been degraded to match the new system operation then the robot's mission controller can use the model to determine if the mission is still achievable, or what level of mission degradation is necessary.

3.6.4 Model-Based Disadvantages

Model-Based diagnostic systems are far harder to develop than Rulebases, although this is mainly due to the difficulty of generating and validating models rather than the development of the diagnostic code. The evaluation code itself consists of a 'perturb and compare' engine that alters modelled system variables to try to match the observed parameters, as discussed in Section 3.4. This code is, in itself, not particularly complicated.

System models, however, can be extremely hard to generate, particularly at high levels of detail and/or accuracy. They can also be difficult to modify, as this can involve as much testing and development time as generating the model in the first place.

The overhead required from the host computer is also far higher than with a Rulebased system. This is mainly due to the large numbers of system model variables that may need to be perturbed during diagnosis. Where more than one variable is perturbed at a time (multiple faults) the number of combinations, and so the number of perturbations, increases dramatically. Each perturbation takes a certain number of processor cycles (and time) to run and at high levels of detail and/or accuracy the process can take an appreciable time to complete.

3.6.5 Summary

Both Model-Free and Model-Based systems have strong advantages and disadvantages which tend to complement each other. The ease of development, low runtime and required foresight of a Rulebase is complemented by the developmental difficulty, high overheads but powerful diagnostic capabilities of Model-Based diagnostic systems.

Havlicsek [50] recognises this and proposes a system called the Westinghouse Diagnsotic System (WEDS). This uses both approaches, combining a rulebase with a Model-Based Diagnosis engine. The rulebase retains human expert diagnostic knowledge, whilst the Model-Based Diagnostic engine extracts diagnostic information directly from design knowledge. He finds that *"Combining these approaches overcomes limitations of the individual techniques and provides a more powerful di-*

agnostic system."

3.7 Integrated Diagnostics

3.7.1 Introduction

Integrated Diagnostics has been around since the early 1980's. It is a process by which large organisations hope to reduce the total life cycle costs of their systems. These organisations have found that a great deal of the total cost of system is attributed to maintenance. In an effort to reduce maintenance cost it was realised that it was vital to design diagnostics into the system from conception, it was felt that this would increase testability, reduce maintenance effort and also increase diagnostic accuracy.

A formal definition of Integrated Diagnostics is given by Dean [31]:

"Integrated Diagnostics is not a technology or a product in itself. Instead, it is a systematic process which integrates and applies technologies and products during system development to achieve a desired result: the most effective diagnostic solution for a given cost."

Integrated Diagnostics has a logistical focus - its prime aim is to reduce the logistics needed to support weapons systems by reducing test time and improving diagnostic accuracy to prevent the replacement of not-faulty components.

It was originally envisaged by the US DoD. The US Armed Forces (and their contractors) have been the prime drivers ever since. Due to the sensitive nature of military and commercial technology the information available is limited, but a representative selection of papers showing how some of these organisations intended to introduce Integrated Diagnostics are: McDonnell Aircraft Company (Ofsthun [78], Bartz & Sallade [4]), Sikorsky Aircraft (Marcus, Mahanna & Gruessner [68]), Unisys Shipboard & Ground Systems Group (Price [86]), Martin Marietta Automation Systems (Brazet [15]), Giordano Associates (Nolan [77]), US Army Aviation Systems Command (Brassel & Burkhart [13]), US Naval Air Warfare Section (Lebron & Rossi [65]), Dassault Aviation (Courtois & Pouilly [26]), General Electric (Brazet [14]), US Air Force (Dean [31]), Bendix Guidance & Control Systems (McCown [71]),

Lockheed (Gaffney & Morones [41]).

All of the above papers stay at the conceptual level due to the sensitivity of the technology.

3.7.2 The US Navy's Integrated Diagnostics Support System (IDSS)

The US Navy was already moving towards the concept of integration with its AEGIS weapons system's Operational Readiness & Test System (AEGIS ORTS) - this has since come to be seen as a precursor to an Integrated Diagnostic System (Brazet [15]. Since then the US Navy has embraced Integrated Diagnostics, developing its own Integrated Diagnostics Support System, or IDSS, an approach based upon using a common set of tools for weapon systems design (Cigler [25] and Bearse & Carrier [5]). These tools are centred around dependency models. It has been a successful approach that has since been transferred to other domains, such as Iridium Telecommunication Satellites (Bearse, Dill & Lynch [6]), and automated test equipment (Franco [38]).

3.7.3 QSI's TEAMS Toolset

A similar approach to the US Navy's IDSS has been taken by Qualtech Systems Inc, who produce a commercial set of software tools for Integrated Diagnostics system monitoring and maintenance, an overview of which is given in Deb, Pattipati & Shrestha [32]. The toolset, which is model based, is known as TEAMS, and features both human-assistance and real-time diagnostic capability. The TEAMS modelling methodology, which they term multi-signal modelling, models the system in its failure space. All potential faults are identified and their symptoms identified, the cause-effect dependencies between faults, symptoms and test events are then modelled (Mathur, Deb & Pattipati [69]. Proprietary algorithms are then used to work from observed symptoms back to the fault. These algorithms may be used to help a human repair technician, a real-time version is also available for use on embedded systems.

The drawback of the TEAMS approach is that all faults must be anticipated -

the system cannot diagnose faults that are not in its knowledge base. However, it has been taken up by Sikorsky (Mathur *et al* [70]) and the US Naval Air Warfare Centre (Ghoshal *et al* [43]), and is currently being expanded into a 'virtual test bench for life cycle support', which focusses on the design modelling and analysis stage (Cavanaugh [22]). The success of TEAMS shows that there is a definite need for Integrated Diagnostic tools in the real world.

3.7.4 US Navy Helicopter Integrated Diagnostic System (HIDS)

The Helicopter Integrated Diagnostic System (HIDS) is a state of the art US Navy program which focusses on fault detection, diagnosis and prognosis of helicopter drive-train systems. The HIDS program aims to evaluate different diagnostic systems, integrate them into a flight worthy system and act as a showcase Integrated Diagnostics system in anticipation of a fleet-wide implementation (Hardman, Hess & Scheaffer [48]). The focus on mechanical drive-train faults means that HIDS is currently centred on vibrational analysis techniques.

The HIDS team have located a number of vibrational sensors (accelerometers) at suitable places on the test helicopter drive-train, with each sensor's position being optimised for detecting the vibrational signature of a particular mechanical component. All sensor data is analysed and recorded for post-mission processing.

The method used for fault detection is Hotelling's T2 Multivariate Statistical Analysis, which combines various parameters into a single indicator of system health. This method has a well proven ability to reduce false alarms (Mimnagh, Hardman & Scheaffer [72]).

The HIDS team has also evaluated the performance of various diagnostic systems such as classical Model-Based Diagnostics, Neural Networks, Fuzzy-Logic and other helicopter-specific techniques such as oil-filter debris monitoring systems . No information is yet available on the results of these evaluations. The next step is to develop and validate advanced model-based analysis, data fusion and other techniques. They are also planning to add a 'Reasoner' to various levels of the diagnostic process (Hardman, Hess & Scheaffer [49]).

The HIDS program shows that Integrated Diagnostics is highly relevant to both

logistical operations and in-flight (or embedded) diagnostics. It is unfortunate that little information is available on the underlying HIDS integration technology.

3.7.5 Misnomers

Some systems that claim to be Integrated Diagnostics are actually hybrid systems, where different diagnostic systems have been 'integrated' together (such as Cavallini *et al* [21]), or integrated into an area where previously there were none (Motorola have now integrated diagnostics into their embedded memory devices (Hunter [56])), but this is not a proper use of the term.

3.8 Diagnostic Fusion

3.8.1 Introduction

A relatively new diagnostic field is Diagnostic Fusion, which aims to overcome the limitations of using a single diagnostic tool by fusing together different types. In this way the weaknesses of a Rulebase may be augmented by the strengths of, say, a neural network to provide a powerful diagnostic system.

Fused diagnostic systems have been around for some time in the form of Hybrids (see Section 3.5), but there is now a push to develop underlying methods to allow the fusing of many types of diagnostic system, rather than simply generating an individual application-specific hybrid system.

Diagnostic Information Fusion is defined as *"... the method by which one would determine a system's state for those instances where several different diagnostic tools, and possible other sources, are used for state estimation."* Garbiras & Goebel [42]

Goebel, Krok & Sutherland [44] highlight some of the key problems with Diagnostic Fusion:

- *"... when information is expressed in different design domains, such as probabilistic information, ... binary information or weights, the fusion scheme needs to map the different domains into a common one to be able to properly use the encoded data."*

- *"The fusion scheme also has to deal with [diagnostic] tools that operate at different sampling frequencies."*

- *"...if [diagnostic] tools disagree, one has to decide which tool to believe and to what degree."*

They also state a key benefit: *"No one [diagnostic] tool is required to deal with all faults at a high level of accuracy because no one method can do so."* Goebel [44].

Garbiras and Goebel [42] have shown that Integrated Diagnostics can improve the diagnostic performance over that of individual diagnostic tools .

To date there have been only a few attempts at presenting a unified framework for Integrated Diagnostics, most of which concentrate on military weapons programmes. They are not focussed on the needs of autonomous robots.

3.8.2 General Electric's IMATE System

Goebel, Krok & Sutherland of the General Electric Corporation have recently presented a system called IMATE (Intelligent Maintenance Advisor for Turbine Engines) that aims to provide a framework for diagnostic fusion. It concentrates on providing a real-time solution aimed at fusing the diagnostic outputs of different types of diagnostic systems, concentrating on the requirements flowdown and interface issues. They do not consider an offline, post-processing version (Goebel [44]).

In order to fuse differing diagnostic tools they place a constraint on the tool providers, ensuring that each tool provides a confidence level for each individual fault rather than a crisp 0/1 value. They do not detail their method of integrating diagnostic tools that operate at different sampling frequencies.

They evaluate their system on an aircraft gas turbine engine using three discrete diagnostic tools fused together with an algorithmic 'Intelligent Fusion Module', the details of which are not revealed, presumably for commercial reasons. They also use secondary diagnostic information, which may be used to support but not discount a diagnostic finding. This information consisted of outputs from various engine deterioration models, vibration analysis, thermal history and other fault related information. Effectively, this provides a degree of heterogeneous knowledge integration.

At the time of writing the system had only been tested in simulation, which gave promising results. In particular, they *"...achieved a more accurate diagnostic estimate of system health than the individual estimates provided by a heterogeneous collection of diagnostic tools"* (Goebel [44]).

3.8.3 Neural Fusion

An alternative, more limited approach is proposed by Garbiras & Goebel [42], in which a neural network is used to fuse the outputs of different diagnostic tools. They focus on providing a system to recognise faults without *a priori* knowledge of the system.

To evaluate their proposal they use a two tier strategy. The lower tier consists of four diagnostic tools: a Case Based Reasoning (Nearest Neighbour) tool, two neural networks looking at different data aspects and a fuzzy inference engine. The upper tier is a neural network, four types of which are evaluated.

They find that *"The use of fusion neural networks provides a means to improve performance of individual diagnostic tools."* (Goebel [42])

3.8.4 The U.S. Army's Prognostic Framework

The U.S. Army is developing a 'Prognostic Framework' (Su & Nolan [96], [95]) aimed at integrating logistical infrastructure with embedded diagnostics, so providing total health management for its weapons systems. The foundations of this framework are hierarchical modelling and the separation of test and diagnostic functions.

The core of the prognostic framework is a 'design based model', called the Fault Propagation Model, consisting of relationships between faults and symptoms. This model is essentially a two dimensional matrix that maps information from raw sensor data, embedded diagnostic tools, pilot debriefing, etc to known faults. A set of 'intelligent' algorithms, collectively known as the Diagnostician, then operate on this matrix to isolate faults from given symptoms. The model maps sensor data to physical components rather than the more conventional sensor to system mapping. Using this model they are able to extend built-in test information to diagnosis. The hierarchical model can be dynamically reconfigured to reflect changes in reconfig-

urable systems.

This model is currently being augmented with secondary information, such as historical data, trend analysis, failure modes etc. Although the present Diagnostician is essentially static in operation (it can only work with 'snapshots' of the system) it is being upgraded to dynamic operation by the inclusion of a 'wear/degradation' dimension in the Fault Propagation Model.

They highlight the usefulness of such a system for autonomous vehicles; *"An autonomous real-time health monitoring system which provides automatic fault detection, isolation, reconfiguration and reporting increases the chance for these [autonomous] missions to be successful"* (Su *et al* [96]).

The system is still under development and so results are not available at the time of writing.

3.9 Blurring the Line: The Open Systems Approach to Integrated Diagnostics Demonstration (OSAIDD) Study

3.9.1 Introduction

Recently the line between Integrated Diagnostics and Diagnostic Information Fusion has become blurred. It is becoming clear that Integrated Diagnostics needs to be more than a process. To realise its full potential it must become a technology. Diagnostic Information Fusion, the methods that must be developed to integrate heterogeneous diagnostic systems, is one of the key technologies necessary.

Because of this the United States Department of Defense recently conducted a study on the approaches taken towards integrated diagnostics (see the study report [40]). Their aims were to reduce total ownership cost, shorten acquisition cycle time for technology insertion and increase the interoperability of their in-field systems. The study was aimed at large organisations, such as the military, with the focus on integrating embedded diagnostics within the organisation's information structure and logistical framework. Ten case studies of existing Integrated Diagnostic frame-

works were performed, covering the three main areas of interest; defense, non-defense government and commercial.

One of the key findings from the OSAIDD study was that: *"...a consistent approach to integrating diagnostic functions did not exist, was feasible and should be advanced as a core tactic in achieving the DoD's [United States Department of Defense] strategic goals."* (Freschi [39])

The study recommended the use of: *"...an information-based, open systems approach to defining and integrating diagnostic functions within the components of a generic architecture of hardware and software elements"* (Freschi [39]). Their proposed architecture is shown in Figure 3.1.

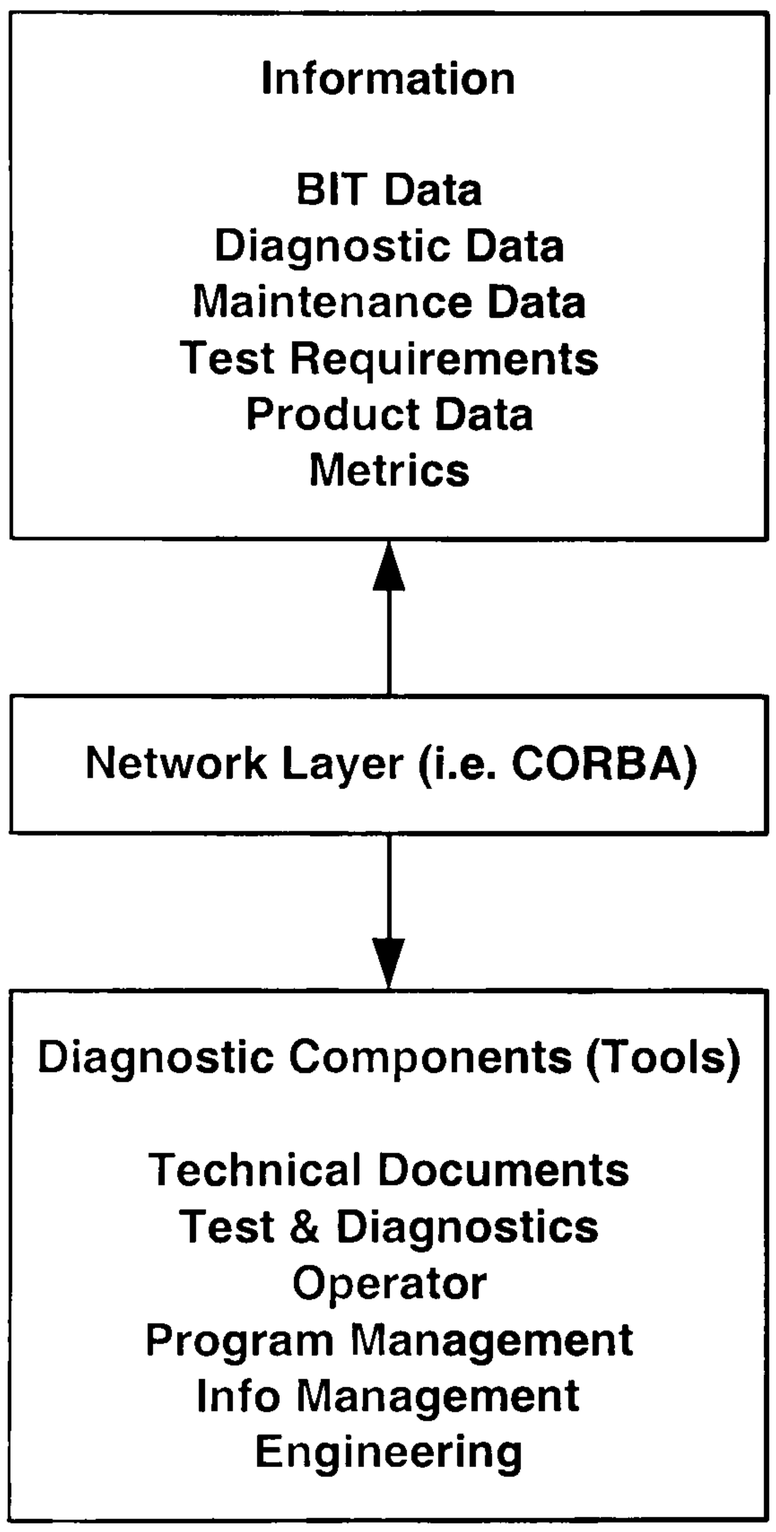


Figure 3.1: The Proposed Architecture from the Open Systems Approach to Integrated Diagnostics Demonstration

The study states that: *"The basic premise of the recommended technical approach is the concept of a formal model of diagnostic information, which is shared by all participants in the system test and diagnosis process. The mechanism for this approach is an information model, which is a rigourous, formal specification of the*

information used within the system test and diagnostic domain. Optimising this process comes from sharing this information throughout the diagnostic process."(Freschi [39])

The OSAIDD study is the most thorough and far reaching analysis of Integrated Diagnosis performed so far.

3.9.2 Critique of OSAIDD Proposal

The OSAIDD study focussed on the integration of embedded diagnostics within the organisation's logistics system. This is presumably due to the study being sponsored by the United States Department of Defense, who are interested in reducing maintenance costs for their huge array of weapons systems. There is little attention paid to the needs of fully autonomous vehicles apart from a recognition that they could also benefit from Integrated Diagnostics, with the on/off-product diagnostic partition moved towards on-product.

The requirements of an autonomous vehicle operating far from human help are very different to that of a weapons system or civilian airliner operating within a human-orientated logistical infrastructure. Within a logistic-orientated environment most of the diagnostic framework may be run off-vehicle and offline but an autonomous vehicle requires everything to be run on-vehicle and online.

Architecture:

The proposed architecture is highly conceptual, without a detailed specification for any of the architectural components. There are no real-time aspects, essential for on-vehicle operation, or any consideration of reducing processor operations in order to conserve power. These are fundamental requirements for an on-vehicle integrated diagnostic system.

There is no provision for selection of diagnostic tool, which can depend heavily on the situation, as detailed in Goebel, Krok and Sutherland [44]. There is also no way to influence the selection of information for analysis, which is vital for a time-constrained system.

Knowledge:

The OSAIDD proposal defines key types of information for Integrated Diagnostics, cited in Section 3.9, but a detailed analysis of the information is not provided. Some crucial types of information are missing. For a thorough analysis of available information see Chapter 4.

Information Model:

The mechanism for the recommended Information Model, as shown in Figure 3.2 is very vague.

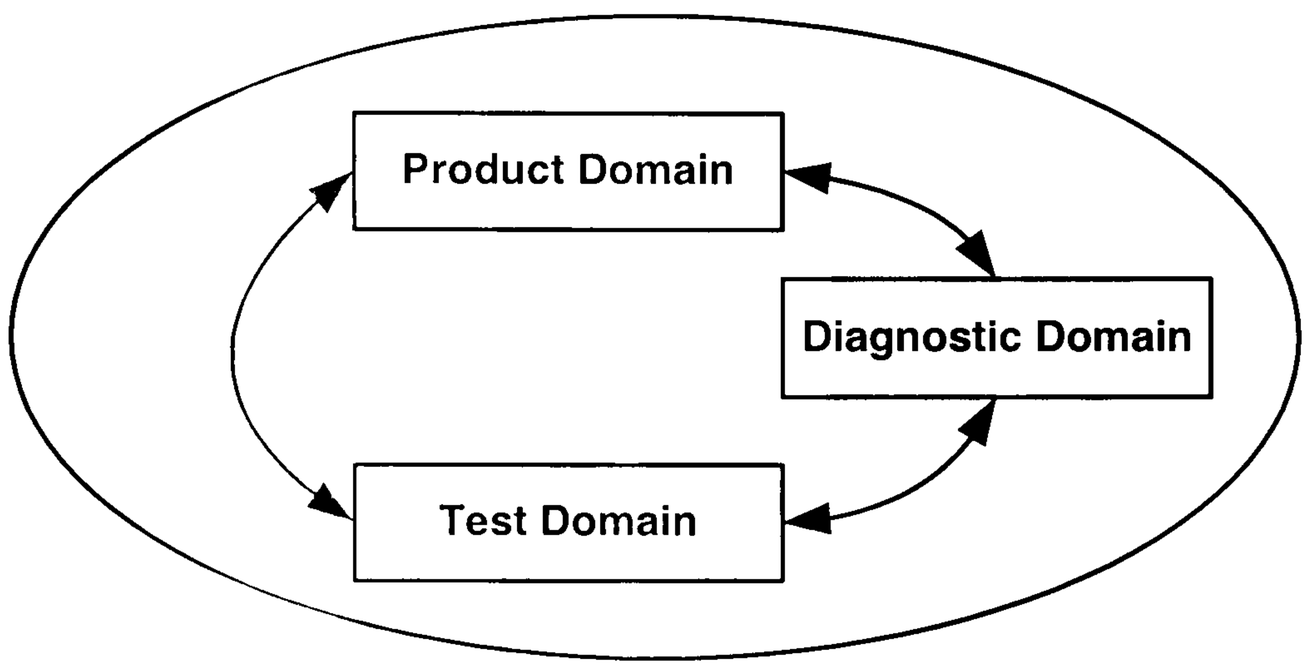


Figure 3.2: The Proposed Information Model from the Open Systems Approach to Integrated Diagnostics Demonstration

The study states the need to integrate information from three domains: product, test and diagnostic, but goes no further. There is no breakdown of the information in these domains, or whether the information will be partitioned by physical component, function or system.

Interface:

The interface between diagnostic tools is based on the internationally and commonly used Open Systems Interconnection 7 layer network model, as shown in Figure 3.3.

This proposed interface concentrates on the passing of information via different physical and software technologies and, as such, is a good choice for Integrated Diagnostics.

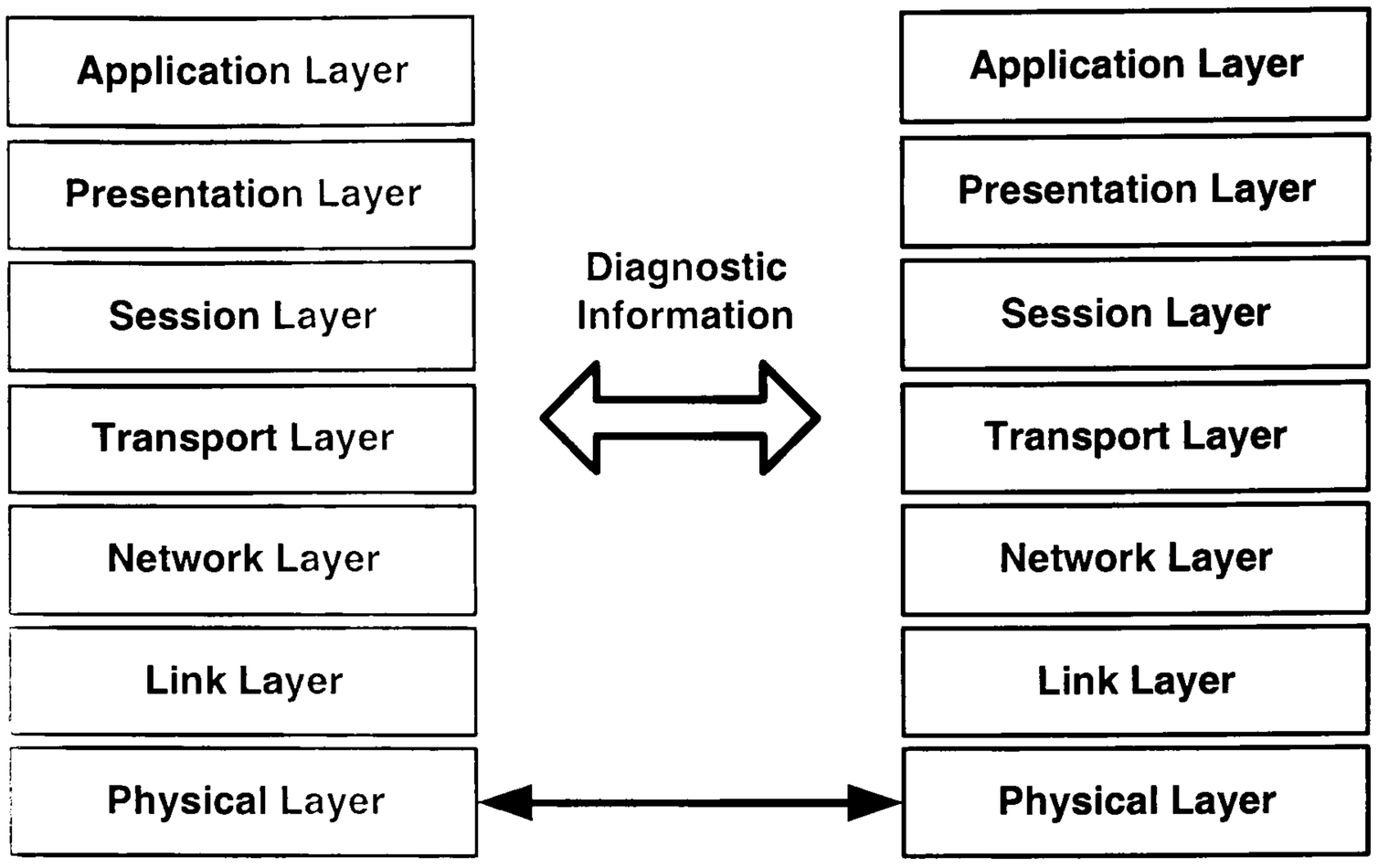


Figure 3.3: The Proposed Information Interface from the Open Systems Approach to Integrated Diagnostics Study

There is no detail on how to actually interface the different types of information produced by different diagnostic paradigms. For instance, a conventional Model Based Diagnosis system simply nominates components with no probability attached, whereas probabilistic methods provide a nomination and probability.

Summary:

The OSAIDD proposal is conceptually thorough for logistical-based integration of diagnostics but misses out crucial aspects for autonomous robotic vehicle implementation. No details of any part of the architecture are provided apart from critical classes of information, which are again vague.

3.10 Chapter Summary

This chapter first reviewed the two main types of individual diagnostic systems, model-based and model-free, discussing and giving examples where appropriate. Some representative examples of hybrid diagnostic systems were then given.

The focus then moved to Integrated Diagnostics, a field largely supported by the US military about which there is little technical information due to the sensitive nature of the technology. The related area of Diagnostic Information Fusion was

then discussed and its relevance to this research shown. The OSAIDD study was shown to recognise that Integrated Diagnostics must move from being a process to a technology. This is particularly crucial for autonomous robotic vehicles.

The chapter showed that there are many types of individual diagnostic systems but few attempts to provide an underlying architecture for integrating them together. The strengths and weaknesses of the existing attempts, particularly the OSAIDD study, were detailed. The OSAIDD study was shown to be a thorough analysis of requirements but was focussed on logistical, not vehicular, integration. The proposed concept was shown to be vague.

The next chapter introduces the original architecture designed, implemented and evaluated during the course of this research. It is shown to be a more advanced, detailed and working architecture that is conceptually similar to the OSAIDD proposal.

Chapter 4

RECOVERY Concept

4.1 Aim of this Chapter

From this point on the thesis details original contributions from this research.

This chapter describes original contributions from the RECOVERY system and explains the underlying concepts. Differences from the OSAIDD study, a major study on Integrated Diagnostics, are highlighted and justified.

4.2 Introduction to RECOVERY

The RECOVERY architecture and system developed during the course of this research is more detailed, advanced and wider ranging than the OSAIDD study detailed in Chapter 3. It is a working system, fully focussed and implemented on a real autonomous vehicle. It works on real-world missions with real data. Many of the OSAIDD recommendations (published mid-way through the research period) are expanded to a much more detailed level, particularly the highly critical Central Information Model.

RECOVERY is designed to work as a standalone module in conjunction with an enveloping mission controller, as shown in Figure 4.1. This concept has been successfully evaluated on NASA's Deep Space 1 probe. The mission controller is preferably a goal-orientated planner but could also be a common Finite State Machine controller or other type of mission controller.

RECOVERY monitors the vehicle's sensor suite, using heterogeneous (different types of) knowledge in conjunction with heterogeneous diagnostic tools to detect and diagnose faults. Once the fault is diagnosed the relevant fault information is passed to the vehicle's mission controller for enhanced fault recovery/accommodation.

RECOVERY is designed to be able to use commonly available design information

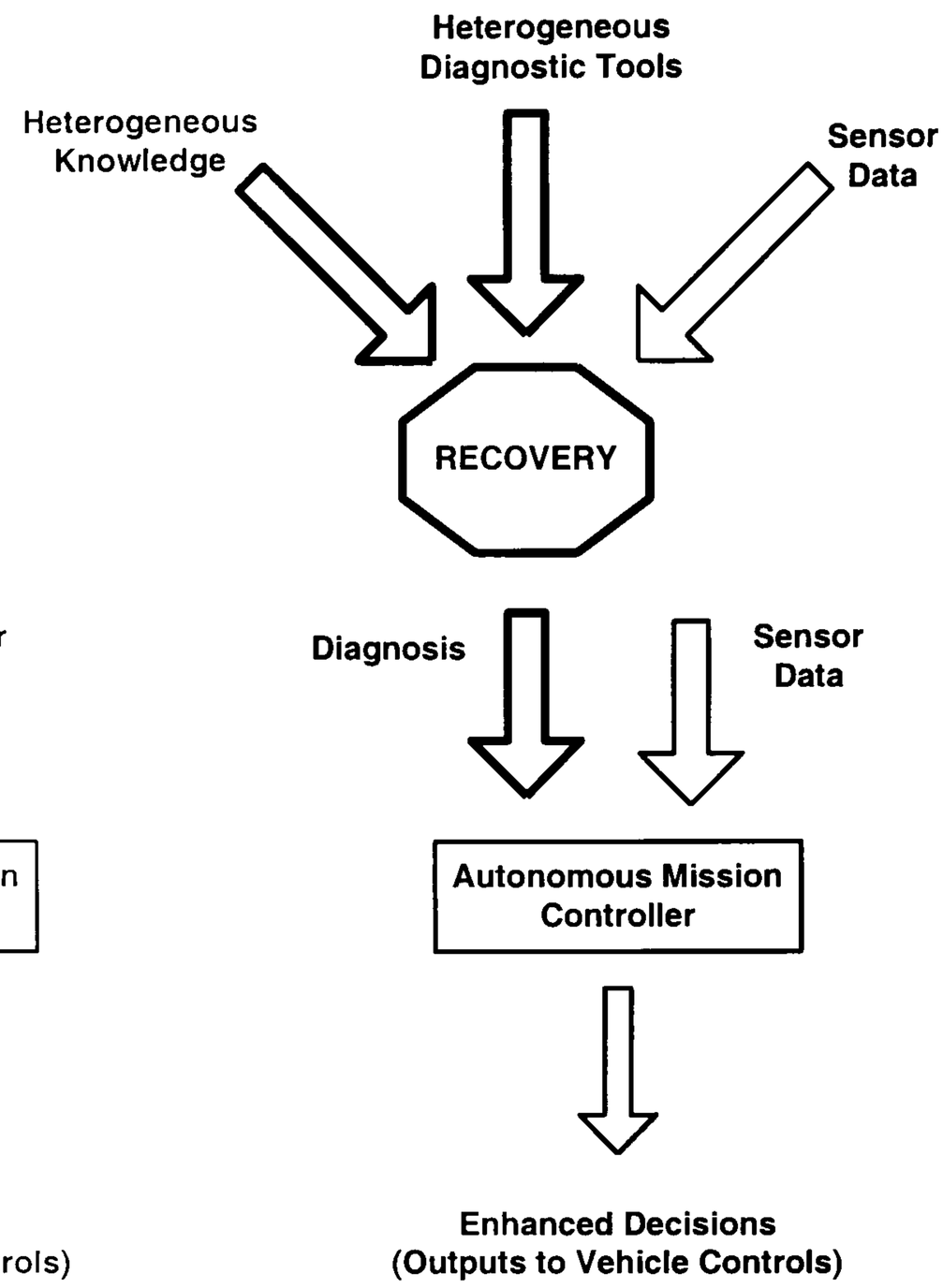


Figure 4.1: RECOVERY concept showing (a) conventional mission controller using only sensor data and (b) a mission controller enhanced with diagnostic information

such as hydrodynamic models, circuit diagrams, schematic diagrams and expert human knowledge. Used together this information can greatly enhance diagnostic capability.

4.3 Original Aspects of RECOVERY

There are seven major original contributions from this research, together with supporting work. These contributions are:

- A novel, information based architecture for Integrated Diagnostics, capable of integrating heterogeneous diagnostic knowledge and tools

- A detailed, working Dynamic Relational Construction Model (Relational Model for short), which fulfills and expands the concept of a Central Information

Model as recommended by the OSAIDD study

- Methods for storing, searching and operating on the Relational Model
- A method for reducing the search space of a multidimensional, dynamic Integrated Diagnostic system using a Focus of Suspicion approach
- A novel approach to Domain Independent Diagnostics using the Relational Model. The use of Domain Independent Diagnostics alongside conventional diagnostic paradigms is evaluated. Advantages and disadvantages of Domain-Independent Diagnostics are shown
- The implementation and evaluation of the above on a real autonomous vehicle using real missions in real water
- Supporting work is given on the types of knowledge available on an autonomous robotic vehicle for aiding diagnosis. A breakdown of types of components and the relationships between them is proposed

To the author's best knowledge, at the time of writing this is the first time that Integrated Diagnostics have been specifically focused, implemented and evaluated on an autonomous robot.

4.4 Architecture

In concept the RECOVERY architecture consists of three parts: plugged-in diagnostic tools, plugged-in diagnostic knowledge and a central Relational Model, as shown in Figure 4.2. The RECOVERY architecture aims to provide an information-based, modular architecture suitable for integrating various types of diagnostic knowledge and diagnostic tools, including model based, model free, static, dynamic and multi-dimensional approaches.

RECOVERY provides common ground for the various types of diagnostic methods and knowledge using the central Relational Model, which is the focus of this research. The Relational Model acts as a map of the vehicle, providing a common

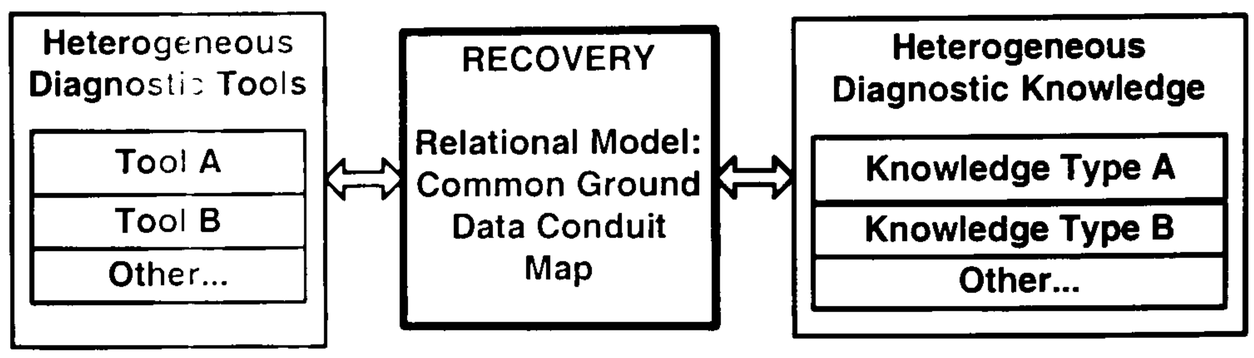


Figure 4.2: RECOVERY architecture concept showing that it provides common ground between different diagnostic tool inputs (knowledge) and outputs (diagnoses)

set of components for the diagnostic tools to diagnose. It also acts as a data conduit for passing the correct type of diagnostic information to the relevant diagnostic tool.

Figure 4.3 shows a more detailed conceptual view of the RECOVERY architecture. RECOVERY's Relational Model provides a structure upon which the novel Domain-Independent Diagnostics (developed during the course of this research)can operate. These are effectively a second set of diagnostic tools that may be plugged-in at will and so they are shown outside the RECOVERY architecture in Figure 4.3.

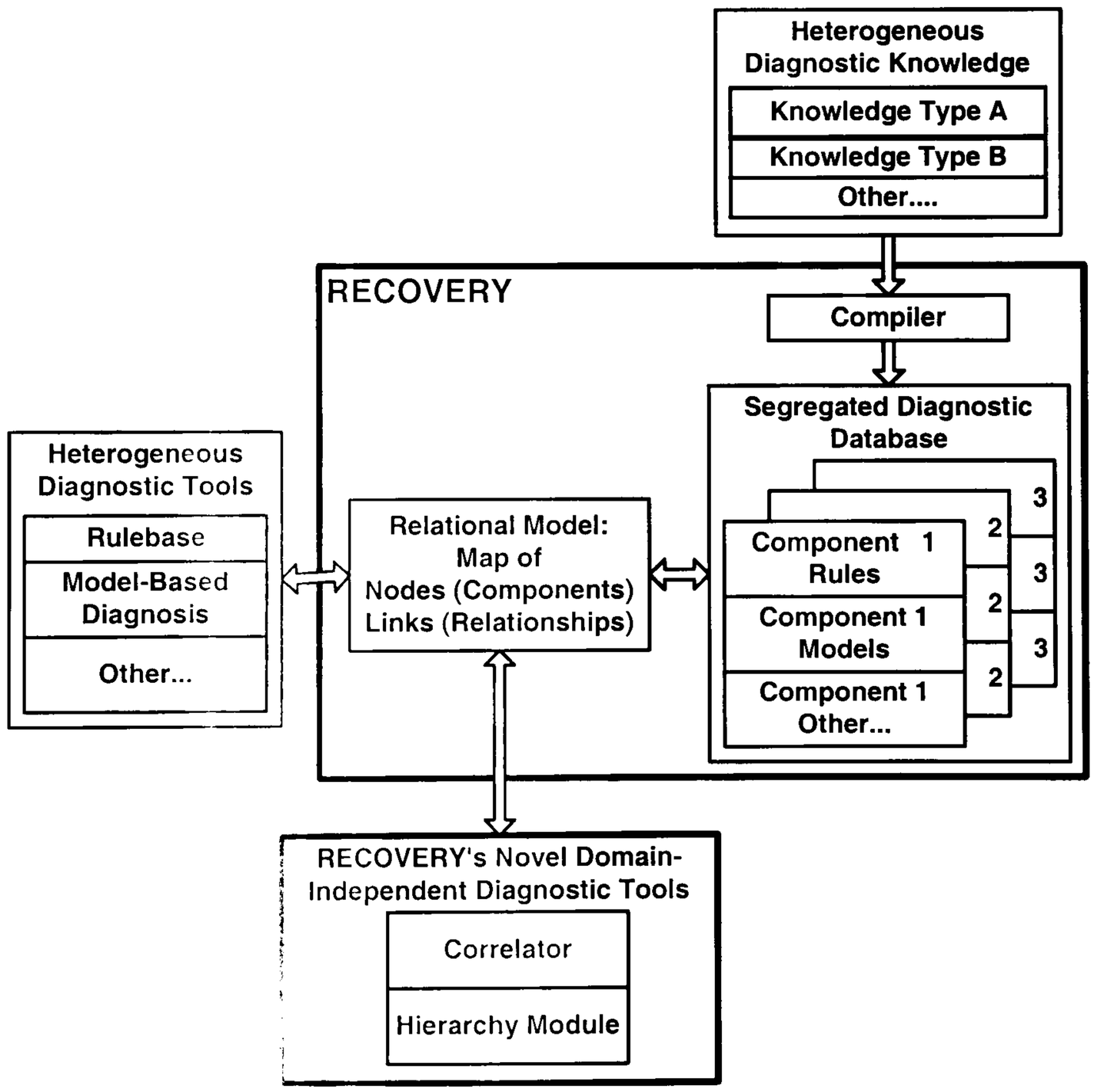


Figure 4.3: Detailed RECOVERY architecture concept showing the compiler segregated database, component map and novel domain-independent diagnostic tools. Blocks outlined in bold are original contributions.

RECOVERY contains its own compiler that can take Component Information Files (containing specific diagnostic information on a particular component) and construct a heterogeneous database. This database contains heterogeneous knowledge, such as rules and models. Crucially, this database is segregated by component to allow the use of RECOVERY's novel search-space reduction method.

For completeness the full RECOVERY architecture is shown in Figure 4.4. A fully detailed description of the architecture and implementation follows in Chapter 5.

4.4.1 Modularity

RECOVERY uses a centralised approach to modularity. The RECOVERY kernel will accept information files containing descriptions of the various vehicle components, their relationships to other components and specific fault knowledge of each component. In this way, the knowledge file represents each significant vehicle component. These knowledge files are similar to the Component Information Files used in Windows NT and Plug and Play systems. In a similar way, diagnostic tools may be plugged in to the architecture.

4.5 Heterogeneous Diagnostic Tools

There are many different types of diagnostic tool available for use in an Integrated Diagnostics system, ranging from simple Rulebases to powerful probabilistic techniques, as detailed in Chapter 3. Different types of tools have different input and output requirements, leading to the key questions:

- How can the outputs of different tools be combined?
- How can the correct information be passed to a tool?

These questions are directly addressed by the use of RECOVERY's novel Relational Model, detailed later in this chapter.

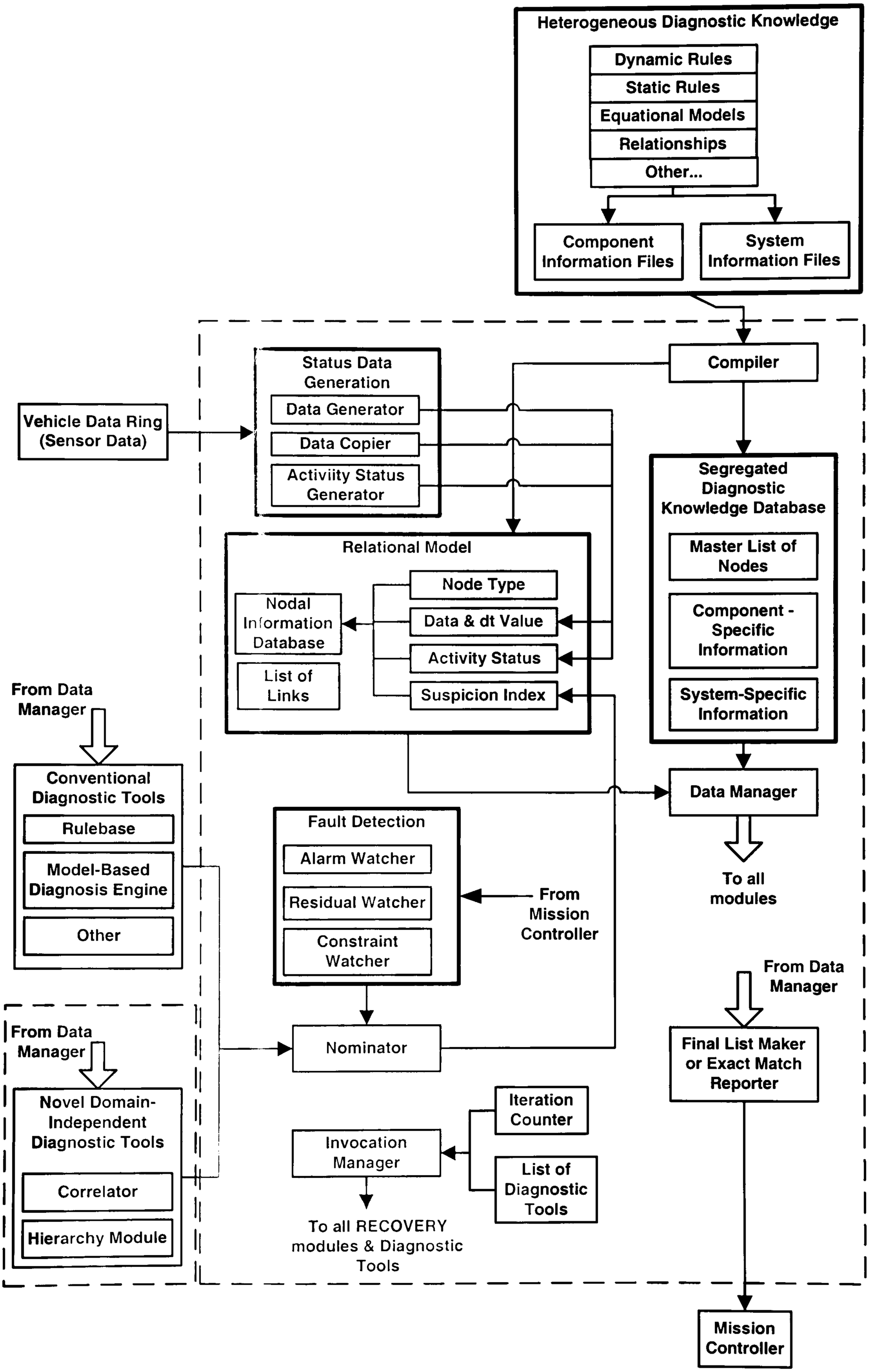


Figure 4.4: RECOVERY Architecture

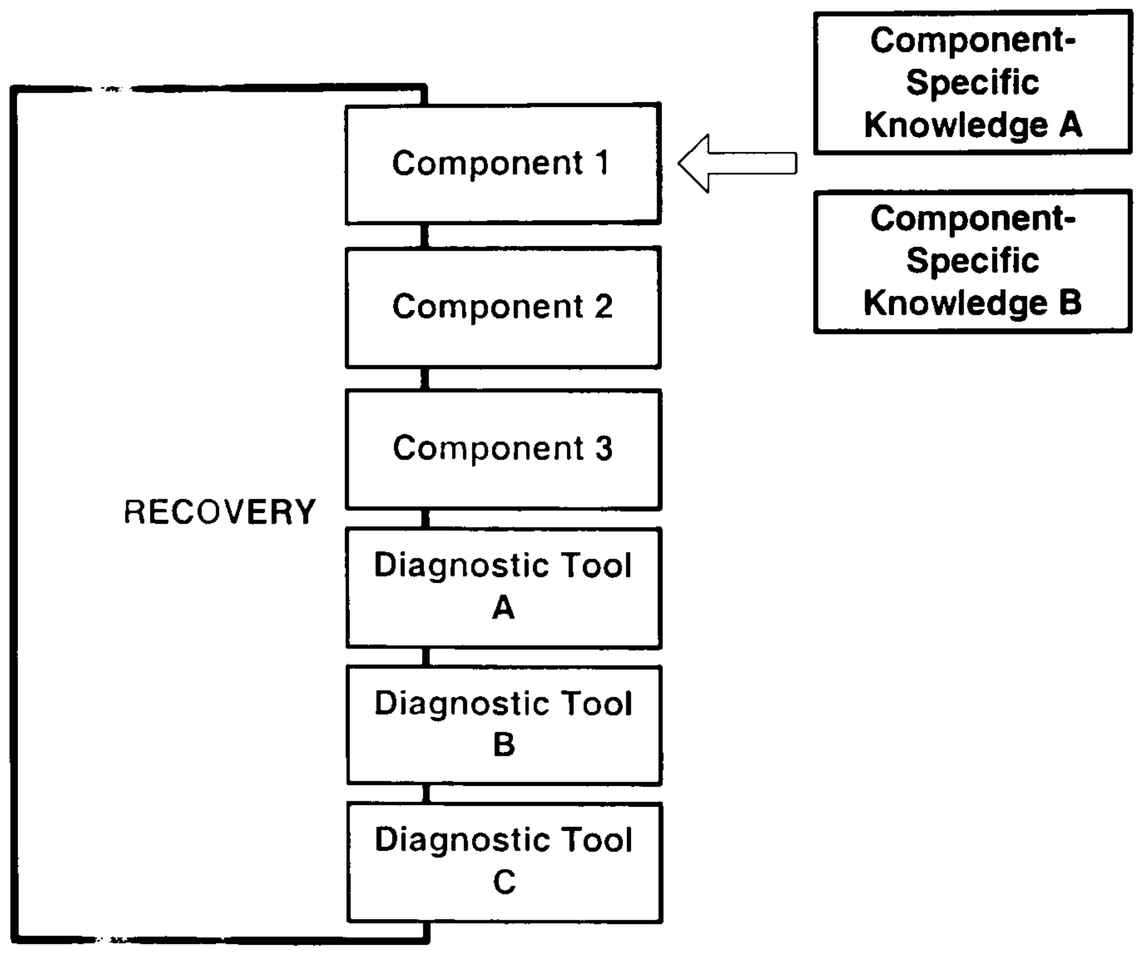


Figure 4.5: RECOVERY Modularity Concept

4.6 Heterogeneous Diagnostic Knowledge

Many different types of knowledge are available for aiding diagnosis. Types of information vary from simple rules to first-principle models, or large data sets suitable for training Neural Networks. This section details the types of knowledge available for an autonomous robotic vehicle and its suitability for diagnosis.

The OSAIDD study proposes seven types of knowledge that are critical for diagnosis:

- Product Data: Component serial number, place of manufacture, etc.
- Test Requirements: How a system must be tested.
- Diagnostic Data: Information pertaining to a fault.
- Built-In Test Data: Health status data provided by individual components.
- Sensor Data: Readings from sensors, such as depth from a depth sensor.
- Maintenance Data: When system maintenance was performed and what was done.
- Metrics: The metrics used to test and evaluate a system.

The OSAIDD study ignores some key classes of information as given in Hamilton *et al* [46]:

- Design Knowledge: All the information that is generated during the design process.
- Domain-Independent Knowledge: Generic diagnostic techniques.
- Sensor Knowledge: Not just sensor readings, but information about the sensor itself.
- Mission Knowledge: What the vehicle was trying to achieve when the fault occurred.
- Fault Knowledge: Information on faults that have occurred before, mean time between failure information, etc.
- Historical Knowledge: What happened in the period leading up to the fault.

Design Knowledge: 'I am a vehicle'

One of the most vital types of knowledge for in-vehicle fault diagnosis is the design knowledge, or the knowledge that tells the vehicle that it is a vehicle. Research into this area is ongoing but at present it consists mainly of relationships between components, parameters and sensors.

A great deal of vehicle design knowledge is often available, much of it generated as part of the vehicle design process. For instance, circuit diagrams, mechanical drawings, data flow diagrams, system schematics, fault log-books, software class diagrams are all useful. This information is usually stored until needed by personnel trying to diagnose a problem. Other knowledge, such as sensor data, is generated whilst the vehicle is performing its mission.

There is also a growing movement towards Failure Modes and Effects Analysis (FMEA) to be incorporated into the design cycle. If this is realised then a great deal of extra knowledge in the form of failure models will become available for diagnosis. Work on automating the generation of FMEA knowledge is detailed by Price [84].

Domain-Independent Knowledge

A useful form of knowledge is generalised fault diagnosis knowledge, such as the procedural knowledge that engineers use to track down faults. For instance, if the brakes are applied on your car and there is a loud bang followed by a great deal of shuddering, a fair assumption is that there is a brake problem. This is because the braking system was activated just before an overall system fault (with the car) became apparent. This sort of general knowledge can be applied across many types of systems, not just motorcar brakes. Of course, the problem may lie elsewhere but there is a good chance that the problem is with the brakes.

Sensor Knowledge: 'What's going on?'

There are generally two types of sensor present on an autonomous robotic vehicle: systemic and environmental. Systemic sensors, such as thermal or power consumption sensors, sense the state of the vehicle. Environmental sensors, such as magnetometers or sonars, sense the state of the environment. All these sensors produce information on what is currently happening in their particular domain but information is also available on the sensors themselves.

Built-In Test Data

An extension to systemic sensor data is Built-in Test Data, which is the data generated by a system's self-test procedures. For instance, the dc motor controllers used on the evaluation platform continuously monitor themselves for faults such as an overvoltage, thermal trip or broken motor connection. This is extremely useful information for vehicle diagnosis, but the motor controllers represent this information by varying the flash rate of a LED. Other components use different methods, making their input to an Integrated Diagnostics system difficult.

Mission Knowledge: 'What am I supposed to do?'

A major part of the knowledge available to an autonomous robotic vehicle concerns its mission. Providing information to the diagnostic system on what the vehicle was doing at the time of the fault, or which goal constraints have been violated, can aid

diagnosis. Known environmental conditions pertaining to the mission may also be of use, such as high levels of water current or the presence of hydrothermal vents. MetOcean data can also be of use.

Fault Knowledge: 'What's wrong?'

Some types of information will pertain directly to faults, such as models of known faults, logs of faults that have already happened, mean time between failure of particular components, or the likelihood of failure of a specific component. This information is extremely valuable as it can relate directly to a fault. If used in conjunction with other information it can increase the confidence of a particular diagnosis.

Historical Knowledge: 'What's happened?'

Much of the above information will be logged during vehicle operation. Other information can be provided to the vehicle before the mission, such as known faults of certain components, the length of time for which a component has been used, likely time of failure, etc.

4.6.1 Extracting Knowledge

An ongoing research area is that of extracting knowledge. The technique of extracting expert knowledge from humans is known as knowledge engineering (Russell & Norvig [90]), this is a difficult area as human knowledge is often subjective, with different experts expressing the same knowledge in different ways, or using different methods to arrive at the same conclusions.

A further research area is that of integrating and extracting knowledge from heterogeneous databases, as covered by Dao & Perry [29] and Buchner *et al* [17].

4.7 Relational Model

It is clear that many different types of knowledge exist, or can be usefully added, on an autonomous robotic vehicle. In order for this heterogeneous knowledge to be used

for fault diagnosis, some way must be found to link it together and operate upon it. In particular, the selected methodology should support modularity, machine learning and the ability to use generalised knowledge to aid diagnosis.

There are also many different types of diagnostic tool, with different input and output information requirements. Common ground must be provided in order to combine the outputs of these tools, and to allow the selection of the correct input information from the heterogeneous knowledge database.

In RECOVERY this facility is provided by the Relational Model, which is equivalent to the Central Information Model proposed in the OSAIDD study. The Relational Model specifies diagnosable components and their relationships (the topology of the system), in this way it provides domain-independent diagnostics and the ability to diagnose components that are not directly sensed. The Relational Model is shown in Figure 4.6.

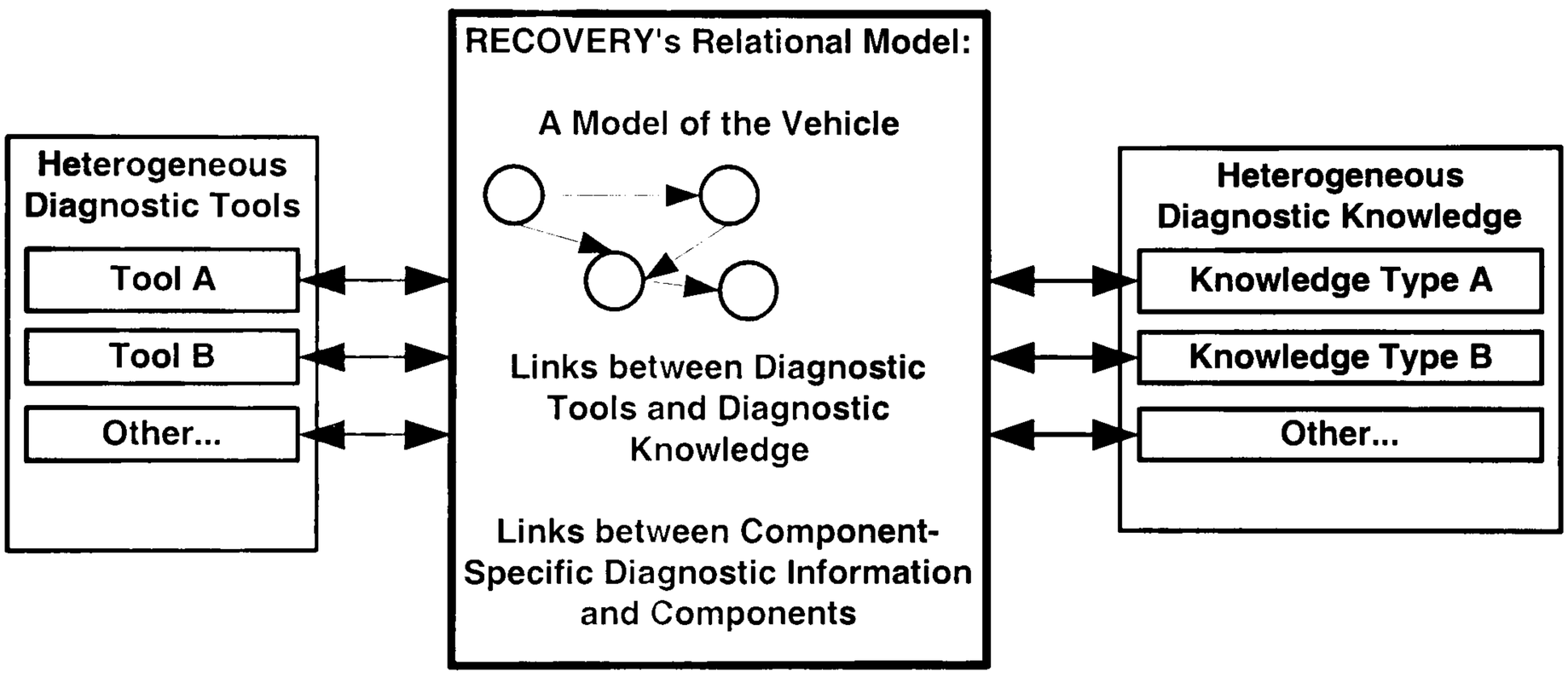


Figure 4.6: The Relational Model Concept

4.7.1 Purpose

The purpose of the Relational Model is:

- To model the construction of the vehicle by specifying relationships between different types of components and parameters
- To represent the current state of the vehicle as completely as possible

- To provide an interface between heterogeneous diagnostic tools
- To tie together heterogeneous diagnostic tools and knowledge
- To link component-specific knowledge to specific components
- To provide methods for reducing the diagnostic search space and hence minimise the amount of time and number of processor operations required for diagnosis
- To provide a structure upon which Domain-Independent Diagnostics may operate

The various functions of the Relational Model are detailed in the following sections.

4.7.2 Modelling the Construction of the Vehicle

Providing a model of the construction of the vehicle that contains all diagnosable components ensures that all the diagnostic tools are working in the same diagnostic space. Further, by modelling the topology of the vehicle it is possible to run analysis algorithms to enhance fault diagnosis to the point where faulty components may be diagnosed without explicit sensing.

What is the Relational Model based on?

Partitioned semantic networks, modified to reflect the change from natural language to robotic diagnosis applications, have been chosen as the underlying methodology in the RECOVERY system. Each component and vehicle sub-system is represented by a complete, heterogeneous set of knowledge tied together using an individual semantic network. These components and subsystem networks are then joined together to form the complete vehicle model.

Why Use a Semantic Network?

A semantic network provides the necessary flexibility to cope with such varied host platforms as autonomous vehicles. They trade expressiveness for modularity and

ease of understanding. As a robotic vehicle is a well defined, manufactured system with components operating in a limited context this trade-off is acceptable. Semantic networks are also efficient, as inference is performed by simply following links very few processor cycles are required (Russell [90]).

Ng and Chow [76] have also achieved some success by representing circuit diagrams with semantic networks and operating on them with a structural analyzer to infer topological relationships between network nodes.

What Does the Relational Model Contain?

The Relational Model contains nodes, which have various attributes, and relationships, or links, between Components and Parameters. These links are multidimensional, so two components may be connected, for instance, by relationships in the power and thermal dimensions. Hierarchical relationships are also represented, such as between a subcomponent and component, which could be a thruster gearbox subcomponent and a thruster component.

4.7.3 Representing the State of the Vehicle

The Relational Model also stores the current state of the vehicle. In order to keep RECOVERY portable the sensor data stored in the Relational Model is divorced from the vehicle's own data network. A map must be created to detail which data from the vehicle's data network is equivalent to nodes in the Relational Model. A 'watcher' module then takes vehicle status data from the vehicle's data network and copies it into the relevant attributes of the nodes. Extra status information is also generated at this time, such as whether a particular component is active or not. This information is then used for diagnosis. Figure 4.7 shows the status information being generated, copied and stored in the Relational Model.

The sum total of all systemic sensor data together with the relationships stored in the Relational Model provides as complete a representation of the current state of the vehicle as possible given modelling and design constraints.

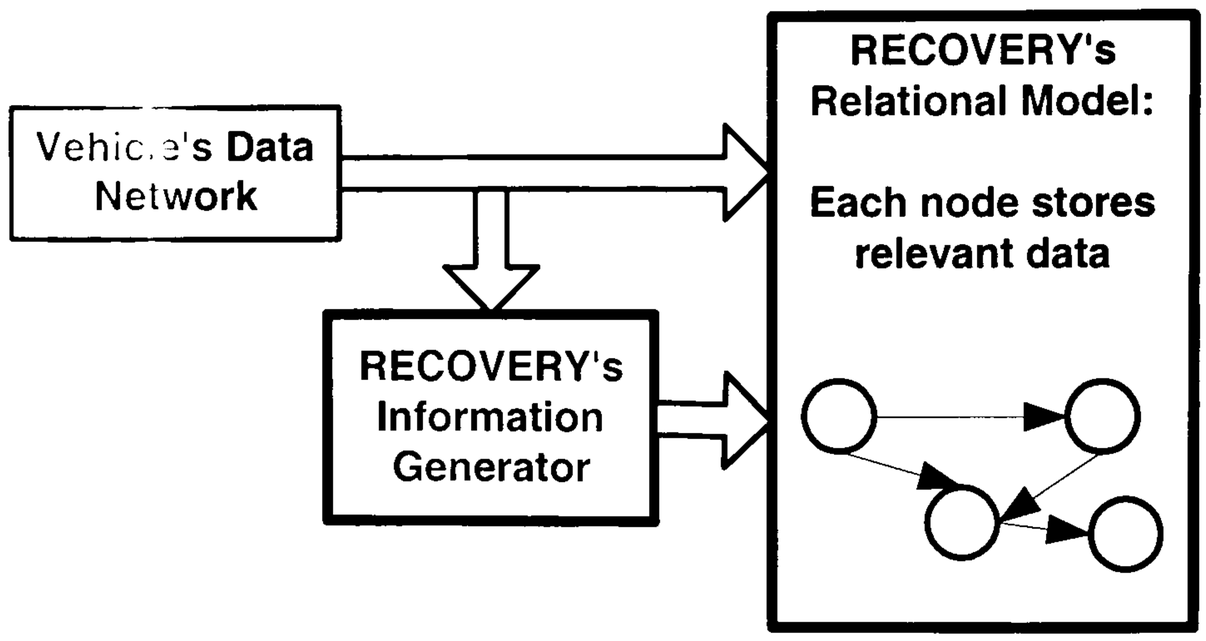


Figure 4.7: Representing the State of the Vehicle

4.7.4 Interfacing Heterogeneous Diagnostic Tools

The Relational Model provides an interface between different diagnostic tools by providing a common set of diagnosable components, as described in Section 4.7.2. Due to time constraints a simple form of interface, weighted nominations, was used. Each diagnostic tool nominates one or more components, the number of nominations are simply added together.

Where tools, such as Neural Networks, provide a belief value then a threshold is set, belief values above the threshold are counted as a nomination, values below are discarded.

4.7.5 Linking Knowledge to Components

To encompass modularity each component node may have component-specific diagnostic knowledge attached. Rather than imposing a single format for diagnostic knowledge the differences and diversity of heterogeneous knowledge types are accepted. Each component may have different types of knowledge attached, such as rules or models. Each type of knowledge is linked to the component node with an appropriate link, enabling the knowledge manager to pass the correct type of knowledge to the relevant diagnostic tool.

4.7.6 Reducing the Search Space

By having component-specific knowledge attached to the component it is possible to reduce the search space, or the amount of knowledge that must be processed before

the correct diagnosis is found.

For instance, if the only type of knowledge available was simple rules which were stored in a single block, then the Rulebased diagnostic tool would have to run through all the rules before finding the one that matched the current situation.

But if the rules were partitioned into blocks related to each component, and a method was provided for narrowing the search to particular components, then the number of diagnostic rules to be processed may be greatly reduced.

RECOVERY provides such a method using a Focus of Suspicion technique. As each diagnostic tool nominates components using low-detail, fast running information, the components most often nominated become the focus for a second round of diagnosis, during which the detailed, component-specific information is analysed.

4.7.7 Interfacing Heterogeneous Knowledge to the Diagnostic Tools

In order for the diagnostic tools to access the correct type of diagnostic information the links defined in the Relational Model are used to pass the correct information to them. For instance rules to a Rulebased tool, models to a Model-Based tool.

4.8 Domain-Independent Relational Diagnostics

The ability to use generalised, domain-independent fault diagnosis knowledge can be extremely useful. This knowledge, primarily gathered from human experts, can then be used to guide the diagnosis and model selection process. This realistic approach saves a lot of development time compared with getting the machine to learn these guidelines itself.

4.8.1 Domain Equivalence

There is a well-known equivalence between domains. For instance, friction in a mechanical system is equivalent to resistance in an electrical system, or a constriction in a hydraulic or pneumatic system. Pressure in a hydraulic system is equivalent to voltage in an electrical system, current flow is equivalent in both.

By expressing these equivalent relationships in the Relational Model, domain-independent diagnostic knowledge may be applied across many domains. To a certain degree, the knowledge used to diagnose a faulty electrical power supply can also be used for mechanical transmission systems - only the names of the units change.

4.8.2 The Need for Diagnostic Guidance

The need for diagnostic guidance results from the large solution space generated by the integration of the many and varied types of information. Each model takes time to evaluate, the more detailed models will take substantial time. Dynamic fault models, which are concerned with describing the behaviour of a faulty system over time, can take a great deal of processor time to look over a mission log file that could easily be several hundred megabytes long.

Much time can be saved by embedding human diagnosis knowledge and using this to steer the diagnosis procedure, either by eliminating some components and their associated information or by highlighting others. There are well known methods for the elicitation of knowledge from human experts developed for first generation rule based expert systems (described in Russell [90]). In future, it may be possible to get a robot to learn and enhance diagnosis techniques by itself, but at present this is impractical.

Domain-Independent Diagnostics can also provide the ability to diagnose components that are not directly sensed by analysing the effect that they have on other components. The Relational Model, consisting as it does of components and their inter-relationships, is particularly suited to this task.

4.9 Chapter Summary

This chapter detailed the original contributions from this research and introduced the RECOVERY concept. The RECOVERY architecture was then introduced and relevant concepts discussed. These were the types of heterogeneous knowledge available for automated robotic diagnostics, how the novel Relational Model provides common ground for heterogeneous diagnostic tools and knowledge, the need for

Search Space Reduction, and finally Domain-Independent Diagnostics.

The next chapter provides detail on each RECOVERY module and describes the overall operation of the system.

Chapter 5

RECOVERY Operation

5.1 Aim of this Chapter

This chapter details the operation of the existing implementation of the RECOVERY architecture. First the overall operation is shown, followed by detailed descriptions of the individual modules with examples where appropriate.

5.2 Overview of Operation

In general, RECOVERY follows the classic pattern of fault detection followed by diagnosis, as shown in Figure 5.1. Once the Relational Model has been compiled then RECOVERY runs as a background task, monitoring the vehicle and providing fault detection. During vehicle monitoring, status data is copied into the Relational Model and useful diagnostic information is generated.

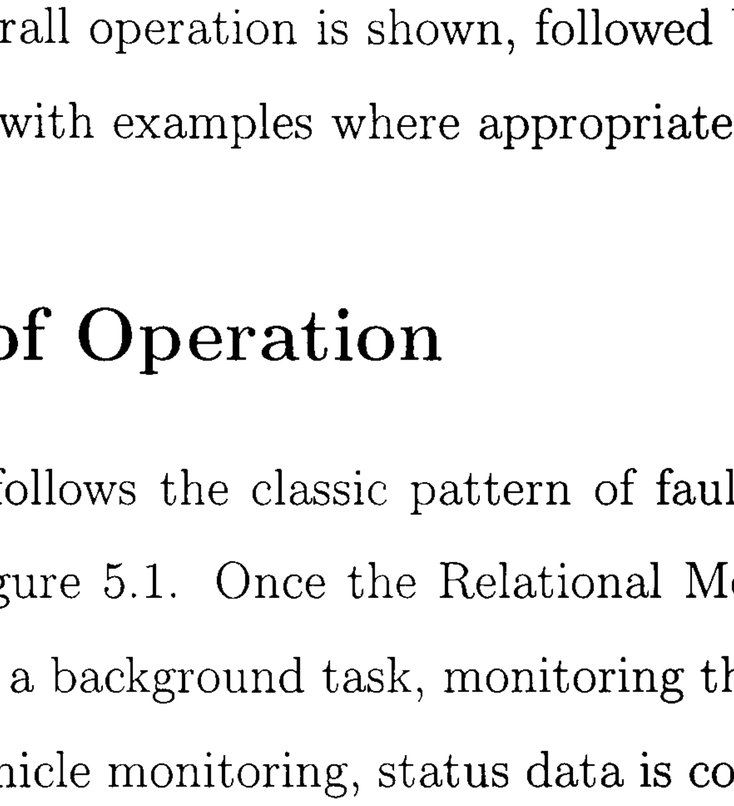


Figure 5.1: RECOVERY operation concept

The overall RECOVERY operation is shown in Figure 5.2. Detection is per-

formed by several methods ranging from simple rule triggering through residual generation to mission goal violation. Diagnosis is in two stages; component level and sub-component level. When diagnosis is complete the information is presented to the enveloping mission controller to allow enhanced fault accommodation.

Once a fault is detected the entire generation, detection and component-level diagnostic process is run over several iterations to provide increased diagnostic data. A post-diagnostic stage is then run using the novel Domain-Independent Hierarchy Module to look for common connections between faults. A list of most suspicious components is produced and used to reduce the diagnostic search space using a focus of suspicion method. The component-specific information attached to these components via the Relational Model is evaluated during the component-specific diagnostic level. The final diagnostic output is a list of suspicious components ranked in order of suspicion.

RAUVER, the evaluation robot, has a command and control loop that runs at 5Hz. Because of this the current RECOVERY implementation is invoked at a matching 5Hz so that every time RAUVER updates its status data then RECOVERY is invoked.

5.3 Full Recovery Architecture

The full RECOVERY architecture is shown in Figure 5.3. The modules fall into the following broad areas:

- Initiation
- Information Generation
- Fault Detection
- Fault Diagnosis
- Search Space Reduction

The Invocation Manager module controls the operation of the RECOVERY system, deciding when each module should be invoked. The following sections detail each module.

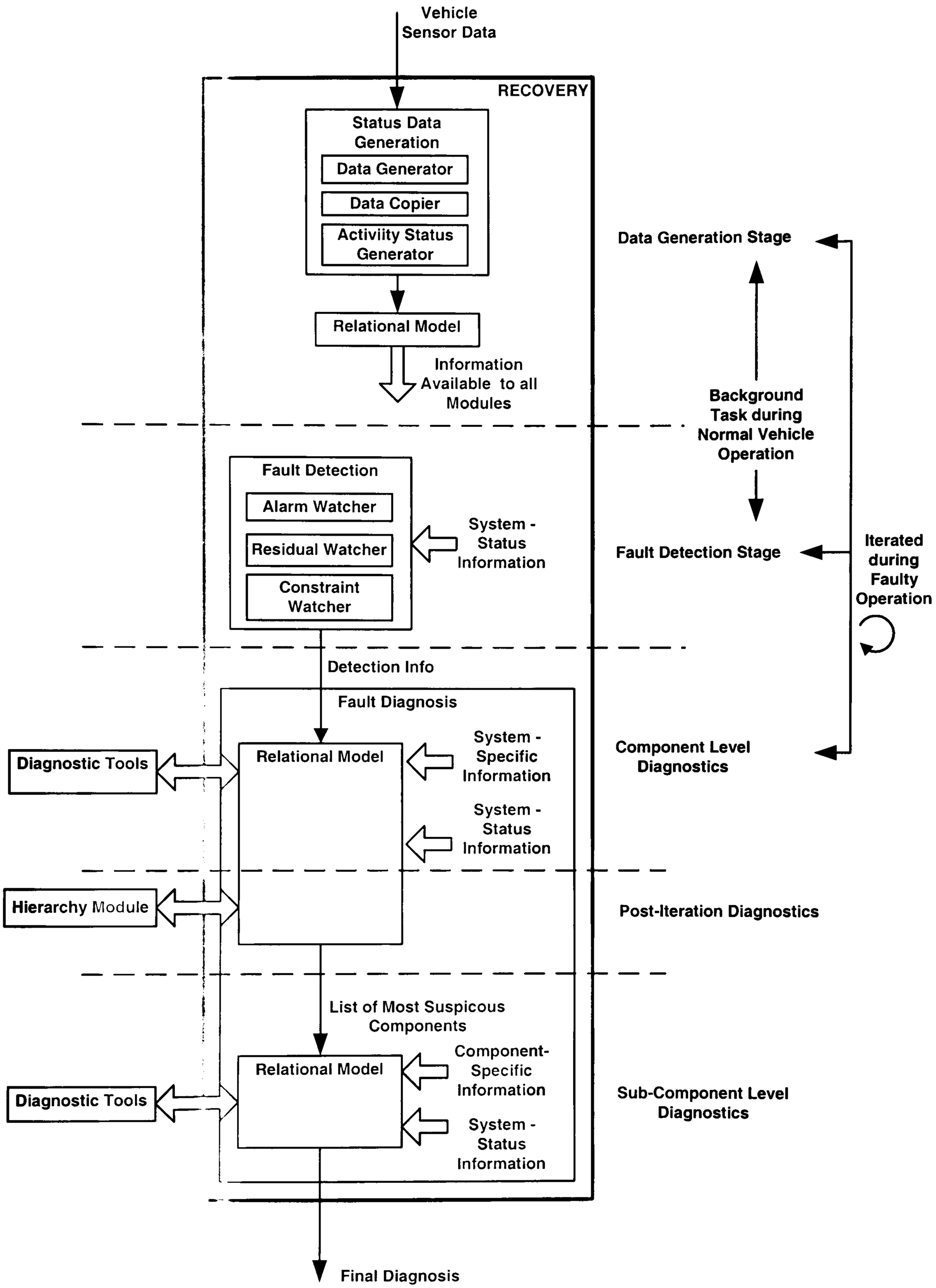


Figure 5.2: RECOVERY Operation

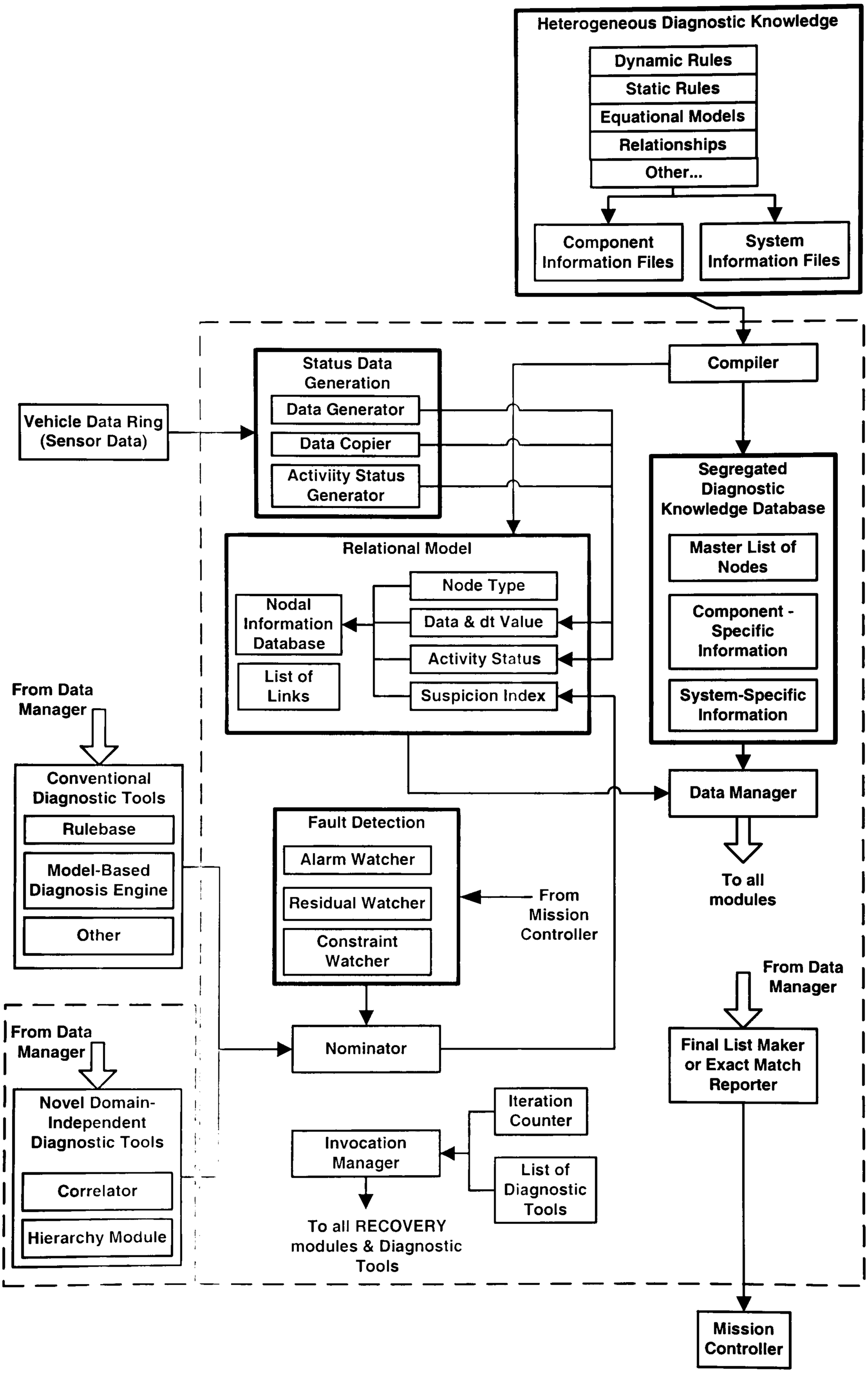


Figure 5.3: The full RECOVERY architecture. The modules enclosed in the dashed lines are original contributions from this research

5.4 Initiation: Building the Relational Model and Segregated Diagnostic Knowledge Database

The first step in using RECOVERY is to build the Relational Model and Segregated Diagnostic Knowledge Base. These are compiled at startup and then used throughout the detection and diagnosis stages. The knowledge contained in them is available to all modules via the Data Manager.

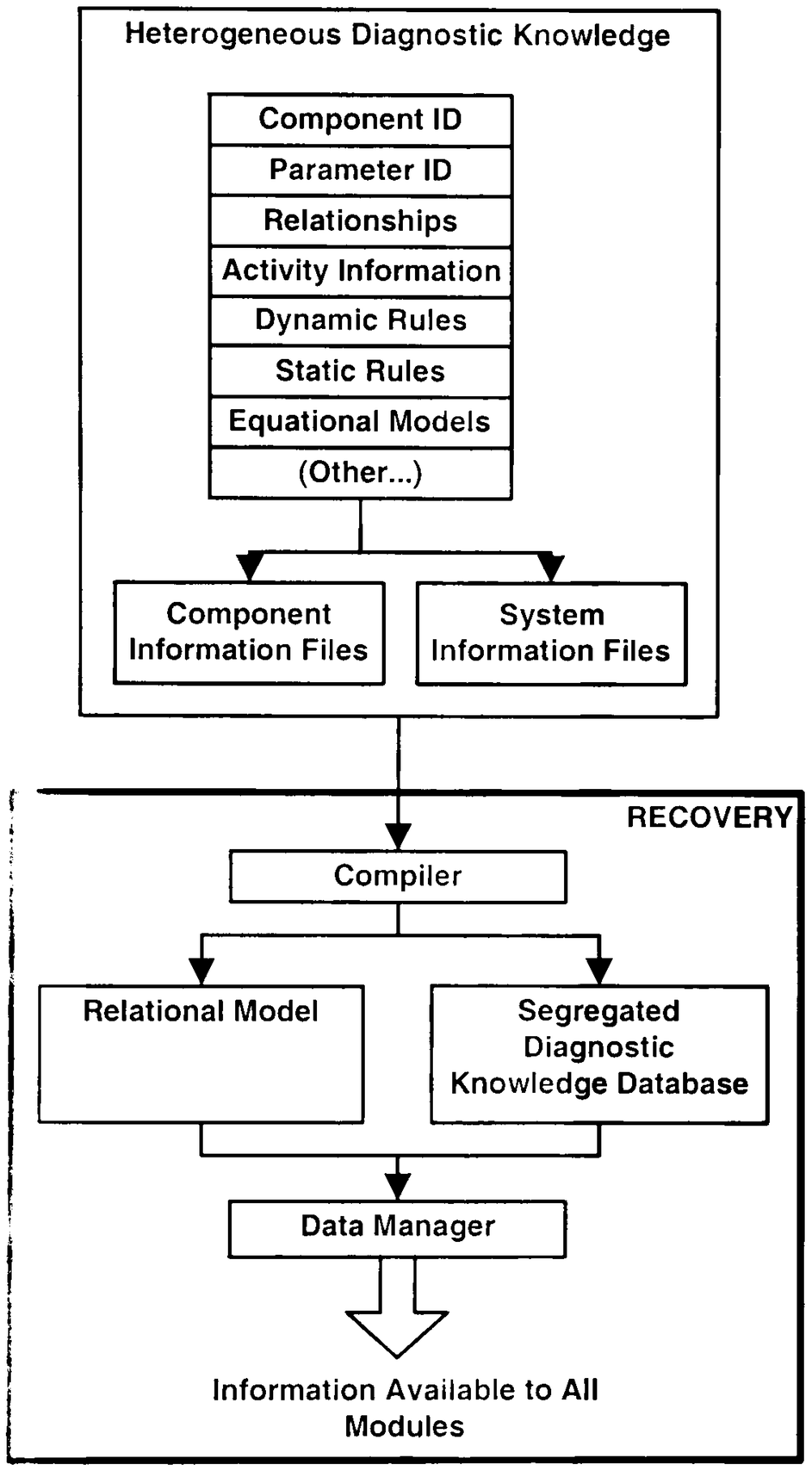


Figure 5.4: RECOVERY initialisation operation

5.4.1 Component & System Information Files

All information relevant to a component is placed in a knowledge file (a text file), which is then linked into the main vehicle library during compilation. The main

vehicle library is segregated into different types of knowledge, such as rules, equational models, links between components. This allows different types of diagnostic tool easy access to the relevant type of information. The generation of component information can be performed by the vehicle engineer using knowledge available from the component manual or, preferably, more detailed information can be obtained from the component manufacturer.

The file also system level information such as the immediate relationships between this and other components. For instance, the file will include details of which power supply a component is connected to. The full range of links is detailed below.

5.4.2 The Compiler

RECOVERY has its own compiler for producing a Relational Model from the Component Information Files. This makes RECOVERY highly portable and enables easy modification of the model to reflect changes in vehicle configuration.

The compiler copies the knowledge stored in the file into a central segregated diagnostic knowledge database, with different types of information being stored in homogeneous blocks. Links between components are verified before being added into the Relational Model.

5.4.3 Nodal Information Database

The Relational Model effectively consists of multidimensional blocks of nodes and links. The Nodal Information Database is the heart of the Relational Model, where each node represents a component or parameter and stores several modifiable attributes. The types of node currently used by RECOVERY are shown below.

Types of Node

- Systemic Sensor Component: A sensor that senses the state of the vehicle
- Environmental Sensor Component: A sensor that senses the state of the environment

- Power Supply Component: A 'source' component that provides power to other components
- Component: A generic component of no specific type and supplied with power by a Power Supply Component
- Sub-Component: A component that is part of another component and is only diagnosable using the component-specific knowledge attached to that component.
  - Sensed Parameter: A parameter that stores data from a sensor
  - Unsensed Parameter: A parameter that is not sensed but is used to store a desired value
  - Modelled Parameter: A parameter that stores data produced by a model
  - Alarm Parameter: Used to store an alarm flag

Attributes of Nodes

Each node has several attribute slots which are used to store data on various aspects of the node:

- Type of Node: An enumerated value representing the type of node
- Data & Data dt: Parameter nodes store sensor data and their automatically generated differential here
- Activity Status: A flag generated using component-specific activity information showing whether the component is active at this moment in time
- Suspicion Index: A value reflecting the number of nominations this component has received from the various diagnostic tools

5.4.4 Reducing the Search Space with the Suspicion Index

Every time a diagnosis module, such as a diagnostic engine, nominates a component then that component's Suspicion Index is incremented by the Nominator module.

The size of the increment is called the Nomination Increment. The Suspicion Index is one of the attributes of that component's node in the Relational Model and is used to determine which components should have their component-specific diagnostic knowledge evaluated in the sub-component diagnosis stage.

By highlighting components that have been proposed as fault candidates the amount of diagnostic information to be analysed is reduced compared with simply running through the entire set of diagnostic knowledge.

5.4.5 Master List of Nodes

The Nodal Information Database is accessed using Node Identifiers (Node IDs), a table of which is generated during compilation. This is called the Master List of Nodes.

Each Node, whether it is a component or parameter, has a unique identifier which is stored in a Master List. The list is sectioned into different types of node, such as Power Supply Component, Modelled Parameter, Sensed Parameter etc. Each section has delimiters, allowing easy access to either an individual node or groups of homogeneous nodes.

```
40: FIRST_MODELLED_PARAMETER
41: PORT_NAV_POWER_MODEL
42: STBD_NAV_POWER_MODEL
43: ROLL_MODEL
44: PITCH_MODEL
45: DEPTH_VELOCITY_MODEL
46: YAW_VELOCITY_MODEL
47: HEADING_MODEL
48: DEPTH_MODEL
49: LAST_MODELLED_PARAMETER
```

Figure 5.5: A Segment of the Compiled Master List of Node Identifiers

A sample segment of the compiled Master List of Node Identifiers is given in Figure 5.5. The segment begins and ends with the section delimiters, with Modelled

Parameter identifiers in between.

5.4.6 List of Links

Links do not have unique identifiers as they exist purely to specify a type of relationship between uniquely identified nodes. Links are segregated by type, such as Component to Power Supply, Sensed Parameter to Sensor, etc. Homogeneous types of links may then be operated on, or searches performed to find links of various types attached to an individual node. The links currently implemented in RECOVERY are described below:

Component to Component

- Component to Power Supply Component: This should more properly be called 'Sink to Source Component' as it is a domain-independent link. It links any type of component (including a Power Supply Component) to a Power Supply Component. For instance, a 5V power supply may be connected to (supplied from) the 12V battery supply.

- Sub-Component to Component: Indicates that the sub-component may only be diagnosed using component-specific information attached to the component

- Sensor Component to Sensed Component: Some systemic sensor components may directly sense the state of another component. For instance, a temperature sensor is attached to the Port Navigation Power Supply, which would be represented using this type of link

Component to Parameter

- Sensor Component to Sensed Parameter: Shows which parameter stores the data gathered from a particular sensor

- Unsensed Parameter to Component: Some parameters are not sensed but instead hold desired values, such as desired thrust. This link shows which component the desired value is related to, such as desired thrust linked to an open-loop thruster controller

Parameter to Parameter

- Modelled Parameter to Sensed Parameter: This shows that the modelled parameter is intended to reflect the state of the sensed parameter, for instance modelled vehicle roll to sensed vehicle roll

- Sensor to Equivalent Sensor: Shows that two sensors can be considered equivalent, or overlapping, such as a magnetic compass and a gyro compass

Figure 5.6 shows a segment of the compiled list of links, detailing part of the Sensor Component to Sensed Parameter section. The first node on each line is the Sensor Component, in this segment five are shown: the AOSI Compass, the Port and Starboard battery current sensors (called LEMs), and the Port and Starboard Powerstack temperature sensors. The second node on each line is the Parameter Node that stores the data from that sensor component.

```
SENSOR_PAR :
0: AOSI_EZ3_COMPASS_SENSOR          AOSI_HEADING
1: AOSI_EZ3_COMPASS_SENSOR          AOSI_PITCH
2: AOSI_EZ3_COMPASS_SENSOR          AOSI_ROLL
3: AOSI_EZ3_COMPASS_SENSOR          AOSI_TEMP
4: AOSI_EZ3_COMPASS_SENSOR          AOSI_MAGX
5: AOSI_EZ3_COMPASS_SENSOR          AOSI_MAGY
6: AOSI_EZ3_COMPASS_SENSOR          AOSI_MAGZ
7: PORT_LEM_SENSOR                  PORT_LEM
8: STBD_LEM_SENSOR                  STBD_LEM
9: PORT_POWERSTACK_TEMP_SENSOR      PORT_DIODE_TEMP
10: STBD_POWERSTACK_TEMP_SENSOR     STBD_DIODE_TEMP
```

Figure 5.6: A Segment of the Compiled Master List of Links

Some sensors generate more than one parameter, for instance the AOSI Compass generates Heading, Pitch, Roll and more. Because of this each sensor (and other types of nodes) may have more than one link to them.

5.4.7 A Segment of a Compiled Relational Model

Figure 5.7 shows a section of the Relational Model that details the links between some common parameters and components. In this figure can be seen four types of nodes and four types of links, where the links are lines:

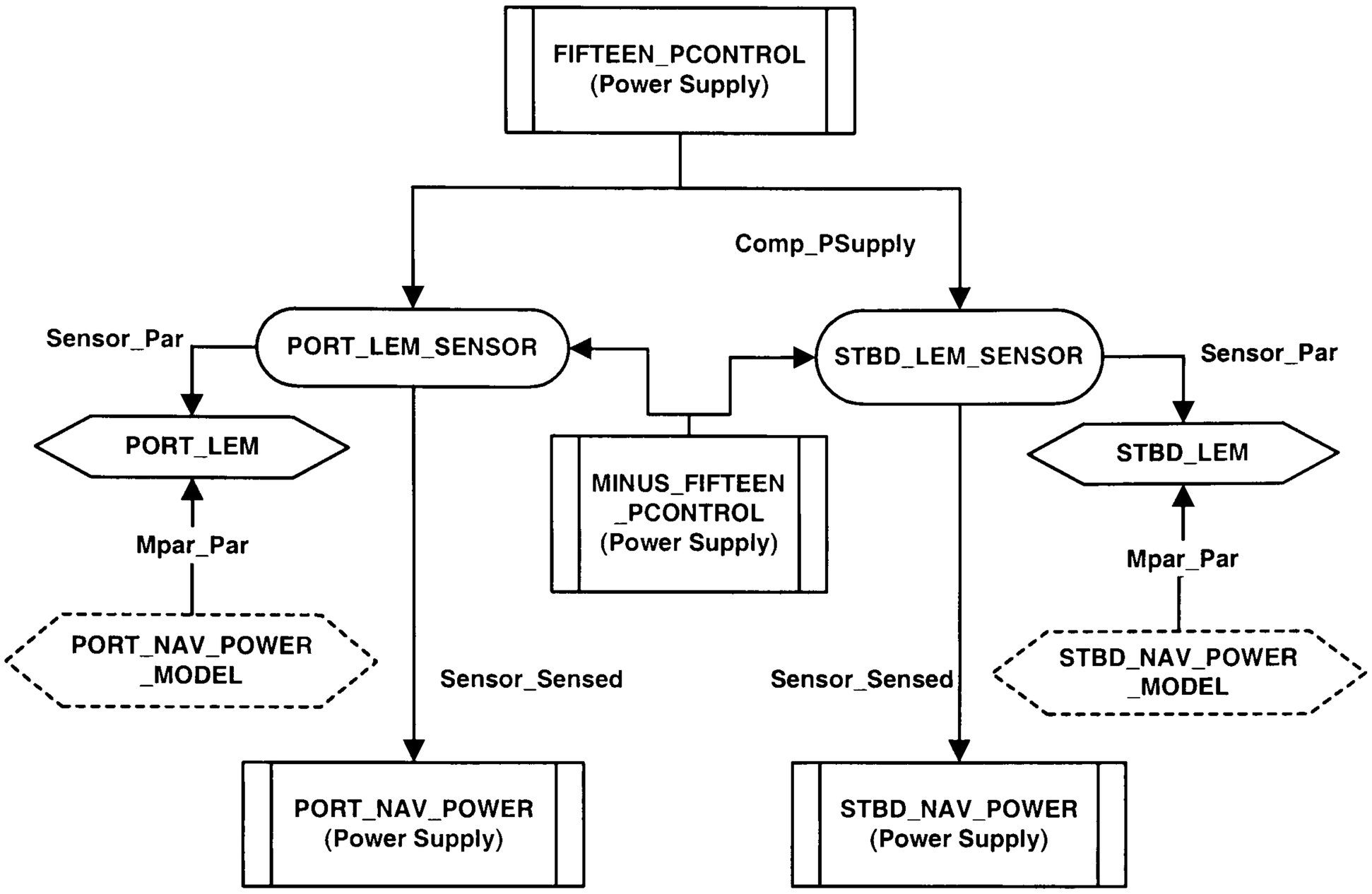


Figure 5.7: Relational Model Segment Example

- Node 1: Power Supply (source) Component
- Node 2: Systemic Sensor Component
- Node 3: Modelled Parameter
- Node 4: Sensed Parameter

- Link 1: Between the 'FIFTEEN_PCONTROL' and the 'PORT_LEM_SENSOR' component is a link showing that the Sensor is supplied by the 15V Power Supply.
- Link 2: Between 'PORT_LEM' and the 'PORT_NAV_POWER_MODEL' is a link showing that the the modelled PORT_NAV_POWER_MODEL parameter

models the sensed PORT_LEM parameter, this information is later used for fault detection.

- Link 3: Between the PORT_LEM_SENSOR component and the PORT_LEM sensed parameter is a link showing that the parameter is sensed by that sensor.

- Link 4: Between PORT_LEM_SENSOR sensor component and the PORT_NAV _POWER power supply component is a link showing that the power supply is sensed by the PORT_LEM_SENSOR.

The complete Relational Model used to evaluate this research is detailed in Appendix B.

5.4.8 Segregated Diagnostic Knowledge Database

The compiler also produces the Segregated Diagnostic Knowledge Database as shown in Figure 5.8. This stores all component-specific information in a single database, segregated into different types of knowledge. Each component has its own selection of different types of diagnostic information such as rules, equational models, etc. This aids search space reduction as once a component has been found suspicious then its specific information is readily retrievable.

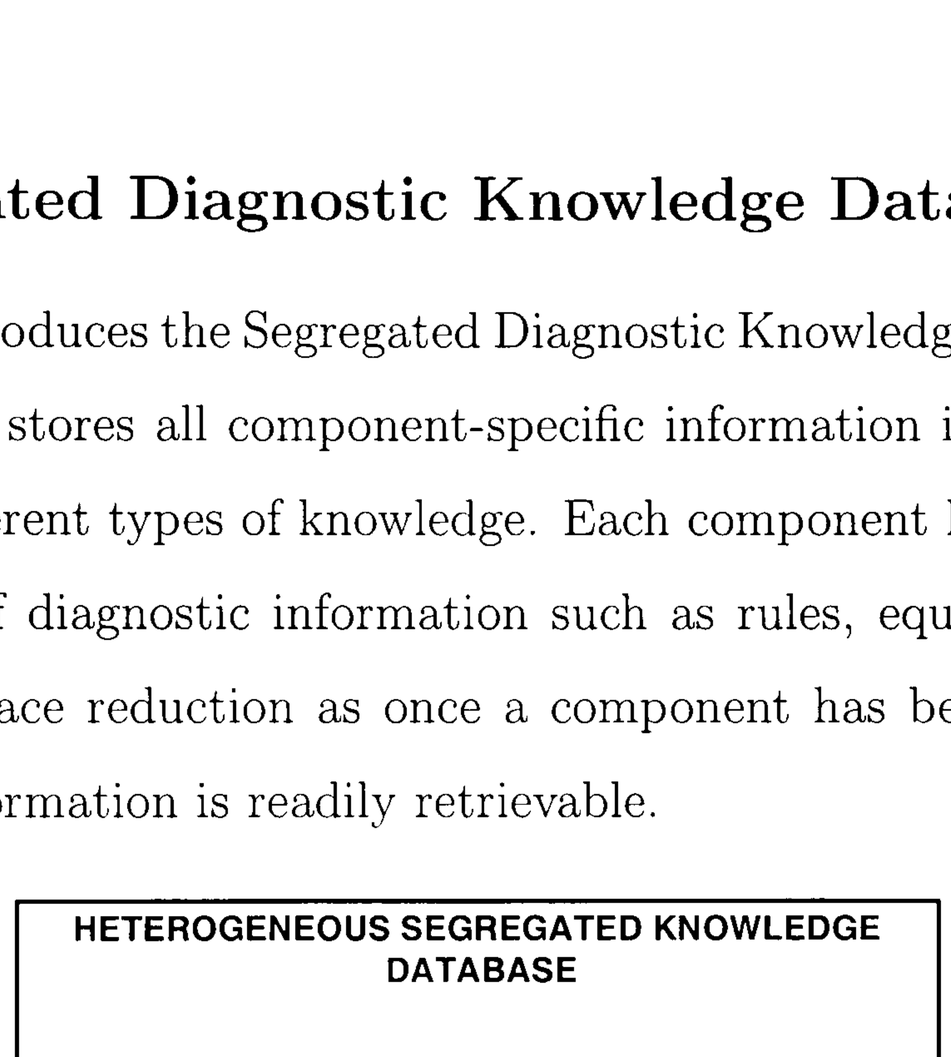


Figure 5.8: The Segregated Diagnostic Knowledge Database

5.5 Domain Independent Tools

RECOVERY contains two types of domain-independent diagnostics. The Correlator Module looks for correlations (mutual relationships) between components and parameters. The Hierarchy Module looks for common connections between components that have been diagnosed as faulty by the various diagnostic tools. It then searches the hierarchical relationships specified in the Relational Model to determine whether a single component could be responsible for multiple faults.

5.5.1 The Correlator Module

The Correlator Module (or Correlator) is one of the main RECOVERY modules that use domain-independent diagnostic knowledge. At present, this module uses the following domain-independent diagnostic information:

- Delta Index: Parameters that track faulty parameters are likely to be related to the fault.

- Recently Used: Components that become active (after a period of inactivity) just before the occurrence of a fault are likely to be related.

- Active: Components that are being used at the time of a fault are more likely to be related to the fault than inactive components.

When a fault occurs the Correlator looks back through the mission log file (which represents the temporal dimension) and looks for the correlations described above. If correlations are found between various components and parameters this information is used to modify the Relational Model by incrementing the relevant component's Suspicion Index. This modification effectively highlights suspicious components or parameters, which are then used to guide the diagnosis engines towards sections of knowledge thought to be most relevant.

Delta Index: An example of this is thermal drift of sensor readings. If a wheeled robot had a speed sensor and a maximum set speed and the measured speed exceeded maximum speed then a snapshot of the system would show only that the robot was travelling too fast. By looking back at the history of the vehicle, it could be seen

that the maximum speed of the robot was creeping up at approximately the same rate as the temperature. This would represent a correlation as described in item 1 above: "Parameters that track faulty parameters are likely to be related to the fault".

A correlation of this kind would be extremely useful in diagnosing the actual fault. This is, perhaps, a rather simplistic example. The robot is assumed to be travelling constantly at maximum speed, which is unlikely to be the case. Even so, the concept is valid as the Correlator would have noticed the association between temperature and sensor drift without this knowledge being explicitly represented.

For a parameter to be nominated as tracking a faulty parameter its delta index (a value that relates to the parameters motion over time) must be within a certain amount of the faulty parameter's delta index. This amount is called the Delta Window.

Recently Used: This is as discussed in Chapter 4. For instance, if the brakes are applied on your car after being unused for a while, and there is immediately a loud bang followed by a great deal of shuddering, a fair assumption is that there is a brake problem. Two parameters are needed to match this type of information; the length of time of inactivity and the length of time before the fault that the component was activated.

Active: This simply states that an active component is more likely to contribute to a fault than an inactive one. Note that this means a component that is supposed to be active, including components that are inactive through failure.

5.5.2 The Hierarchy Module

This again uses domain-independent knowledge in a similar way to the Correlator, but where the Correlator is primarily concerned with searching the mission log file the Hierarchy Module checks relationships between Relational Model nodes. It uses the following domain-independent diagnostic information:

- If several components are diagnosed as faulty and they have a common supply connection which is further up the hierarchy then it is the commonly connected component that is likely to be faulty

- If a commonly connected supply component has been nominated as faulty then the supplied components should be exonerated of suspicion.

For instance, if multiple components are thought to be faulty and there is a common connection then it is likely that the connection is the problem. This is readily apparent with power supplies; if all the active thrusters 'fail' then it is likely to be the Navigation Power Supply that is actually at fault. Power Supply connections are numerous and one of the most common forms of component information. It is readily available on AUV circuit diagrams, and so forms one of the most common types of link in the Relational Model.

For a common source component (i.e. a power supply) to be nominated as faulty then a certain amount of its active supplied components must have been nominated as faulty. This is known as the Nomination Threshold.

5.5.3 Invocation of Domain-Independent Diagnostic Tools

Because it looks for links between components and parameters the Correlator is effectively a diagnostic tool, and so is invoked at every iteration. The Hierarchy Module is concerned with components that have been diagnosed as faulty. To increase the performance of this module it is invoked after all iterations are finished, when faulty components have the most chance of having been detected or diagnosed. At present it only has a single invocation and so it can only find the first level of common connections. Future versions of the Hierarchy Module will be able to move several levels up the hierarchy, finding common connections between the first level of common connections.

5.6 Diagnostic Tools Used

Currently three diagnostic paradigms are integrated with RECOVERY: Rulebases, Model-Based Diagnosis and the novel Domain Independent Diagnosis techniques proposed in this research. These are represented by five different diagnostic tools: a Model-Based Diagnosis engine, static and dynamic rule-based pattern matchers, a

domain-independent Correlator Module and a domain-independent Hierarchy Module.

The Model-Based Diagnostic engine is used to generate fault candidates, specifically, components that are thought to be faulty. This is performed using standard observe/perturb/match model based diagnosis techniques as described in Section 3.4. Currently, RECOVERY supports equational models, such as common movement models, together with static and dynamic rules. Multiple faults may be diagnosed in some cases. For a model-based diagnostic engine to nominate a component as faulty the modelled output must be within a certain amount of the sensed output, this is known as the nomination window.

There are two types of rule-based pattern matcher, static and dynamic, currently used with RECOVERY. Static matchers are only capable of finding faults that can be diagnosed with a single snapshot of the system. Dynamic matchers can find faults that are only apparent from studying the system behaviour over time. Due to the large amount of information involved in dynamic diagnosis it can take far longer than static diagnosis.

The domain-independent diagnostic tools are described above in Section 5.5.

5.7 Diagnostic Knowledge Used

The types of knowledge integrated into RECOVERY reflect the choice of integrated diagnostic paradigms. Standard if-then rules are supported, these are used to represent static faults, dynamic faults and domain-independent diagnostic information. Equational models are also supported, these are currently used only for static fault diagnosis. At present vehicle movement (roll only) and vehicle power consumption (port and starboard hulls) are modelled.

5.8 Modularity

RECOVERY is a modular architecture in that different diagnostic tools and knowledge may be inserted or removed at will. The actual implementation of RECOVERY was limited by time and so there is enough modularity to show that the concept is

valid. For instance, although each diagnostic tool may be removed easily this must be done by commenting out a line in the source code rather than by checking a tick box on the screen. In a similar way the equational models are embedded in the source code rather than scanned in from a text file. This is because the C compiler would effectively have to be duplicated in order to support mathematical functions etc. It was felt that this was not relevant to the focus of the research and so the equational models were inserted into standard C header files and compiled as part of the source code.

5.9 Status Data Generation

The first step of RECOVERY operation is to copy sensor and status data from the vehicle's data ring and to generate other useful information for detection and diagnosis. This performed in three stages; data copying, data generation and activity status generation.

5.9.1 Status Data Copying and Generation

The generation and processing of raw sensor data is left to the vehicle's own sensor suite. RECOVERY copies relevant data into the Relational Model using the Data Copier, this uses a map that specifies which vehicle data parameter is copied into which Nodal Information Database slot. Once the copying operation is complete RECOVERY automatically generates Rate of Change values using the Data Generator for all data parameters as this information is useful for diagnosis.

5.9.2 Activity Status Generation

Component activity information is generated using the activity information contained in the Component Information Files. For instance, the activity logic for a thruster is 'if Speed != 0 then Thruster is Active'. Activity information is extremely useful for diagnosis. Extra useful diagnostic information is also generated at this time, such as the time of each component's last change of activity status, both active to inactive and vice versa.

Generation of this information as a background task during normal operation greatly eases the processor load when performing diagnostics. For instance, to scan back through a mission log file and generate this can take a long time with large mission log files. Although it is essentially simple information that takes few processing operations, this adds up over a mission that may take several hours. Generating it at each invocation spreads the load over the normal operation period when processor power is less in demand. This technique also reduces overall diagnostic time.

5.10 Fault Detection

The RECOVERY Fault Detection modules take responsibility for all fault detection using information provided in the component information files and by the onboard mission controller. Firstly, the Residual Watcher module scans all modelled and sensed parameters for residuals (bigger than that generally caused by noise, etc) using the links defined in the Relational Model. Secondly, the Alarm Watcher scans all alarm parameters and limited parameters (such as control stage temperature) to see that they are within their limits. Thirdly, the Constraint Watcher checks to see that the onboard mission controller has not flagged any violated constraints for the current goal.

RECOVERY's detection operation is shown in Figure 5.9.

5.10.1 Residual Generation

The Relational Model contains links of various kinds between parameters and components. One of the link types defines a relationship between a modelled parameter and an observed parameter. Part of the RECOVERY procedure is to scan through all of these link types using the Residual Watcher module and check that the modelled parameters match the sensed (observed) parameters. Any discrepancy means that the vehicle is diverging from normal operation and the diagnostic operation is invoked.

Because of the presence of noise, parameters are allowed to differ from their models by a certain amount without causing a fault to be detected, this amount is

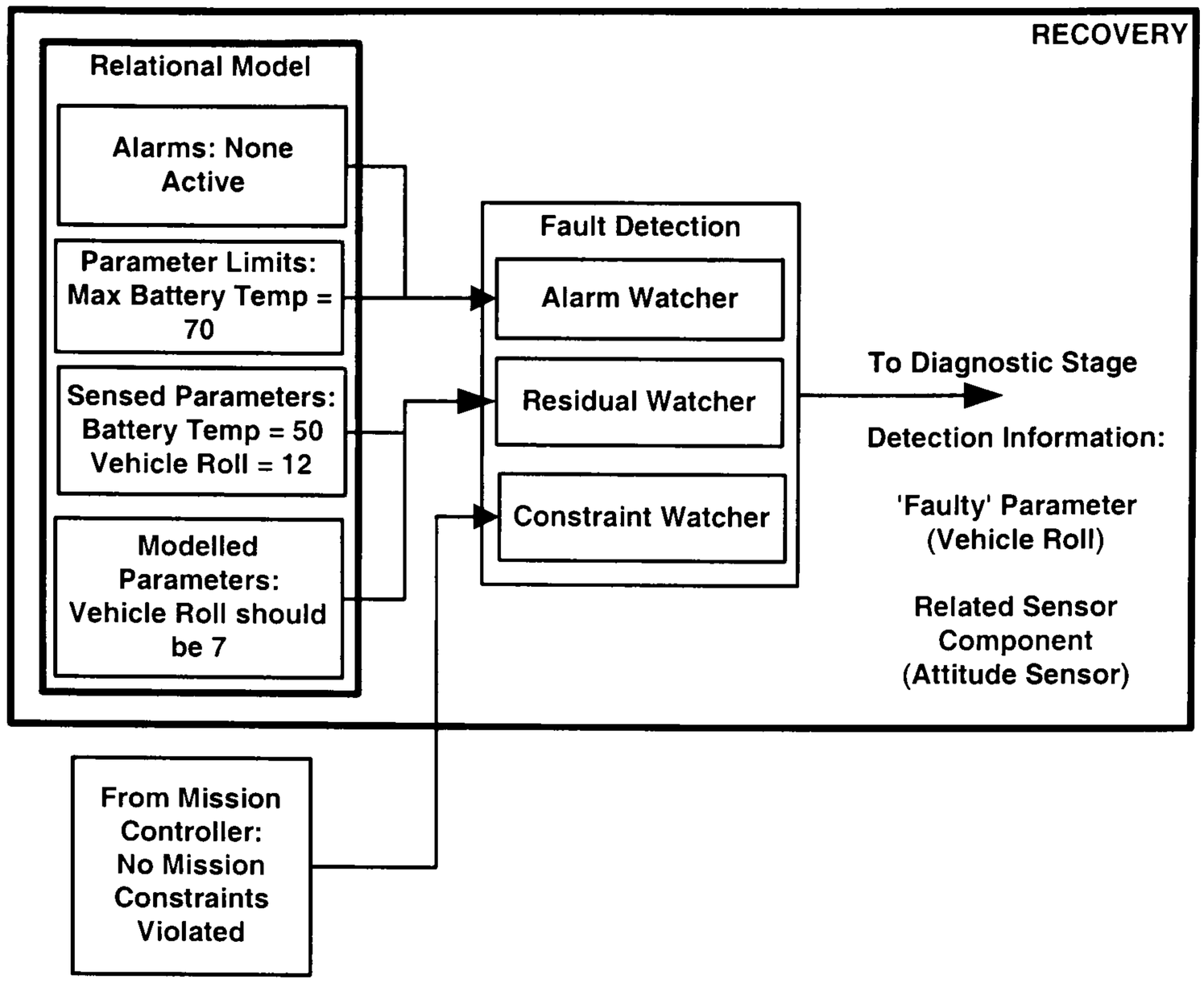


Figure 5.9: RECOVERY Detection Operation

called the Residual Detection Window.

This scanning method provides an easy way to check for system-wide discrepancies without having to specifically code a check routine for each modelled parameter.

5.10.2 Constraint Violation

A common method of detection is to use Constraint Violation, where an alarm flag is set if a parameter exceeds a given threshold, such as a component exceeding a maximum temperature. RECOVERY's Alarm Watcher module also scans parameters that are classed as alarms, where the constraint-exceeded flag is set by the sensor's hardware (such as a thermostat tripping).

This technique can be extended to constrained, goal-orientated mission controllers. If a mission constraint is exceeded, such as vehicle roll, this can be reported to RECOVERY's Constraint Watcher module as all parameters are represented in the Relational Model. This is simplified if the mission controller is tied in to the Relational Model.

5.10.3 Passing of Information to Diagnostic Stage

Any parameters that are found to be 'faulty', whether from alarms, residual generation or constraint violation, have their attached sensor component's Suspicion Index nominated to allow for the fact that the sensor could be faulty. This information is available to the diagnostic stages to assist in diagnosis.

5.11 Diagnostic Operation

Diagnosis is performed in two stages: component-level and sub-component-level. During component-level diagnostics the search is narrowed down to a short list of highly suspect components, after which Sub-Component Diagnosis runs the component-specific information attached to each component to try to identify the fault in greater detail. The Invocation Manager runs each diagnostic tool in sequence using the List of Diagnostic Tools.

RECOVERY's diagnostic operation is shown in Figure 5.10.

5.11.1 Component-Level Diagnostics

During this stage the model-based diagnosis engine is invoked, with the detection information passed to the diagnosis module used to identify which models are to be run. For instance only models attached to faulty parameters in the Relational Model are passed to the diagnosis engines, so saving time and processor operations.

The Domain Independent Correlator module is also invoked at this time in order to scan for relationships between faulty components and parameters.

5.11.2 Diagnostic Iterations

Because faults may show in different dimensions at different times, the Component-Level Diagnosis is run through several iterations over several seconds. The Iteration Counter records the number of iterations. For instance, in the Results section it is apparent that whereas a fault may be instantaneously apparent in the power dimension, it may take several seconds to show in the movement dimension. This is due to the relatively slow dynamics of AUVs. By running Component-Level

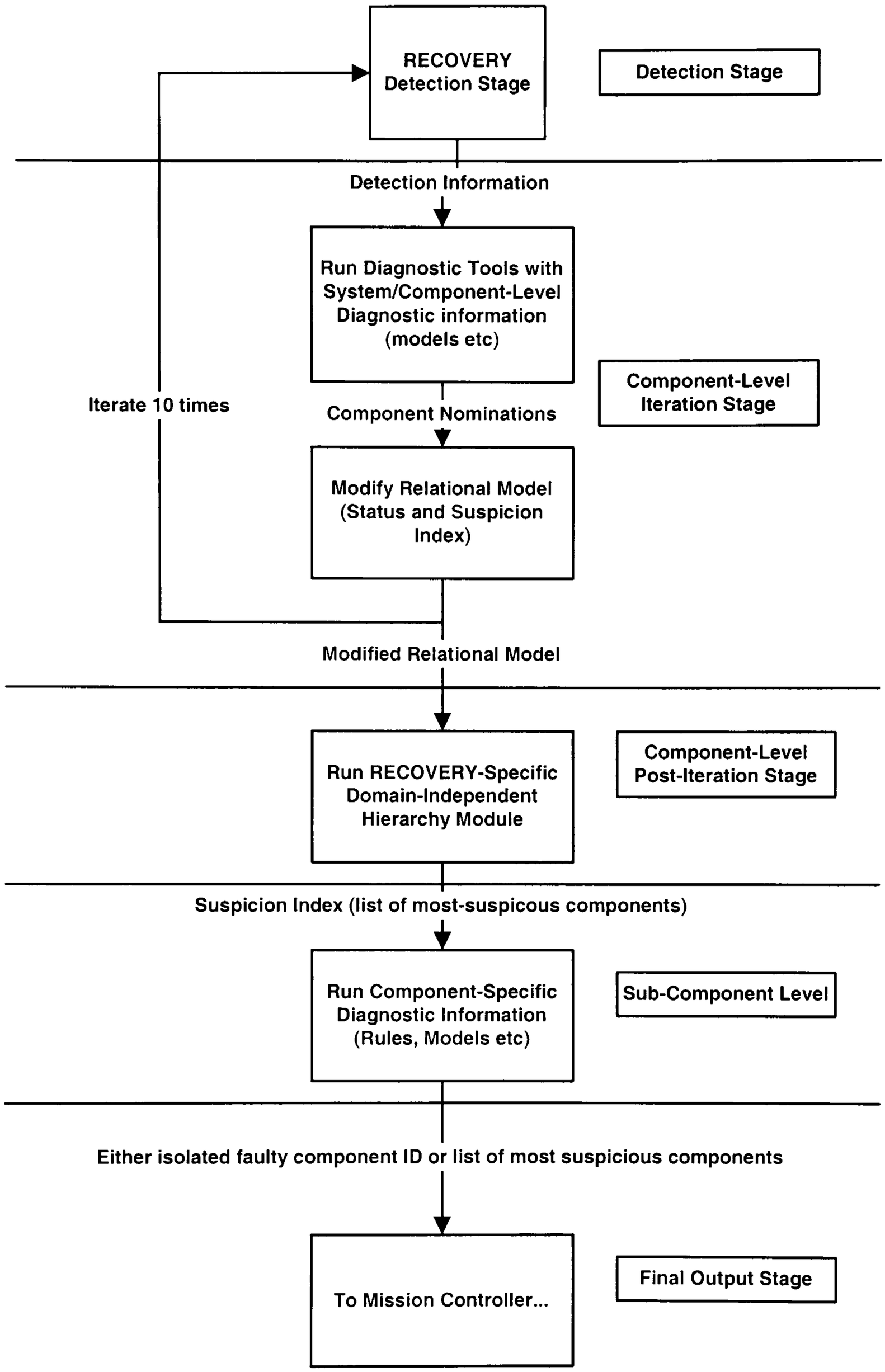


Figure 5.10: RECOVERY Diagnostic Operation

Diagnostics for several seconds the fault is given a chance to show up in all relevant dimensions, so increasing the knowledge available to the diagnostic system.

5.11.3 Post-Iteration: Domain-Independent Hierarchy Module

Once a predetermined number of diagnostic iterations have been performed the Domain-Independent Hierarchy Module is invoked in order to find patterns of faults that may indicate a common cause, as discussed in Section 5.5.

5.11.4 Selection of Most Suspicious Components

At the end of the Component-Level diagnostic stage the Final List Maker uses the suspicion index slots of the Nodal Information Database to produce a list of suspicious components, ranked in order of suspicion. This list is passed to the Sub-Component-Level diagnostic stage for deeper analysis.

5.12 Sub-Component-Level Diagnostics

After component-level diagnostics are complete the focus moves to sub-components, where the component-specific information attached to each component is analysed. The component-specific diagnostic information was compiled into a central vehicle library at the initial compilation stage. Each component has a delimited section, accessed using the unique component node identifier of each segregated knowledge store (static rules, dynamic rules) of the library. This allows each suspicious component's specific knowledge to be easily accessed by the various diagnostic tools.

The knowledge belonging to the most suspicious component is evaluated first, if no match is found then the focus moves down the list to the next component.

The static rules associated with the component are evaluated first as they are much faster to run than the dynamic rules. If a direct match is found the diagnostic process is terminated and the information passed to the mission controller. If no match is found then the dynamic rules are evaluated, these take a great deal longer to run as a lengthy mission-log file must be scanned for each rule.

5.13 Providing Information to Mission Controller

If, after all the suspicious components have been evaluated, with no direct match found, the list of suspicious components produced at the end of the Component-Level diagnostic stage is passed to the mission controller as a 'best guess'.

5.14 Chapter Summary

This chapter detailed the operation of RECOVERY. Overall operation was described first, followed by a detailed description of the initialisation stage when the Relational Model and Central Vehicle Library are compiled. The operation of the two types of novel domain-independent diagnostic tool was detailed. The information generation and fault detection stages were then shown, followed by the operation of the two levels of the diagnostic stage.

The next two chapters present results from real-world experimental evaluation of RECOVERY. Chapter 6 evaluates the novel domain-independent diagnostic tools in isolation. Chapter 7 evaluates the full system using all supported diagnostic tools and knowledge. RECOVERY's focus of suspicion method for reducing the diagnostic search space is then evaluated.

Table 5.1 shows the location of the various experiments used to evaluate the original contributions of RECOVERY.

Claim	*Experiment*	*Location*
$Architecture$	All	$Chapters 6, 7$
$Relational - Model$	All	$Chapters 6, 7$
$Operational - Methods$	All	$Chapters 6, 7$
$Search - Space - Reduction$	SSR	$Section 7.3$
$Domain - Independent - Diagnostics$	DID	$Chapter 6$
$Real - World - Implementation$	All	$Chapters 6, 7$
$Supporting - Work$	$Literature$	$Section 4.6$

Table 5.1: Table Linking RECOVERY Originality to Experiments

Chapter 6

Results: Evaluation of Domain-Independent Diagnostics

This chapter and the next present results that evaluate individual sections and the full RECOVERY system. Experiment setup is detailed for each experimental section. The results consist mainly of verbatim output from RECOVERY and this is summarised in tabular form where appropriate. This Chapter (Chapter 6) details evaluation of the novel domain-independent diagnostic tools. The following chapter (Chapter 7) evaluates the full RECOVERY system and the Search Space Reduction methods in real-world conditions.

6.1 Aim of This Chapter

This chapter presents results that evaluate the RECOVERY-specific Domain Independent Diagnostic methods.

The evaluation vehicle, RAUVER, is described and its operational fault log, recorded during development and deployments, is presented. The reservoirs used for running the RECOVERY evaluation missions are then shown.

Details of the experimental setup and the format of the results are followed by experiments evaluating the Domain-Independent Hierarchical and Correlator Modules.

6.2 RAUVER: The Evaluation Vehicle

RECOVERY is being developed and evaluated on the Ocean Systems Laboratory's own vehicle, RAUVER; a 2m, twin hulled, catamaran submersible capable of either remote or autonomous operation. RAUVER is shown in Figure 6.1.

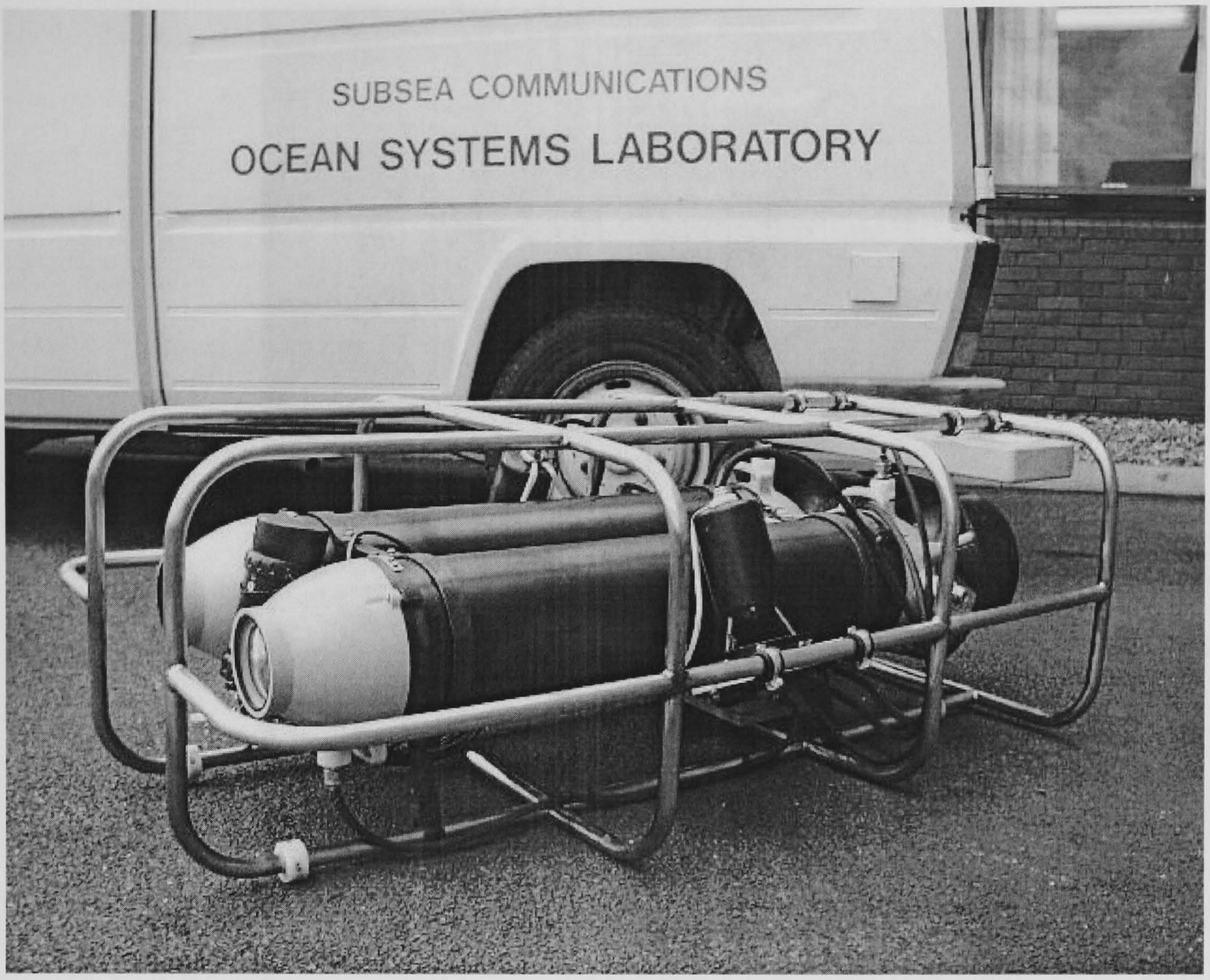


Figure 6.1: RAUVER, The Evaluation Vehicle

RAUVER can be operated as either a Remotely Operated Vehicle (ROV) similar to those used in commercial sub-sea intervention, or as a true Autonomous Underwater Vehicle (AUV). Designed primarily for scientific experimentation and general sub-sea research, and capable of carrying both wet and dry payloads to a depth of 70m, RAUVER is an extremely useful tool. The modular design allows for versatile mission specific alterations to the submersible's configuration, making it possible for a wide range of experiments to be conducted.

During the evaluation of RECOVERY RAUVER was used in tethered autonomous mode, designed for ease of development of autonomous software. The autonomous computing platform stays on the surface where the developer can easily monitor and debug the software. Power and communications between the low-level RAUVER systems and the autonomous computing platform are transmitted down an umbilical, allowing the developer to monitor the autonomous software performance even when RAUVER is at depth. Using remotely supplied power also removes the constraint

Figure 6.2: RAUVER on Evaluation Trials in Crosswood Reservoir

of battery runtime during development; the developer can debug twenty-four hours per day if desired.

When the developer is happy with the performance of the autonomous software the autonomous computing platform is simply inserted into the vehicle and the power supply is switched from remote power to battery power. Untethered autonomous operational tests may then be performed.

6.2.1 RAUVER's Actual Fault Log

During the development of RAUVER a manual fault log was kept, it is shown below. Some of these real faults were later reproduced so that RECOVERY could be evaluated on actual problems. Unfortunately the majority of these failures happened either before RECOVERY's mission log storage routines were fully operational, or when RECOVERY was not running, and so mission logs of the actual faults at time of occurrence are not available.

6/5/00 Thruster Dropout

Fault: Design fault. Surge protection circuitry causing occasional undervoltage under high loading

Cure: circuit modified to reduce surge protection resistance (helped but still present occasionally)

7/5/00 No Comms with Vehicle

Fault: Onboard processor crash caused by bad software download whilst upgrading software

Cure: Manual reset

8/5/00 No Comms with Vehicle

Fault: Loss of vehicle to surface serial communications, traced to loose wire inside junction box

Cure: Connection tightened, junction box reassembled

12/5/00 Water Alarm

Fault: Slight leak found

Cure: Vehicle hull seals reassembled

19/5/00 Water Alarm

Fault: Slight leak found

Cure: Vehicle hull seals reassembled

6/6/00 Starboard Main Thruster Dead

Fault: Design fault. Wires pinched and shorting out against internal decking

. *(Note: This fault was recreated and used to evaluate RECOVERY)*

Cure: Power stack realigned to avoid pinching

14/6/00 No Comms with Vehicle

Fault: Onboard processor crash, cause unknown

Cure: Manual reset performed

22/6/00 No Comms with Vehicle

Fault: Onboard processor crash, cause unknown

Cure: Manual reset performed

13/7/00 Maintenance

Fault: No fault, scheduled upgrade of Power (Current) Sensors to instantaneous type

Cure: Not Applicable

25/7/00 Maintenance

Fault: No fault, scheduled upgrade of monochrome camera to colour

Cure: Not Applicable

25/7/00 Maintenance

Fault: No fault, Power (Current) sensors replaced with originals as couldn't read instantaneous sensors properly due to system limitations

Cure: Not Applicable

29/8/00 Water Alarm

Fault: Crumbled aspirin (used to trip a switch when water present)

Cure: Aspirin replaced

30/8/00 Water Alarm

Fault: Aspirin came loose

Cure: Aspirin replaced securely

1/9/00 Maintenance Fault: No fault, compass calibration work as still only able to use attitude sensing

Cure: Not Applicable

20/10/00 Maintenance Fault: Compass removed and returned to manufacturer as we were unable to calibrate compass

Cure: Not Applicable

2/11/00 Maintenance Fault: Refitted and calibrated compass

Cure: Not Applicable

7/11/00 Maintenance

Fault: Upgrade of temporary cure

Cure: Thruster wire pinch problem permanently solved by machining more space for the wires. Manual reset of processor also made easier by fitting of reset switch in port nosecone.

23/7/01 Water Alarm

Fault: Probably aspirin crumble - accept until overhaul - no deep operations - monitor vehicle trim manually

Cure: Accept for now

6.3 Evaluation Locations: Crosswood and Harperrig Reservoirs

The mission logs for evaluating RECOVERY were gained on missions performed in two local reservoirs, Crosswood and Harperrig. East of Scotland Water Plc were extremely helpful in granting us full access to these reservoirs.

Crosswood Reservoir is approximately 500 metres long, 700 metres wide and 8 metres deep. It has an intake tower sited approximately 20 metres from the bank. Crosswood Reservoir is shown in Figure 6.3

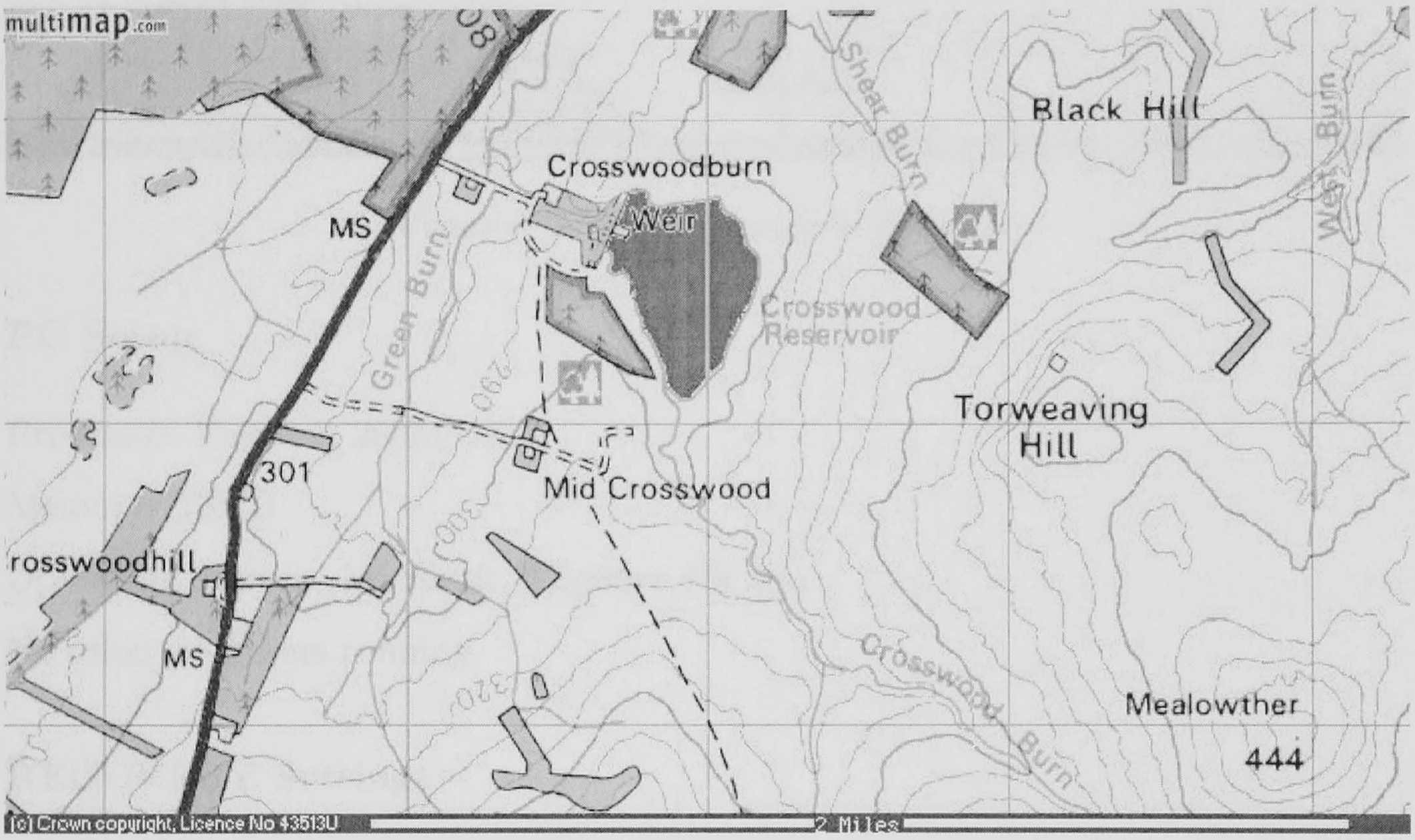


Figure 6.3: Map of Crosswood Reservoir

Harperrig Reservoir is approximately 1km long, 1.5km wide and 10 metres deep. It also has an intake tower sited approximately 20 metres from the bank, as shown in Figure 6.4

6.4 Experiment Setup

For all experiments in this chapter (unless otherwise stated) the setup was:

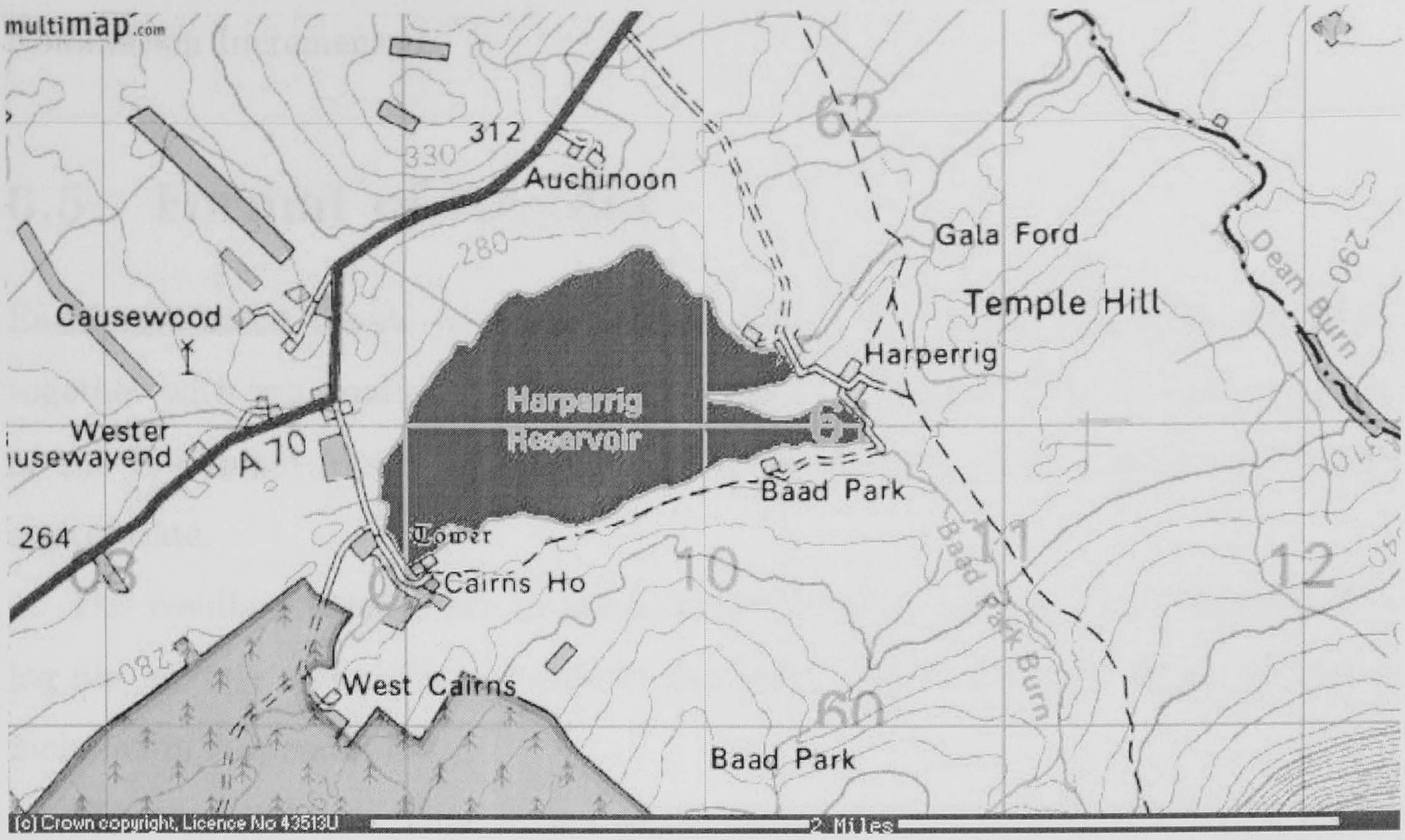


Figure 6.4: Map of Harperrig Reservoir

PC Setup

Processor: Pentium 200MMX

Memory: 32MB

Operating System: Microsoft Windows NT 4.0

No other programs running

RECOVERY Settings

Command and status polling frequency: 5Hz

Residual detection window: ±8.0

Domain-Independent Diagnostics:

Hierarchy Module nomination threshold: 0.6

Correlator Delta window: ±1.0

Correlator Recent window:

Inactive for at least 8 seconds before failure

Activated within 3 seconds of failure

Model-Based Diagnosis nomination windows:

Roll Model: ±1.5 Degrees

Power Models: ±0.5 Amps

Nomination Increment: 1

6.5 Format of Results

Each experiment begins with a short description of the background to the fault, together with information on normal and faulty operation. The relevant segment of the Relational Model is given, showing normal and faulty information where appropriate.

The results presented are in the form of verbatim output from RECOVERY's log file, slightly edited for inclusion in this thesis. The full text of these log files is included in Appendix E

The term 'Mission Time' refers to the time since the vehicle was last powered up, it is used to timestamp all data and log files. The full text output as shown in the appendices also has a date and time stamp generated by the host PC, which is used for debugging and system monitoring purposes, but this is not shown in the results.

6.6 Evaluation of Domain Independent Diagnostics: The Hierarchy Module

This section evaluates the Domain Independent Diagnostic Hierarchy Module, specifically the source/sink methods as detailed in Section 5.5. Real mission log files are used for the evaluation, with various parameters modified to mimic the failure of individual source components, in this case various RAUVER power supplies.

There are six experiments, each of which involves failing an individual power supply. Each experiment consists of the following steps:

1. Modify a mission log file to mimic the failure of a particular supply.
2. Run RECOVERY as normal until the fault is detected.
3. Run a single iteration of the Domain Independent Hierarchy Module, with no other diagnostic tools invoked.

The effectiveness of the diagnostic technique is determined by correctness of

diagnosis.

6.6.1 Experiment: Faulty 5Vc Power Supply

The 5Vc power supply component supplies power to the 8 digital alarms, the AOSI Compass/Attitude sensor and the 3 rate gyros. When the 5Vc supply fails all the alarms, which are active low, appear to be tripped and the output of the rate gyros drops to zero. The relevant section of the Relational Model during 5Vc supply failure is shown in Figure 6.5, where the shading of the Alarm parameters indicate that they are active. The zero output of the gyros is also indicated.

At present the Roll, Pitch and Yaw rate gyros do not have any detection information or models associated with them and so RECOVERY only detects the alarms:

```
Alarm detected: ALARM_WATER at Mission Time: 211.615
Alarm detected: ALARM_PDH at Mission Time: 211.615
Alarm detected: ALARM_SDH at Mission Time: 211.615
Alarm detected: ALARM_PDW at Mission Time: 211.615
Alarm detected: ALARM_SDW at Mission Time: 211.615
Alarm detected: ALARM_PLH at Mission Time: 211.615
Alarm detected: ALARM_SLH at Mission Time: 211.615
Alarm detected: ALARM_SPARE at Mission Time: 211.615
```

'Faulty' parameters automatically have their associated sensor component (specified in the Relational Model) nominated as a possible cause of the fault, and so at detection the Suspicion Index looks like:

```
Suspicion Index at Mission Time: 211.615
Component Level Diagnostic Iteration: 0
PORT_DIODE_50_SENSOR: 1
STBD_DIODE_50_SENSOR: 1
PORT_DIODE_60_SENSOR: 1
STBD_DIODE_60_SENSOR: 1
PORT_NAVSUPPLY_HOT_SENSOR: 1
```

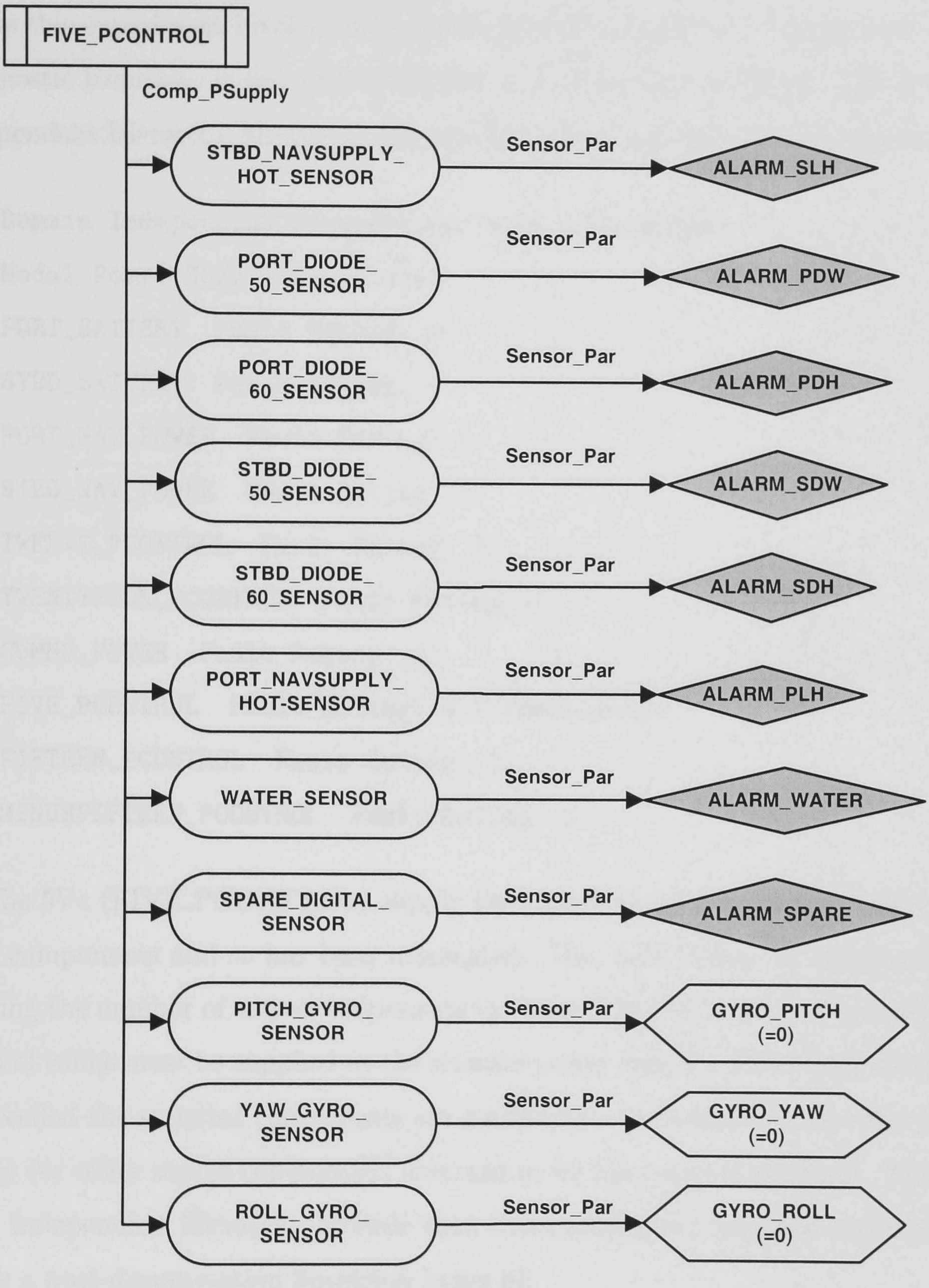


Figure 6.5: The 5Vc Relational Model Segment during Power Supply Failure

```
STBD_NAVSUPPLY_HOT_SENSOR: 1
WATER_SENSOR: 1
SPARE_DIGITAL_SENSOR: 1
```

As this experiment involves only one diagnostic iteration the 'Component Level Diagnostic Iteration: 0' line shows that this is the first (zero indexed). The Domain Independent Hierarchy Module is now invoked, providing the following output:

```
Domain Independent Diagnostics (post iterations)
Nodal Power Supply Diagnosis:
PORT_BATTERY  Fault Rating: 0.
STBD_BATTERY  Fault Rating: 0.
PORT_NAV_POWER  Fault Rating: 0.
STBD_NAV_POWER  Fault Rating: 0.
TWELVE_PCONTROL  Fault Rating: 0.
TWENTYFOUR_PCONTROL  Fault Rating: 0.
COMMS_POWER  Fault Rating: 0.
FIVE_PCONTROL  Fault Rating: 0.7 (nominated)
FIFTEEN_PCONTROL  Fault Rating: 0.
MINUSFIFTEEN_PCONTROL  Fault Rating: 0.
```

The 5Vc (FIVE_PCONTROL) supply has exceeded the threshold of faulty supplied components and so has been nominated. The 'fault rating' is determined by dividing the number of active components by the number of nominated components (both of which must be supplied by the relevant power supply). If the fault threshold is exceeded the supplied components are assumed to be functioning and the power supply (or other source component) assumed to be the cause of the fault. The Domain Independent Hierarchy Module then denominates the supplied components, giving a post-denomination Suspicion Index of:

```
Suspicion Index at Mission Time: 211.615
Component Level Diagnostic Iteration: 0
FIVE_PCONTROL: 1
```

This is a correct diagnosis. The hierarchy module has correctly found that a common component is faulty despite the power supply having no direct measurement, and without any fault-specific information at all. This shows that domain-independent, generalised fault information can be extremely powerful when used with RECOVERY.

6.6.2 Experiment: Faulty 12Vc Power Supply

The 12Vc power supply component supplies power to the depth sensor, the AOSI compass module and the port and starboard analogue temperature sensors, as shown in Figure 6.6.

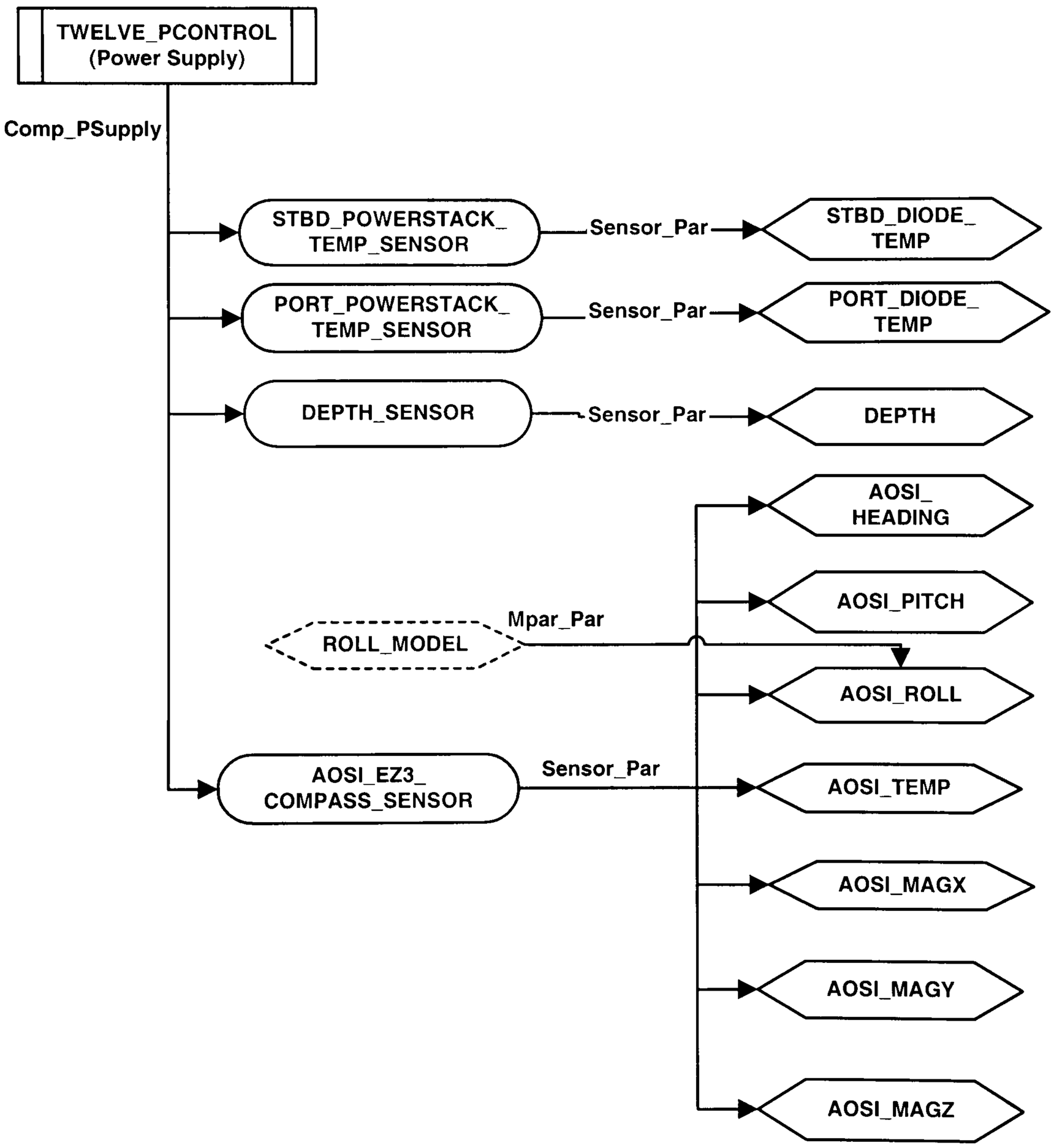


Figure 6.6: The 12Vc Relational Model Segment

On failure of the 12Vc power supply the depth sensor output drops to zero, as do the port and starboard analogue temperature sensors. The implementation of the AOSI compass module, which communicates data via an RS232 Serial Communications link to the onboard computer, means that when it fails no more data is received from it. In this event the onboard computer simply retains the latest values, which do not change until another update is received from the AOSI compass module.

The only detection associated with components supplied by the 12Vc power supply is the modelling of the AOSI Roll parameter. There are no models of the depth, pitch and yaw parameters due to developmental time constraints.

The narrow spread of detection information means that the fault can only be detected when the vehicle is rolling as the observed roll will not match the sensed roll, which will be the same value as when the 12Vc power supply failed. When the vehicle is sitting flat in the water, whether it be motionless, ascending, descending or turning, there will be no detection. Without detection the diagnosis level is not invoked.

12Vc Power Supply Failure when Vehicle Rolling

The 12Vc power supply was failed when the vehicle was rolling so that a fault would be detected. The Relational Model segment in Figure 6.7 shows the detected fault.

This time the RECOVERY detection module picks up the discrepancy between the vehicle's modelled roll and the frozen sensed roll parameter:

```
Residual detected between Modelled Node: ROLL_MODEL
and Sensed Node: AOSI_ROLL at Mission Time: 253.88
```

The associated sensor is nominated, so at detection the Suspicion Index is:

```
Suspicion Index at Mission Time: 253.88
Component Level Diagnostic Iteration: 0
AOSI_EZ3_COMPASS_SENSOR: 1
```

The Domain Independent Hierarchy Module is now invoked, giving the output:

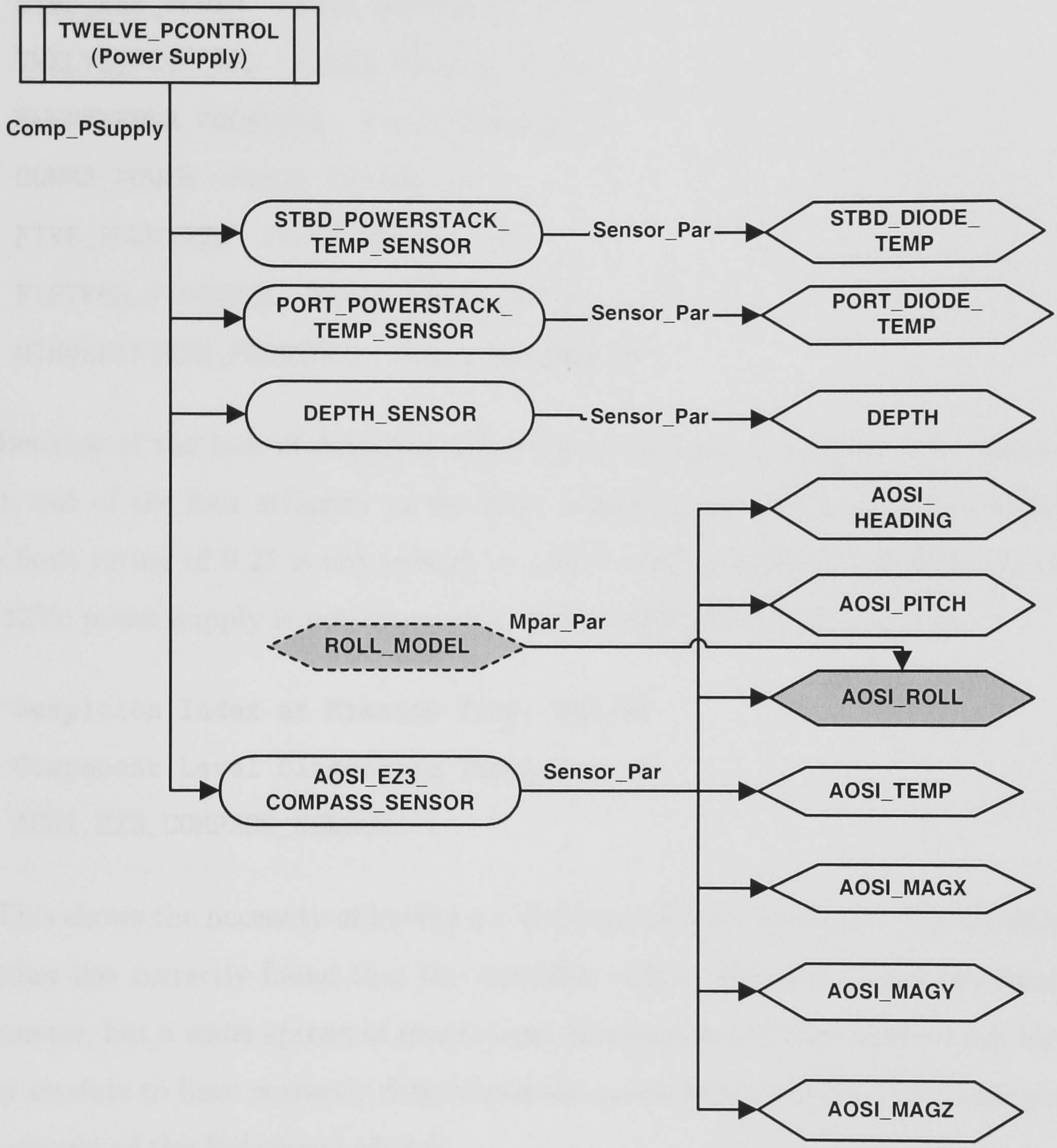


Figure 6.7: The 12Vc Relational Model Segment during Power Supply Failure

```
Domain Independent Diagnostics (post iterations)
Nodal Power Supply Diagnosis:
PORT_BATTERY  Fault Rating: 0.
STBD_BATTERY  Fault Rating: 0.
PORT_NAV_POWER  Fault Rating: 0.
STBD_NAV_POWER  Fault Rating: 0.
TWELVE_PCONTROL  Fault Rating: 0.25
TWENTYFOUR_PCONTROL  Fault Rating: 0.
COMMS_POWER  Fault Rating: 0.
FIVE_PCONTROL  Fault Rating: 0.
FIFTEEN_PCONTROL  Fault Rating: 0.
MINUSFIFTEEN_PCONTROL  Fault Rating: 0.
```

Because of the lack of detection information the only component that shows a fault, out of the four attached to the 12Vc supply, is the AOSI compass module. The fault rating of 0.25 is not enough to exceed the nomination threshold and so the 12Vc power supply is not nominated, giving a final Suspicion Index of:

```
Suspicion Index at Mission Time: 253.88
Component Level Diagnostic Iteration: 0
AOSI_EZ3_COMPASS_SENSOR: 1
```

This shows the necessity of having a wide scope of fault detection. The Residual Watcher has correctly found that the modelled output does not match the sensed parameter, but a wider spread of simple fault detection would have enabled the hierarchy module to have correctly determined the power supply fault. This vindicates the concept of the Relational Model.

6.6.3 Experiment: Faulty ±15Vc Power Supply

The ±15Vc power supply is a dual power supply, i.e. it is a single component that provides two power rails. It is represented in the Relational Model as two separate components because it is possible for one power rail to fail while the other remains working. This could be caused by a loose wire, blown fuse etc. See Figure 6.8

The Port and Starboard Navigation Power Supply current (amperage) sensors, commonly known as 'Lems', require a dual supply and so are connected to both the plus and minus 15Vc. At this time no other components use these power rails.

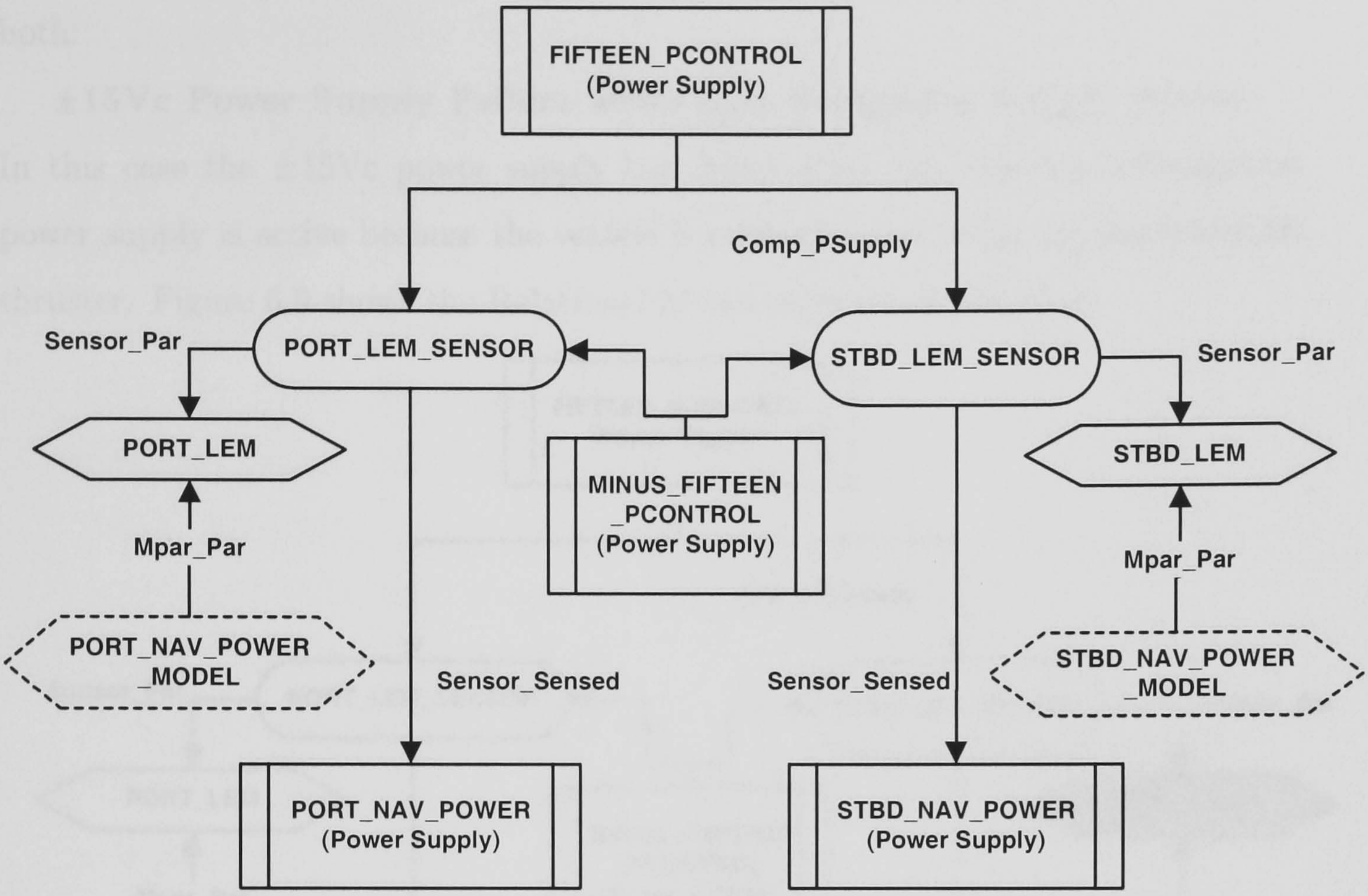


Figure 6.8: The ±15Vc Relational Model Segment

This test assumes that the entire power supply unit fails, in which case both the plus and minus fifteen volt power rails drop to zero. When this happens the output of both Lems also drop to zero.

Both the Lems have detection information attached in the form of the port and starboard power models (which actually only model electric current).

When the current drawn from the Navigation power supplies is zero the output from the Lems is also zero. If the ±15Vc power supply fails, forcing the Lem outputs to zero, when no current is being drawn from the Navigation power supplies there will be no detection. This is because the modelled output is zero, the correct output is zero and the Lem output is zero.

±15Vc Power Supply Failure when Vehicle Moving

If the ±15Vc power supply fails when the vehicle is moving then the fault will be detected. There are two conditions: one Navigation power supply being used, or both:

±15Vc Power Supply Failure when One Navigation Supply Active

In this case the ±15Vc power supply has failed when the Starboard Navigation power supply is active because the vehicle is rolling to port using the starboard lift thruster. Figure 6.9 shows the Relational Model segment at this time.

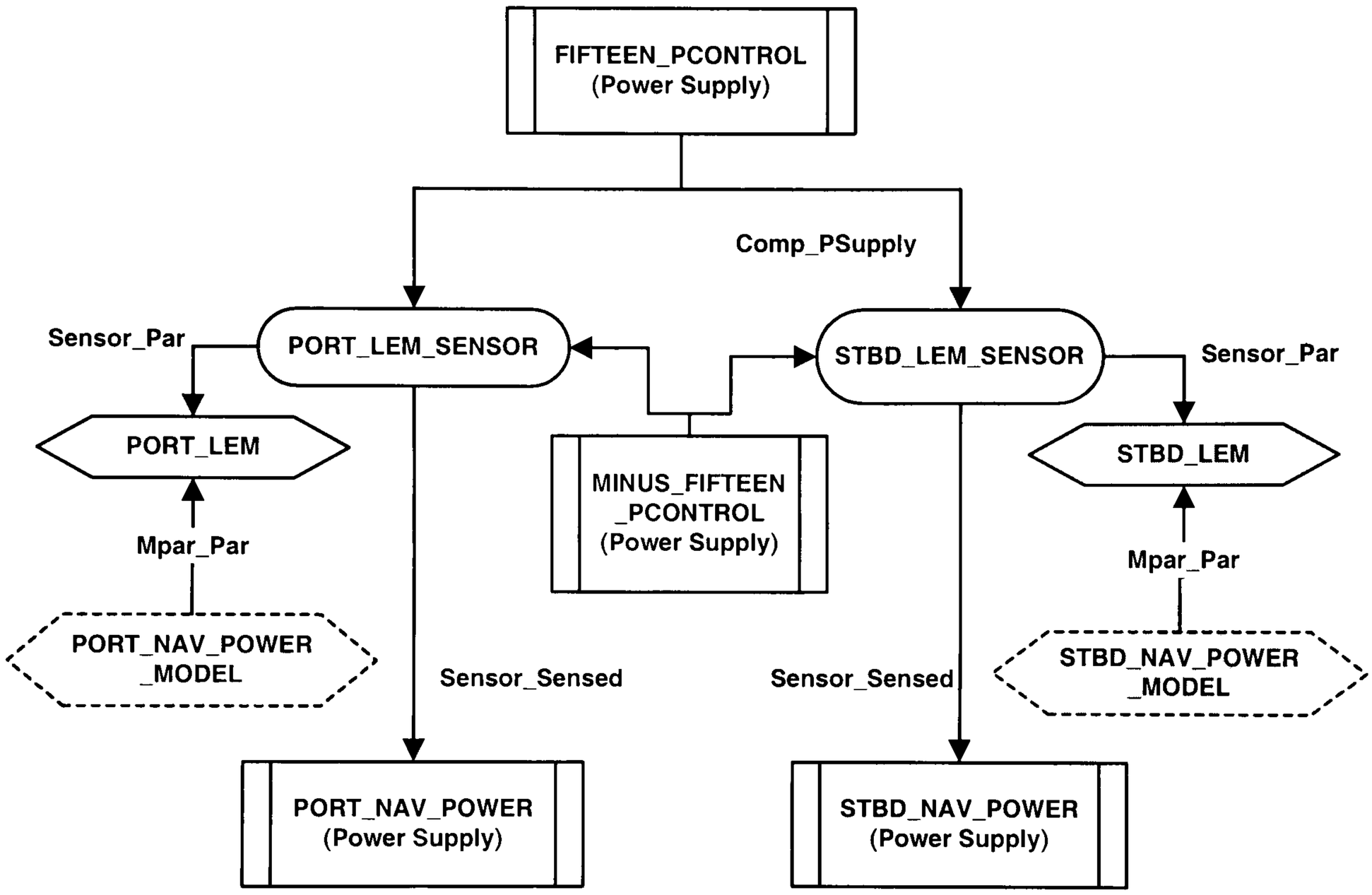


Figure 6.9: The ±15Vc Relational Model Segment during Power Supply Failure with a Single Active Navigation Supply

The discrepancy between modelled and sensed Starboard Navigation power supply current is detected:

```
Residual detected between Modelled Node:
STBD_NAV_POWER_MODEL
and Sensed Node: STBD_LEM at Mission Time: 252.33
```

The 'faulty' parameter has its attached sensor component nominated automatically, so at detection the Suspicion Index is:

```
Suspicion Index at Mission Time: 253.33
Component Level Diagnostic Iteration: 0
STBD_LEM_SENSOR: 1
```

The Domain Independent Hierarchy Module is invoked, providing the output:

```
Domain Independent Diagnostics (post iterations)
Nodal Power Supply Diagnosis:
PORT_BATTERY  Fault Rating: 0.
STBD_BATTERY  Fault Rating: 0.
PORT_NAV_POWER  Fault Rating: 0.
STBD_NAV_POWER  Fault Rating: 0.
TWELVE_PCONTROL  Fault Rating: 0.
TWENTYFOUR_PCONTROL  Fault Rating: 0.
COMMS_POWER  Fault Rating: 0.
FIVE_PCONTROL  Fault Rating: 0.
FIFTEEN_PCONTROL  Fault Rating: 0.5
MINUSFIFTEEN_PCONTROL  Fault Rating: 0.5
```

Both the plus and minus fifteen volt power rails have a fault rating, reflecting the dual-supply requirements of the Lems, but neither of them have reached the nomination threshold. This is because with only one Navigation power supply active only one of the Lem outputs is incorrect, the other is still 'correctly' stuck at zero. The final Suspicion Index is:

```
Suspicion Index at Mission Time: 253.33
Component Level Diagnostic Iteration: 0
STBD_LEM_SENSOR: 1
```

Again, a simple increase in the scope of fault detection would have caused a fully correct diagnosis, showing the usefulness of RECOVERY's Relational Model.

±15Vc Power Supply Failure when Both Navigation Supplies Active

With both Navigation power supplies active the fault is detected, showing up in the Relational Model (Figure 6.10) as both Port and Starboard Navigation power parameters not matching their related model parameters.

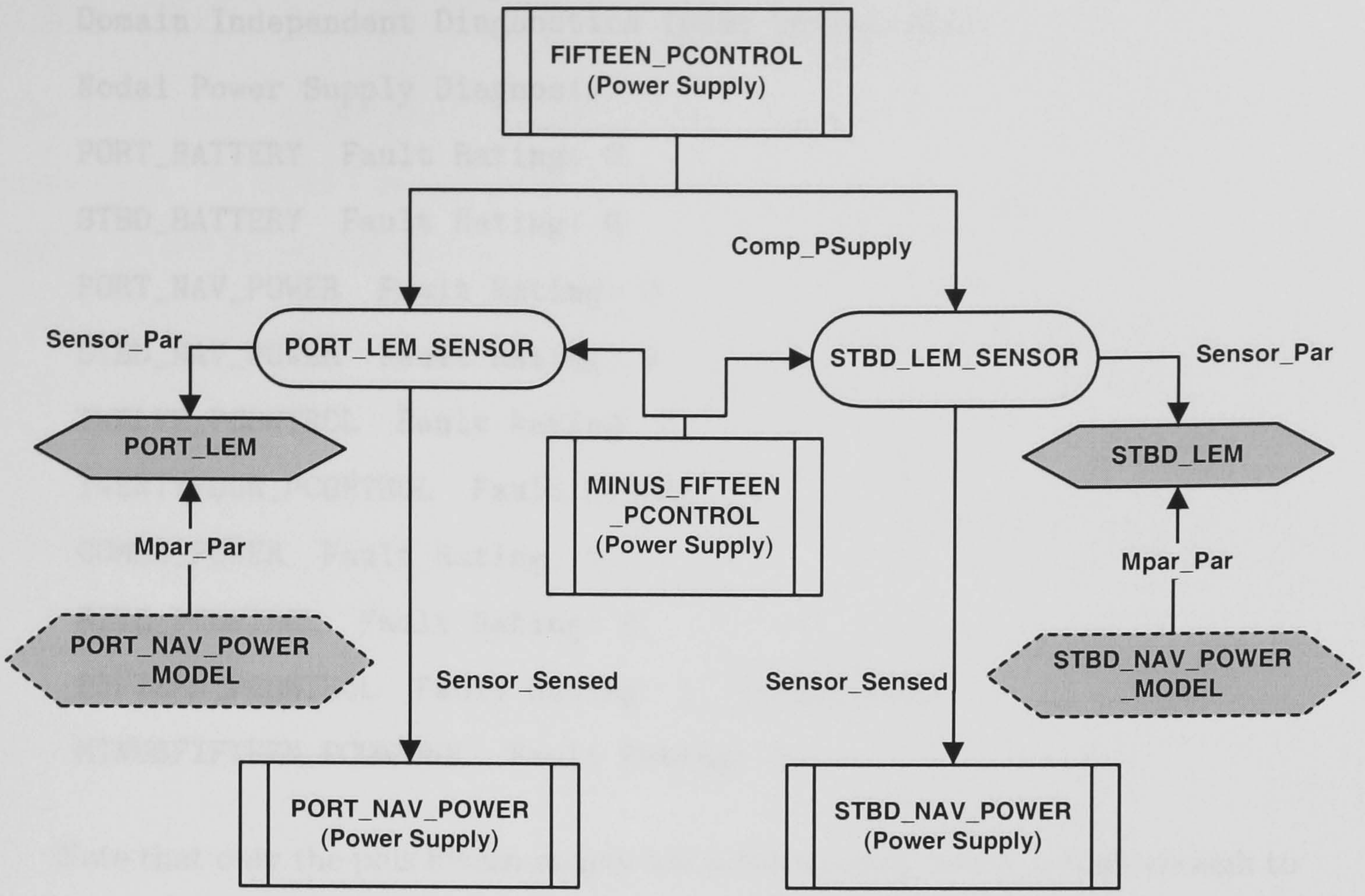


Figure 6.10: The ±15Vc Relational Model Segment during Power Supply Failure with Both Navigation Supplies Active

The detection information is:

```
Residual detected between
Modelled Node: PORT_NAV_POWER_MODEL
and Sensed Node: PORT_LEM at Mission Time: 31.48
Residual detected between
Modelled Node: STBD_NAV_POWER_MODEL
and Sensed Node: STBD_LEM at Mission Time: 31.48
```

Associated sensors components are automatically nominated, giving a Suspicion Index at detection of:

```
Suspicion Index at Mission Time: 31.48
Component Level Diagnostic Iteration: 0
PORT_LEM_SENSOR: 1
STBD_LEM_SENSOR: 1
```

The Domain Independent Hierarchy Module is invoked:

```
Domain Independent Diagnostics (post iterations)
Nodal Power Supply Diagnosis:
PORT_BATTERY  Fault Rating: 0.
STBD_BATTERY  Fault Rating: 0.
PORT_NAV_POWER  Fault Rating: 0.
STBD_NAV_POWER  Fault Rating: 0.
TWELVE_PCONTROL  Fault Rating: 0.
TWENTYFOUR_PCONTROL  Fault Rating: 0.
COMMS_POWER  Fault Rating: 0.
FIVE_PCONTROL  Fault Rating: 0.
FIFTEEN_PCONTROL  Fault Rating: 1. (nominated)
MINUSFIFTEEN_PCONTROL  Fault Rating: 0.
```

Note that only the plus fifteen supply has a fault rating, which is high enough to exceed the nomination threshold, but the MINUSFIFTEEN_PCONTROL (-15Vc power rail) should also have a fault rating.

The Hierarchy module automatically denominates supplied components as soon as the power supply is nominated. Each power supply is checked in turn, and the minus fifteen supply is checked after the plus fifteen so there are no nominated components attached to the minus fifteen at time of checking. This is an error in the operation of the Hierarchy Module, which should be modified to only denominate after all power supplies have been checked. The final Suspicion Index is:

```
Suspicion Index at Mission Time: 31.48
Component Level Diagnostic Iteration: 0
FIFTEEN_PCONTROL: 1
```

This is a fully correct diagnosis as the correct (dual) power supply has been nominated, even though it is not directly sensed and no fault-specific information was used. The novel domain-independent Hierarchy module has worked well, using the structured Relational Model to find common connections between supposedly faulty components. A specific fault has been found using general diagnostic information.

6.6.4 Experiment: Faulty 24Vc Power Supply

The 24Vc power supply provides power to the control circuitry of the four thruster controllers. No sensors are powered by this supply. Figure 6.11 shows the relevant section of the Relational Model.

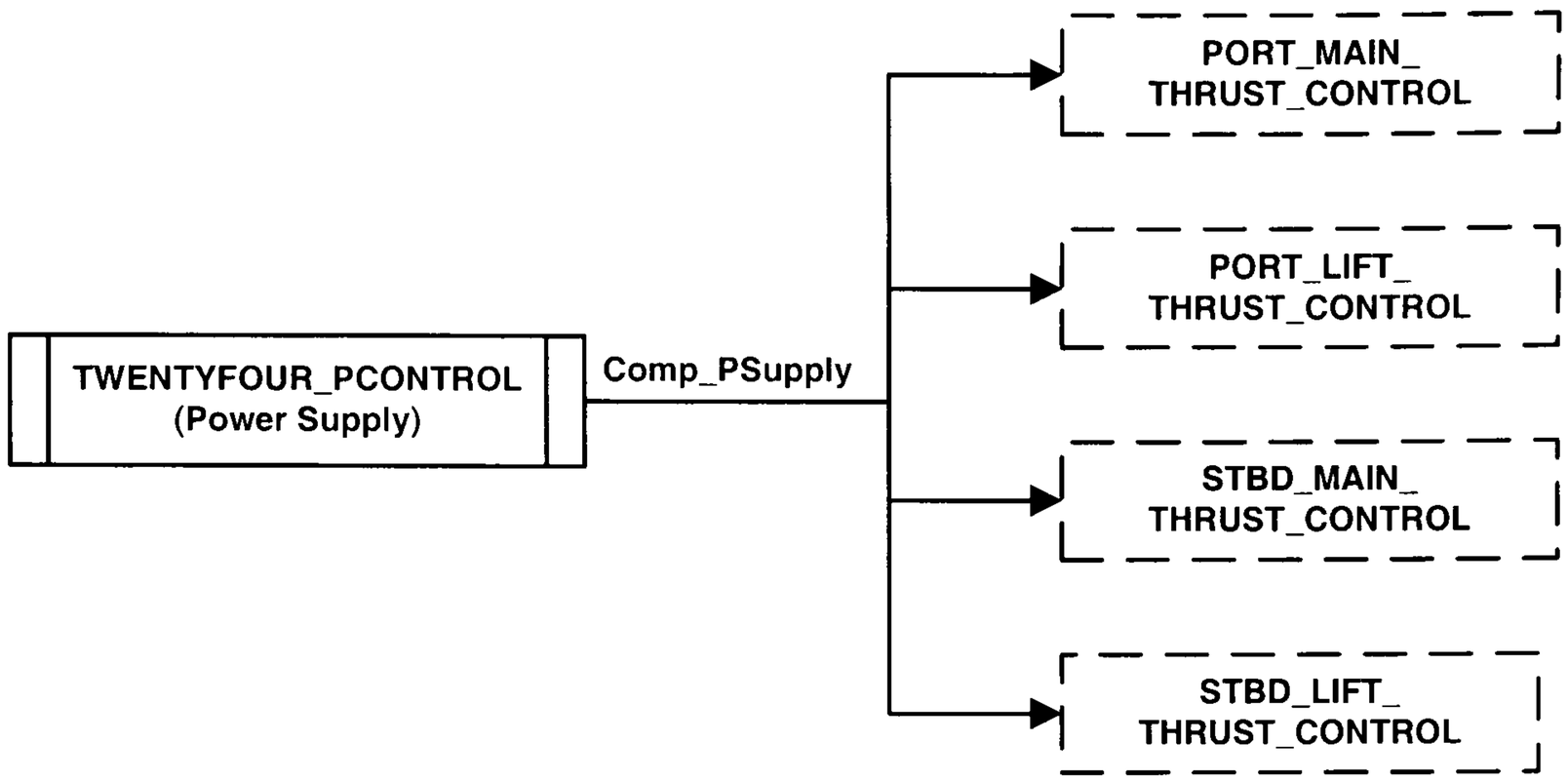


Figure 6.11: The 24Vc Relational Model Segment

When the 24Vc power supply fails all the thruster controllers fail and so no power is sent to the thruster motors. The vehicle can no longer move. When the vehicle is not supposed to be moving no fault will be detected as the thrusters are not being used and no sensors are supplied by this power supply.

24Vc Power Supply Failure when Vehicle Moving

The limited detection information inserted into the system means that this fault is indistinguishable from the failure of the ±15Vc power supply. With the present sensor, modelling and detection suite the failure of the thrusters (caused by the 24V power supply) is indistinguishable from the failure of the movement sensors.

As this is effectively the same as the 15vc evaluation with both lems the experiment was counted as being identical but this time the diagnosis is incorrect.

6.6.5 Experiment: Faulty Port Navigation Power Supply

The Port Navigation power supply provides power to the Port Main-Thruster, the Port Lift-Thruster and the Port Lamp, as shown in Figure 6.12.

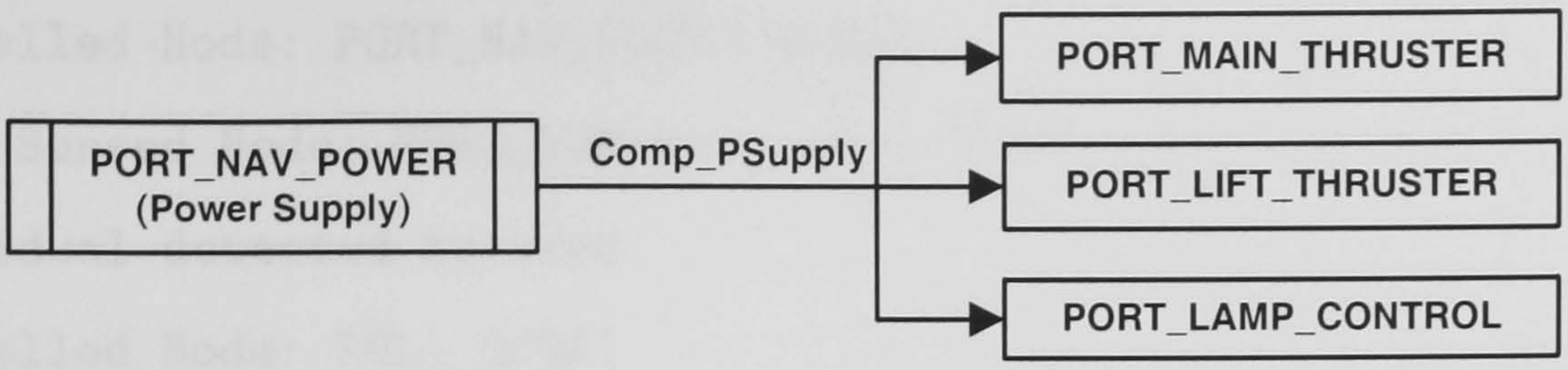


Figure 6.12: Components Supplied by the Port Navigation Power Supply

When the supply fails there is no power available to drive the port thrusters or the port lamp and so the vehicle's movement abilities are severely limited. As with the previous power supply tests if the Port Navigation power supply is not active then there will be no detection as there are no sensors directly measuring the supply voltage.

Because of the slow dynamic response of the vehicle's physical movement there is usually a delay between power and movement problems being detected. This is the reason that the full RECOVERY system runs over several iterations. In the current RECOVERY implementation and during this experiment the Domain Independent Hierarchy Module is run during only a single iteration, as discussed in Section 5.5.3. The single iteration is invoked at a time when faults are detected in both the power and movement dimensions.

Supply Failure when Vehicle Moving: Power and Movement Detected

At this time residuals are apparent with both the Roll and Port Navigation Power parameters, as shown in Figure 6.13.

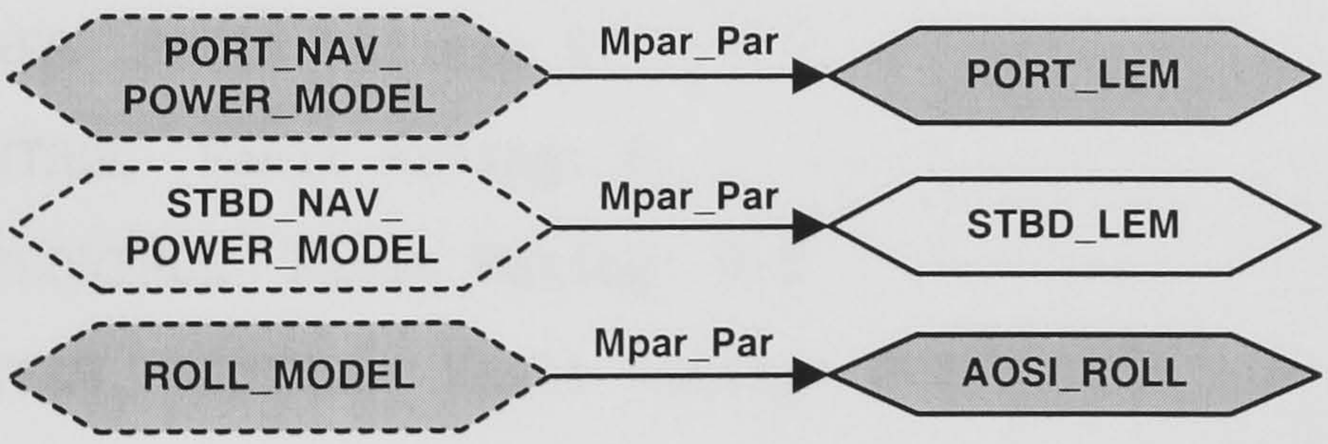


Figure 6.13: Residuals Generated in the Port Navigation Relational Model Segment during Power Supply Failure

The detection information is:

```
Residual detected between
```

```
Modelled Node: PORT_NAV_POWER_MODEL
and Sensed Node: PORT_LEM at Mission Time: 440.535
Residual detected between
Modelled Node: ROLL_MODEL
and Sensed Node: AOSI_ROLL at Mission Time 440.535
```

The associated sensor components are automatically nominated, so at detection the Suspicion Index is:

```
Suspicion Index at Mission Time: 440.535
Component Level Diagnostic Iteration: 0
PORT_LEM_SENSOR: 1
AOSI_EZ3_COMPASS_SENSOR: 1
```

The Domain Independent Hierarchy Module is invoked, producing the output:

```
Domain Independent Diagnostics (post iterations)
Nodal Power Supply Diagnosis:
PORT_BATTERY  Fault Rating: 0.
STBD_BATTERY  Fault Rating: 0.
PORT_NAV_POWER  Fault Rating: 0.
STBD_NAV_POWER  Fault Rating: 0.
TWELVE_PCONTROL  Fault Rating: 0.25
TWENTYFOUR_PCONTROL  Fault Rating: 0.
COMMS_POWER  Fault Rating: 0.
FIVE_PCONTROL  Fault Rating: 0.
FIFTEEN_PCONTROL  Fault Rating: 0.5
MINUSFIFTEEN_PCONTROL  Fault Rating: 0.5
```

The TWELVE_PCONTROL has a fault rating of 0.25 as it supplies four components, one of which is the AOSI compass sensor, which senses Roll. The FIFTEEN and MINUS_FIFTEEN_PCONTROL power supplies have a fault rating of 0.5 as they both supply both Lems, of which the Starboard Lem sensor has been nominated.

The supply that is actually faulty (PORT_NAV_POWER) has a zero fault rating, despite the fact that 3 components supplied by it are not working. This is because the supplied components do not have any sensed parameters or detection information associated with them.

None of the supplies have reached the required nomination threshold. The final Suspicion Index is:

fig: num hits post denomination

```
Suspicion Index at Mission Time: 440.535
Component Level Diagnostic Iteration: 0
PORT_LEM_SENSOR: 1
AOSI_EZ3_COMPASS_SENSOR: 1
```

Again, this would have been a correct diagnosis if the scope of simple detection information had been extended.

6.6.6 Experiment: Faulty Starboard Navigation Power Supply

This fault and experiment is identical to the Port Navigation power supply fault, but it is the matching Starboard Navigation power supply. Because of this only the final nomination index is shown:

```
Suspicion Index at Mission Time: 253.61
Component Level Diagnostic Iteration: 0
STBD_LEM_SENSOR: 1
AOSI_EZ3_COMPASS_SENSOR: 1
```

Again, this would have been a correct diagnosis if the scope of simple detection information had been extended.

6.7 Evaluation of Domain Independent Diagnostics: The Correlator

The current implementation of the Correlator uses items of domain independent diagnostic knowledge, as detailed in Section 5.5. Experiments are performed in this section in order to show the action of each piece of knowledge.

For these experiments real mission files were used without modification.

6.7.1 Experiment: Null Movement with Thermal Drift

This mission was conducted for another Ocean Systems Laboratory project which required accurate magnetic heading data. RAUVER was positioned so that it was within 1 degree of being absolutely flat then left recording data whilst motionless for approximately 15 minutes whilst a mission log file was generated.

When the data was evaluated the magnetic heading was found to drift upwards by approximately 1 degree every ten minutes. On further inspection of the data it was found that the temperature of the compass module, which is self-measured, also drifted up during the mission. The conclusion was that the magnetic heading variation was a thermal drift.

This type of correlation between two parameters is exactly what the Correlator is designed to look for using the 'parameters that track a faulty parameter are likely to be related to the fault' item of domain independent diagnostic information. When RECOVERY was run on this mission it was predicted that it would notice the correlation between sensor drift and temperature increase.

In order for this interesting mission to be used for evaluating RECOVERY a 'false' heading model was implemented, which simply output a fixed value equal to the magnetic heading reading at the start of the mission. When the magnetic heading reading increased above a certain amount, equal to a reading near the end of the mission, the residual was detected automatically using RECOVERY's methods and a single diagnostic iteration was invoked, consisting of the Correlator only.

RECOVERY successfully detected the residual between predicted and observed heading:

```
Residual detected between
Modelled Node: HEADING_MODEL
and Sensed Node: AOSI_HEADING
```

The faulty parameter's related sensor was automatically nominated, giving a Suspicion Index at detection of:

```
Suspicion Index at Mission Time: 689.5
Component Level Diagnostic Iteration: 0
AOSI_EZ3_COMPASS_SENSOR: 1
```

The Correlator was then invoked, producing the output:

```
Correlator Invoked at Mission Time: 689.5
Delta_Correlations:
Bad_Node: AOSI_HEADING
Delta_Matches:
GYRO_YAW
```

The final Suspicion Index was:

```
Component Level Diagnostic Iteration: 0
YAW_GYRO_SENSOR: 1
AOSI_EZ3_COMPASS_SENSOR: 1
```

The Correlator has failed to notice the 'obvious' correlation between magnetic heading drift and temperature increase. Why?

Figure 6.14 shows the magnetic heading, which can be seen to drift upwards over the length of the mission. There is also a discontinuity at approximately 120 seconds, where the heading jumps down to just over 284.5 degrees before resuming its upward climb. The reason for this discontinuity is unknown.

Figure 6.15 shows the compass sensor's temperature superimposed on the magnetic heading. This highlights several key factors: the temperature information is sampled at a lower frequency than the heading, the temperature seems to be increasing in a near-linear manner but the heading is non-linear, the heading is far more noisy than the temperature.

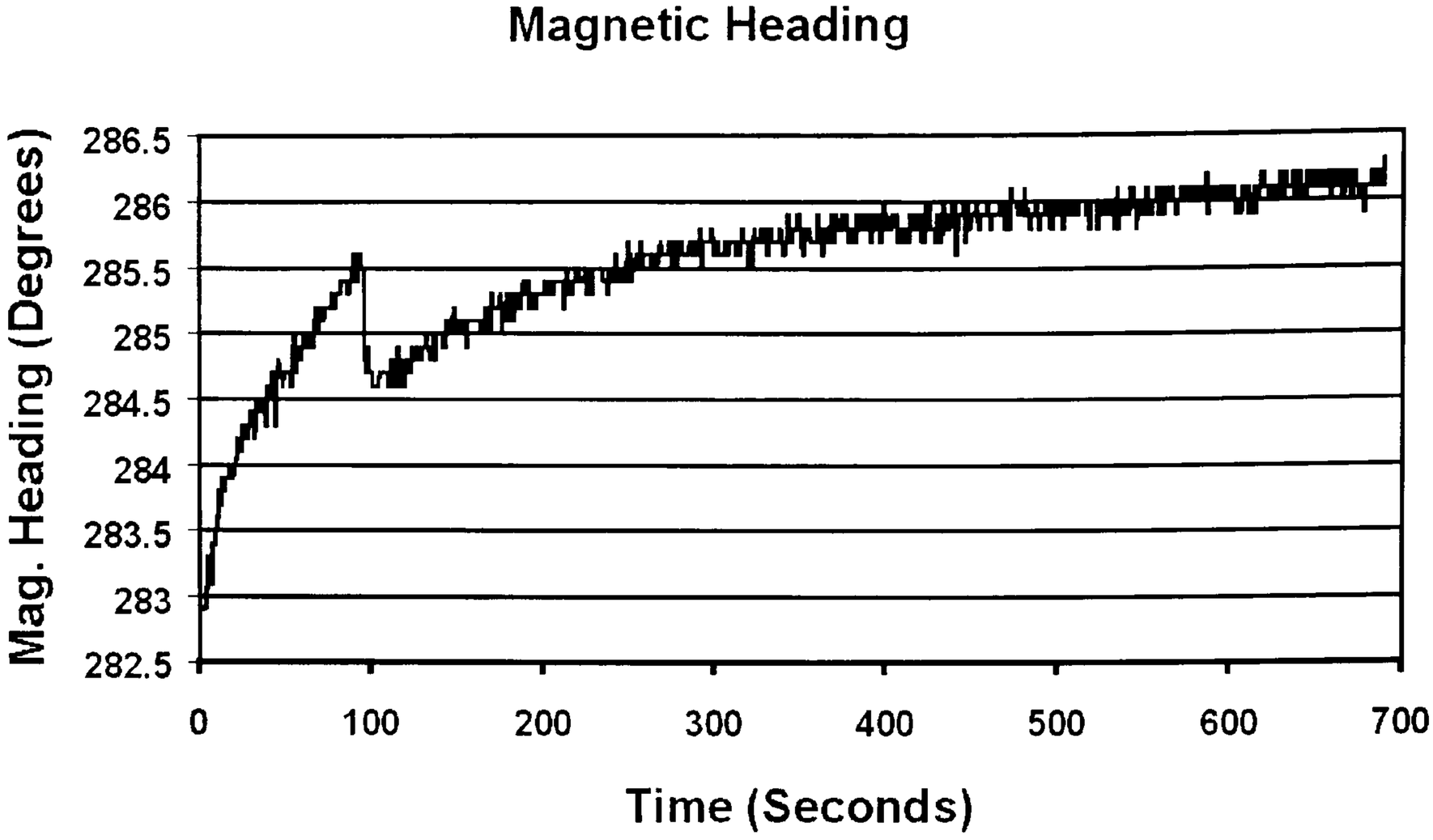


Figure 6.14: Magnetic Heading during the Null Movement Mission

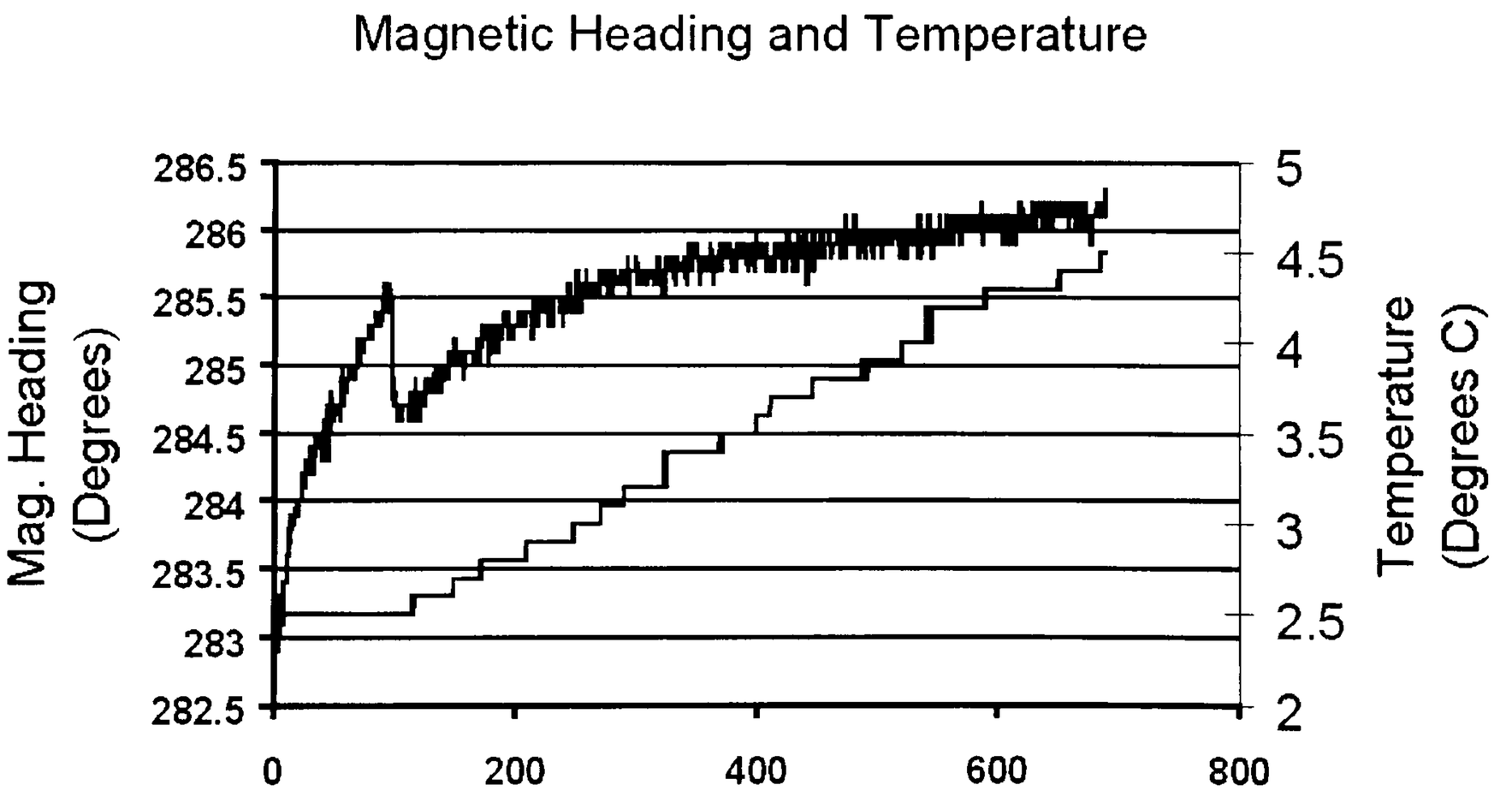


Figure 6.15: Magnetic Heading and Sensor Temperature during the Null Movement Mission

But still, the overall trend of both parameters is upward.

The implementation of the search for 'parameters that track faulty parameters' was kept as simple as possible, both to meet developmental time constraints and to fit in with the 'simplest, crudest, worst case' approach to RECOVERY development. The search consists of generating a Delta Index for each parameter and then comparing it to the faulty parameter's Delta Index.

The Delta Index starts at zero. At each time interval it is incremented by one if the parameter has increased, decremented if the parameter has decreased and not modified if the parameter has not changed. If a parameter has increased at every sample during the mission then the Delta Index will also have climbed constantly.

The Delta Indices are then compared to the faulty parameter, any that are close in value are nominated as a 'tracking parameter'. This is a very crude search that takes no account of different rates of change, delayed tracking or different waveform shapes.

At the time of detection the Delta Indices for the relevant parameters were:

AOSI_HEADING: 12.0

AOSI_TEMP: 18.0

GYRO_YAW: 14.0

The nomination threshold at this time was 2.0, which had been set during development to a value which generally limited the number of parameters nominated to about 3 to prevent flooding the system with irrelevant data.

The nomination threshold was increased to 6.0 to enable the Correlator to nominate the AOSI_TEMP parameter and the test was repeated. The detection information was identical but this time the Correlator output was:

```
Correlator Invoked at Mission Time: 689.5
Delta_Correlations:
Bad_Node: AOSI_HEADING
Delta_Matches:
AOSI_PITCH
AOSI_TEMP
GYRO_YAW
```

```
GYRO_PITCH
```

The widening of the nomination window has enabled the nomination of the temperature parameter, but has also included other parameters. The final Suspicion Index was:

```
Suspicion Index at Mission Time: 689.5
Component Level Diagnostic Iteration: 0
AOSI_EZ3_COMPASS_SENSOR: 3
YAW_GYRO_SENSOR: 1
PITCH_GYRO_SENSOR: 1
```

The AOSI compass sensor has been nominated three times: once at detection for sensing the faulty parameter and twice by the Correlator for being attached to the AOSI pitch and temperature parameters. The two gyros have only been nominated once each by the Correlator for sensing their respective nominated parameters.

The Correlator is shown to be capable of automatically picking up extremely useful correlations using its general diagnostic information. For a human to have extracted these correlations out of a large mission file would have required quite some work, but the Correlator has automatically detected them. This fully vindicates the use of Domain-Independent Diagnostics in conjunction with the Relational Model.

6.7.2 Experiment: Component Fails on Startup

The Full System Evaluation includes the failure of a component at startup and so the Correlator output for this experiment is taken from the Starboard Lift-Thruster failure test. During this test the lift-thrusters have been inactive for several seconds before being activated, at which point the Starboard Lift-Thruster fails. The lack of power consumption is detected by comparing the Starboard Navigation Power Model with the observed power. The detection information is:

```
Residual detected between Modelled Node: STBD_NAV_POWER_MODEL
and Sensed Node: STBD_LEM at Mission Time: 439.735
```

The Correlator is invoked, producing the output:

```
Correlator Invoked at Mission Time: 439.735
Delta_Correlations:
Bad_Node: STBD_LEM
Delta_Matches:
PORT_LIFT
STBD_LIFT
AOSI_MAGX
Recent_Comp_Correlations:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER
```

Whilst not strictly relevant to this test, the Correlator's Delta Index search has determined that the lift thrusters have a similar Delta Index to the current drawn by the Starboard Navigation Power Supply (sensed by the Starboard Lem). During this mission the lift thrusters were run at matching thrusts until failure and they were the only components drawing power from the navigation supplies. The Correlator has correctly noticed a relevant link between components.

The AOSI_MAGX parameter has also been nominated during the Delta Index search. It is probable that this is a coincidence as it should not be affected by operation of the lift thrusters.

During the search for 'components that were activated shortly before the fault' both lift thrusters were again nominated. In this experiment this means that they were inactive for at least 5 seconds before being activated, which must in turn have been within 3 seconds of the fault being detected. These nominations are highly relevant as it is one of the lift thrusters that has failed on activation.

The final Suspicion Index for this experiment was:

```
Suspicion Index at Mission Time: 439.735
Component Level Diagnostic Iteration: 0
PORT_LIFT_THRUSTER: 2
STBD_LIFT_THRUSTER: 2
STBD_LEM_SENSOR: 1
AOSI_EZ3_COMPASS_SENSOR: 1
```

During this test the Correlator again provided highly accurate nominations.

6.7.3 Evaluation: Active Components are More Relevant

The final item of domain independent information used by the Correlator is that active components are more likely to be related to a fault than inactive components. This is shown by the experiments conducted to evaluate the Domain Independent Hierarchy Module (and later experiments evaluating the full RECOVERY system), where it is nearly impossible to detect or diagnose a fault if components are inactive. It is also borne out by the author's experience in test and repair.

6.8 Discussion of Domain Independent Diagnostic Tools

6.8.1 The Correlator

The Correlator is a simple module using a few pieces of generalised diagnostic information and yet in some of the experiments above it has highlighted extremely relevant correlations, showing that it has a great deal of potential.

6.8.2 The Hierarchy Module

The performance of the Hierarchy Module depends greatly on the amount and location of detected faults. The current implementation of RECOVERY has limited detection information and so the Hierarchy Module has had limited success. Detection also depends on the activity of the system, for instance if the vehicle is not moving then it is difficult to detect a thruster problem. However, even with such limited detection capabilities several totally correct diagnoses were obtained, the remainder would have been totally correct with some very simple extra detection.

Where there is a wide spread of detection, such as with the 5Vc and ±15Vc power supplies, the Hierarchy Module works extremely well, successfully diagnosing components without any direct sensing, modelling or fault-specific information. This

shows that the Hierarchy Module in combination with RECOVERY's Relational Model has great potential as a diagnostic tool.

The performance of the Hierarchy Module will also be improved when other diagnostic tools provide component nominations, as in the full system tests. For instance, the Port and Starboard Navigation power supplies have no direct sensing of the port and starboard thrusters, but the thrusters may be nominated using the Model-Based Diagnosis tools, which use models containing the thrusters.

The current implementation of the Hierarchy Module is very simple, using only two simple guidelines (as detailed in Section 5.5):

- If several components are diagnosed as faulty and they have a common supply connection which is further up the hierarchy then it is the commonly connected component that is faulty
- If a commonly connected supply component has been nominated as faulty then the supplied components should be exonerated of suspicion.

Even so it works very well. The use of more powerful diagnostic information should further improve diagnostic performance.

6.9 Chapter Summary

This chapter presented the evaluation vehicle, RAUVER, and the evaluation locations of Crosswood and Harperrig Reservoirs. The format of the results and experimental setup was detailed before moving on to the evaluation of the domain-independent diagnostic tools. Both the Hierarchy and Correlator modules were evaluated in isolation shown to be able to find extremely useful diagnostic information, in some cases correctly diagnosing a faulty component that was not nominated by any other diagnostic tool.

The next chapter evaluates the full RECOVERY system using all supported types of diagnostic tool and knowledge. RECOVERY's focus of suspicion method of search space reduction is then evaluated and the results discussed.

Chapter 7

Results: Evaluation of Full RECOVERY System and Search Space Reduction

7.1 Aim of This Chapter

This chapter is in two parts. In the first part the full RECOVERY system is evaluated on real mission files gathered during RAUVER operations. In the second part the focus is on evaluating RECOVERY's Search Space Reduction method and comparing it to a classic Rulebase. Each section is followed by a discussion.

7.2 Evaluation of the Full RECOVERY System

These experiments use the full RECOVERY system and are primarily used to show that it is a fully operational implementation of the proposed Integrated Diagnostic architecture, using heterogeneous diagnostic tools and knowledge on a real autonomous robot.

There are five different diagnostic tools integrated into the current implementation, these are:

A Model-Based Diagnostic Engine

A Static Rulebase 'matcher'

A Dynamic Rulebase 'matcher'

A Domain-Independent Correlator

A Domain-Independent Hierarchy Module

There are also four different types of information:

Static rules

Dynamic rules

Domain-Independent rules

Algorithmic models

These experiments use real, unmodified mission log files. Details of system operation for the first and last iterations are shown, full diagnostic output is included in the appendices.

The metric for these tests is correctness of diagnosis.

7.2.1 Experiment Setup

For all experiments in this chapter (unless otherwise stated) the setup was:

PC Setup

Processor: Pentium 200MMX

Memory: 32MB

Operating System: Microsoft Windows NT 4.0

No other programs running

For all experiments in this chapter RECOVERY settings were:

RECOVERY Settings

Command and status polling frequency: 5Hz

Residual detection window: ±8.0

Domain-Independent Diagnostics:

Hierarchy Module nomination threshold: 0.6

Correlator Delta window: ±1.0

Correlator Recent window:

Inactive for at least 8 seconds before failure

Activated within 3 seconds of failure

Model-Based Diagnosis nomination windows:

Roll Model: ±1.5 Degrees

Power Models: ±0.5 Amps

Nomination Increment: 1

Models Used

For all results decoupled axis movement models were used as they are simple to generate and suitable for use at the low speeds used here (Caccia, Indiveri & Veruggio [19]). The simplicity of the models also means that they are crude and so a worst-case test for the RECOVERY system.

7.2.2 Format of Results

The first experiment is used to illustrate full RECOVERY operation and so the results are given as verbatim excerpts from the diagnostic output files. The amount of editing is kept to a minimum. To keep the amount of text down only the outputs for the first and last iterations are shown. For the remaining experiments the diagnostic outputs are summarised in table form.

The full text is available in the appendices.

7.2.3 The Mission

The mission segment detailed in this section is taken from a realistic scenario based on a survey and sample mission. The AUV's mission is to follow a waypoint sequence at a depth of a few metres in order to stay out of the splash zone. At each waypoint, the vehicle surfaces for a GPS fix before diving and then proceeding to the next waypoint. At the last waypoint, a final GPS fix and status transmission is performed before the vehicle dives to undertake the survey.

In this mission it is necessary to sample the water column with various sensors, often using a 'vertical lawnmower' pattern, where the vehicle starts a sample at the surface and dives to the seabed whilst sampling the water column. An ambient sample is then taken with the vehicle in standby mode. The AUV then repeats the water column sample whilst returning to the surface for a position fix, then moves to the next sample point before repeating the process. This mission segment details a single leg of this survey method:

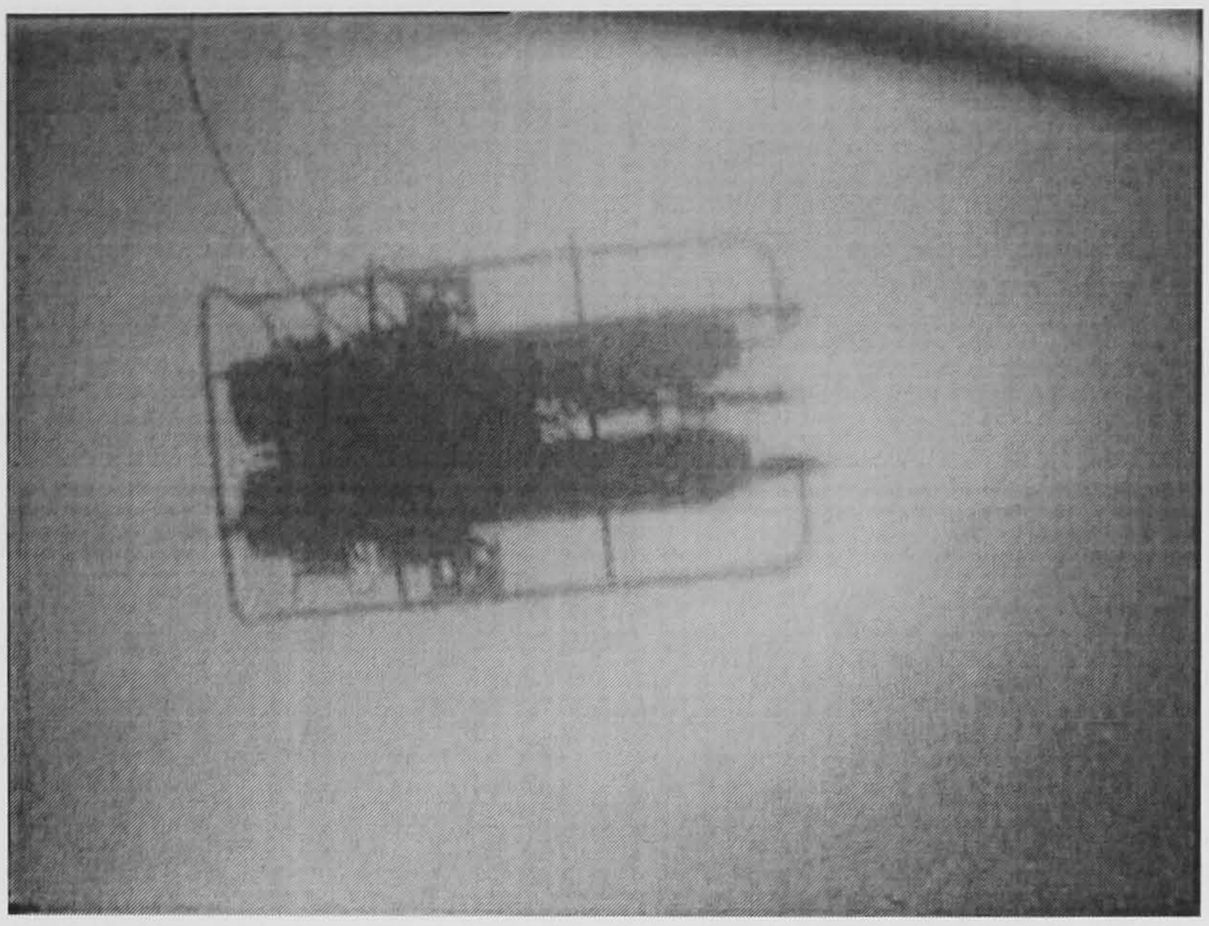


Figure 7.1: RAUVER Dives towards the loch-bed during RECOVERY Evaluation Mission. Picture taken from directly below RAUVER using a support ROV

- Goal 0: Surface and hover for GPS fix.
- Goal 1: Submerge to fully flood instruments.
- Goal 2: Dive vertically to survey-end depth.
- Goal 3: Standby (drift) while ambient readings taken.
- Goal 4: Ascend vertically to surface.
- Goal 5: Hover at surface until standby.

During the vertical sample process, the vehicle attitude must be kept within certain pitch, roll and heading constraints in order for the environmental sensors to work properly.

7.2.4 Experiment: No Fault

This test was performed to show that RECOVERY can distinguish between normal and faulty operation. RAUVER was run through the full mission several times without any component failures being induced.

No faults were detected showing that RECOVERY operation is correct. This is effectively a correct diagnosis.

7.2.5 Experiment: Vehicle Disturbed by Environmental Event

During this test RAUVER was floating without moving as in Goal 3 (Standby/drift while ambient readings taken). It was then given a shove in order to see how the currently systemically-orientated RECOVERY would cope with an unexpected environmental event such as a playful dolphin or a large wave.

The graphs are shown for the full mission with the time of failure marked. RECOVERY finishes its detection and diagnostic operation within approximately two seconds of these failure points, but in the current implementation RECOVERY acts as a mute watcher with no control over the vehicle, and so the mission (and therefore the graphs) carry on until mission end.

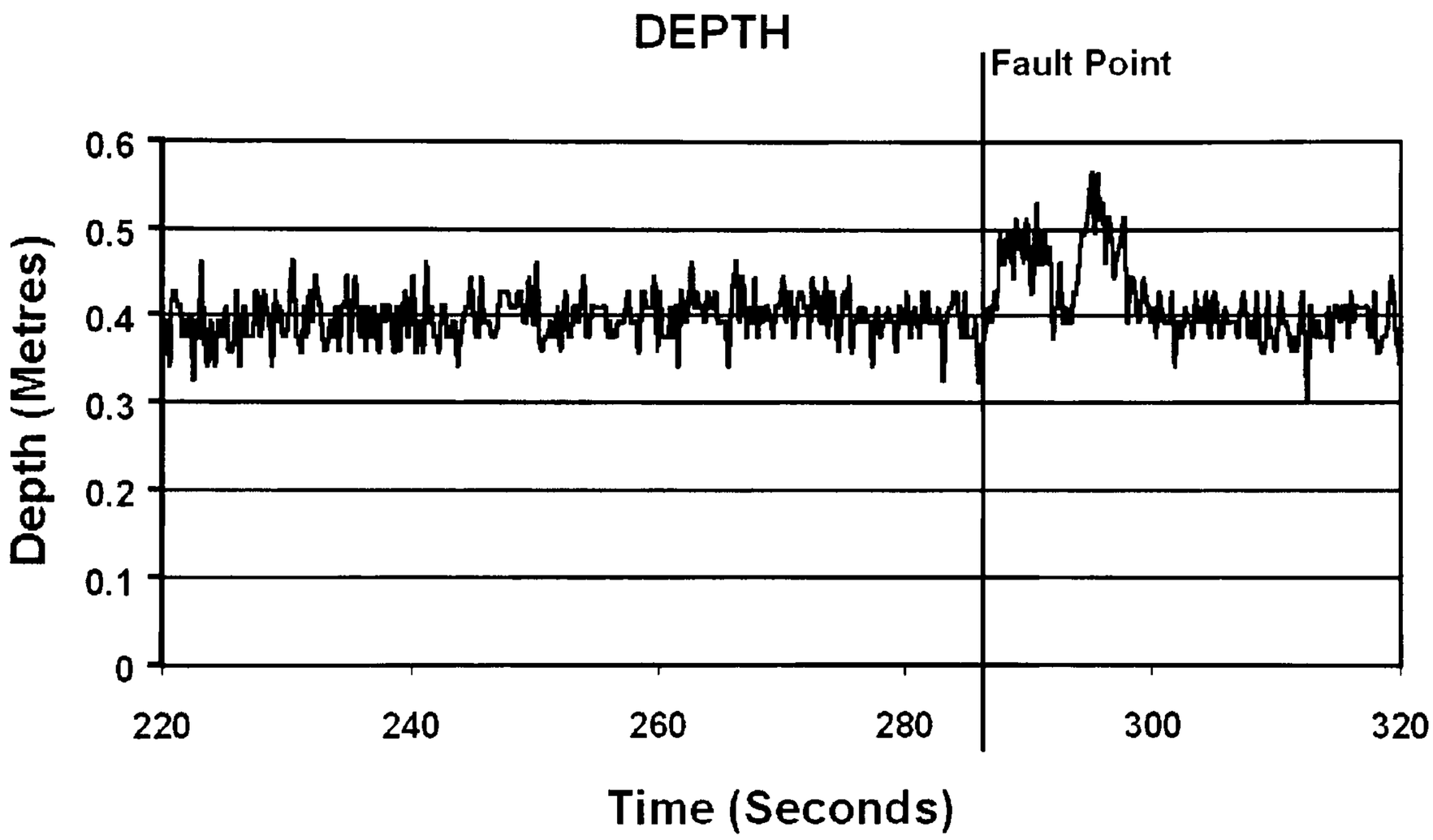


Figure 7.2: Environmental Disturbance: Depth

Figures 7.2 to 7.5 show depth, pitch, roll and yaw during this mission segment. RAUVER can be seen to be gently rocking (due to surface wave action) before the environmental event is initiated at approximately 286 seconds (Mission Time). The event forces a considerable change in pitch, roll and heading, with a minor change in depth.

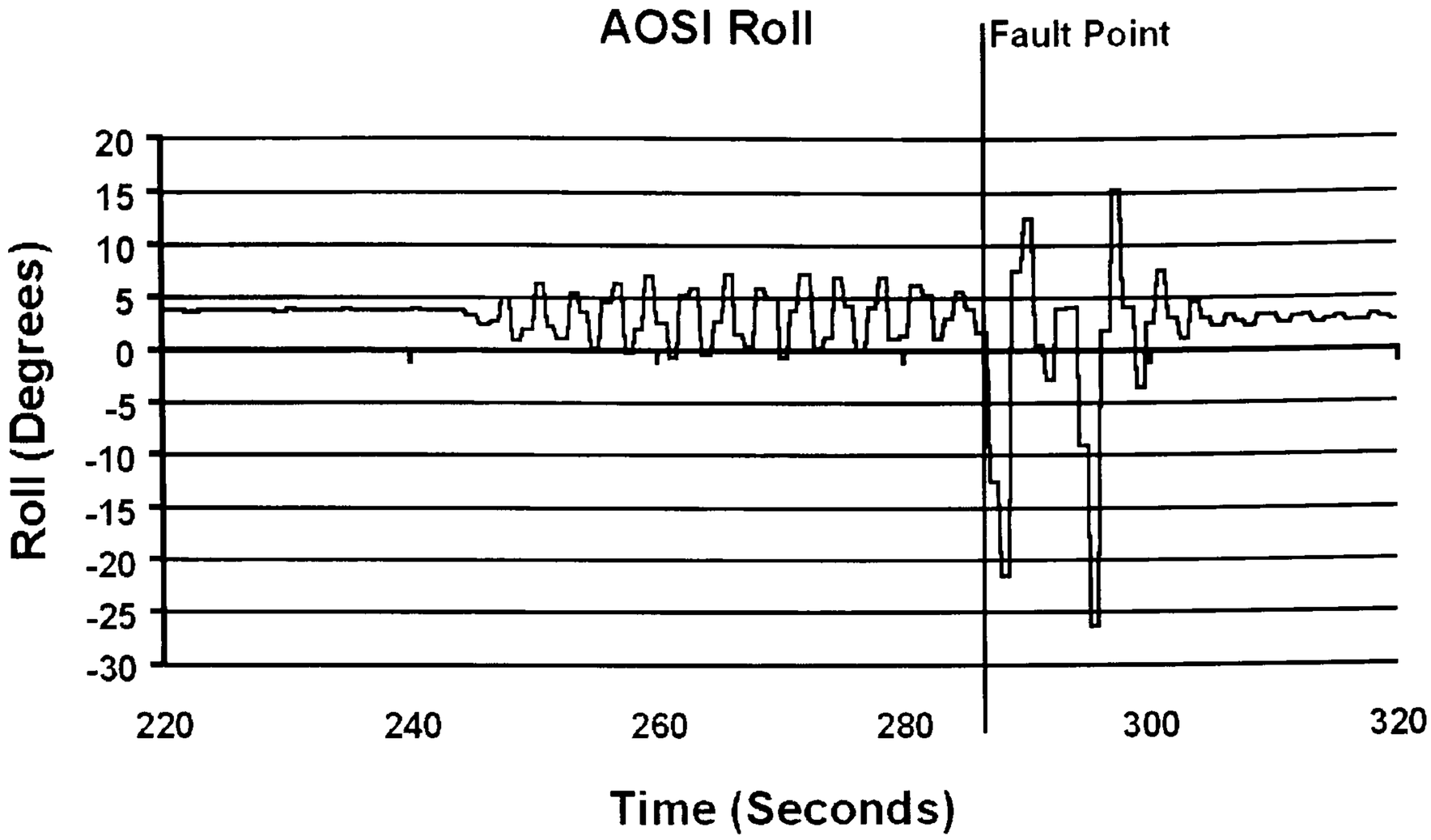


Figure 7.3: Environmental Disturbance: Roll

First Iteration: Detection

RECOVERY correctly detects the event at Mission Time 286.985 due to the residual between modelled and sensed roll parameters. The scant model coverage means that the discrepancies in pitch, heading and depth are undetected:

```
Residual detected between
Modelled Node: ROLL_MODEL
and Sensed Node: AOSI_ROLL
Suspicion Index at Mission Time: 286.985
Component Level Diagnostic Iteration: 0
AOSI_EZ3_COMPASS_SENSOR: 1
```

As usual, the sensor providing the residual is automatically nominated.

First Iteration: Domain Independent (Correlator)

The Correlator is invoked, producing the output:

```
Correlator Invoked at Mission Time: 286.985
Delta_Correlations:
```

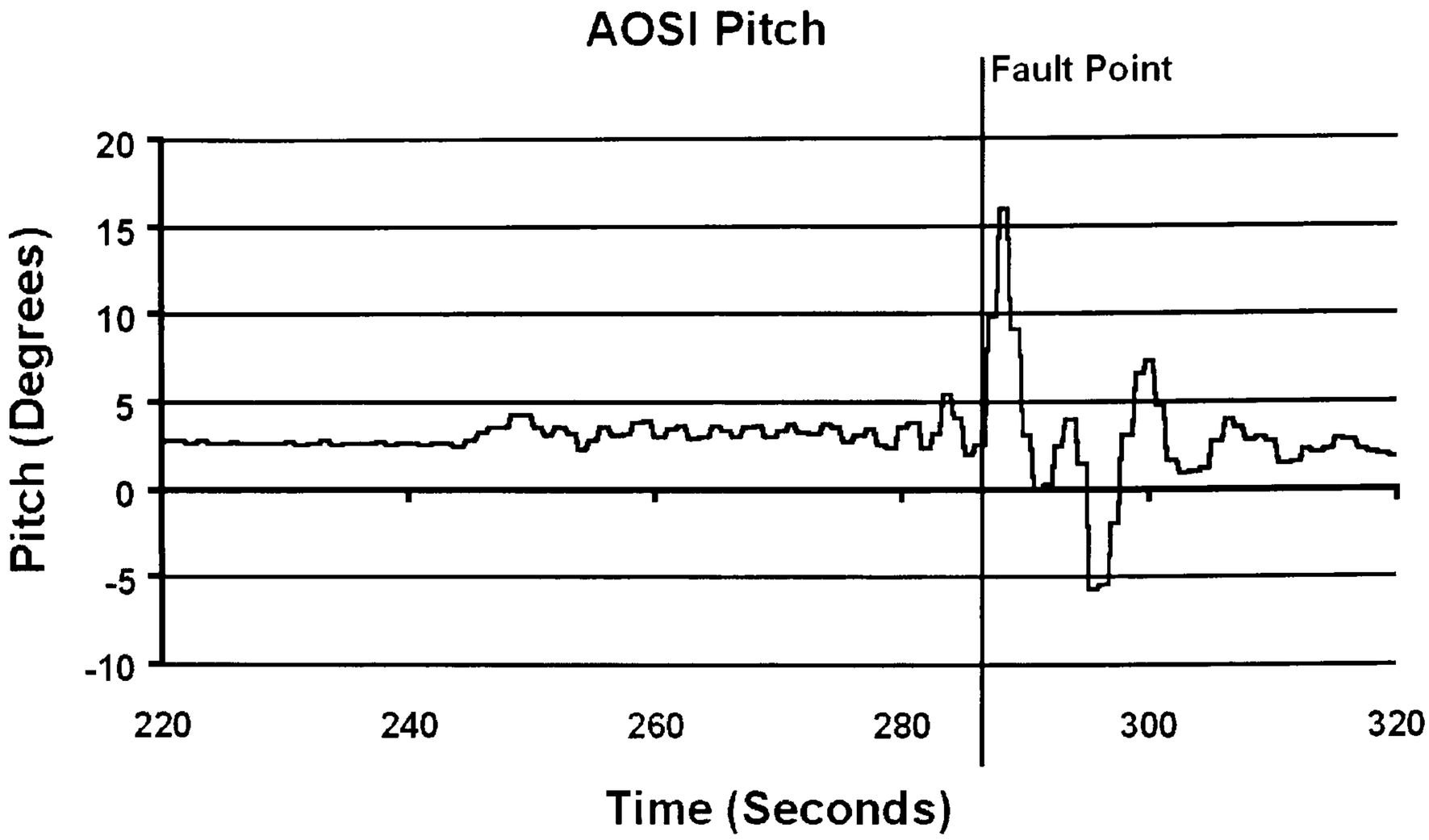


Figure 7.4: Environmental Disturbance: Pitch

```
Bad_Node: AOSI_ROLL

Delta_Matches:

AOSI_MAGZ

Recent_Comp_Correlations:

(none)
```

It has found a coincidental delta match with the AOSI_MAGZ parameter.

First Iteration: Model-Based Diagnostics

The Model-Based Diagnosis Engine is now invoked. As the residual was only detected with the roll model this is the only one to be evaluated:

```
Model-Based Diagnosis Engines invoked at: 286.985

Roll Engine Proposes:

PORT_LIFT_THRUSTER

STBD_LIFT_THRUSTER

Port Power Engine Proposes:

(none)

Stbd Power Engine Proposes:
```

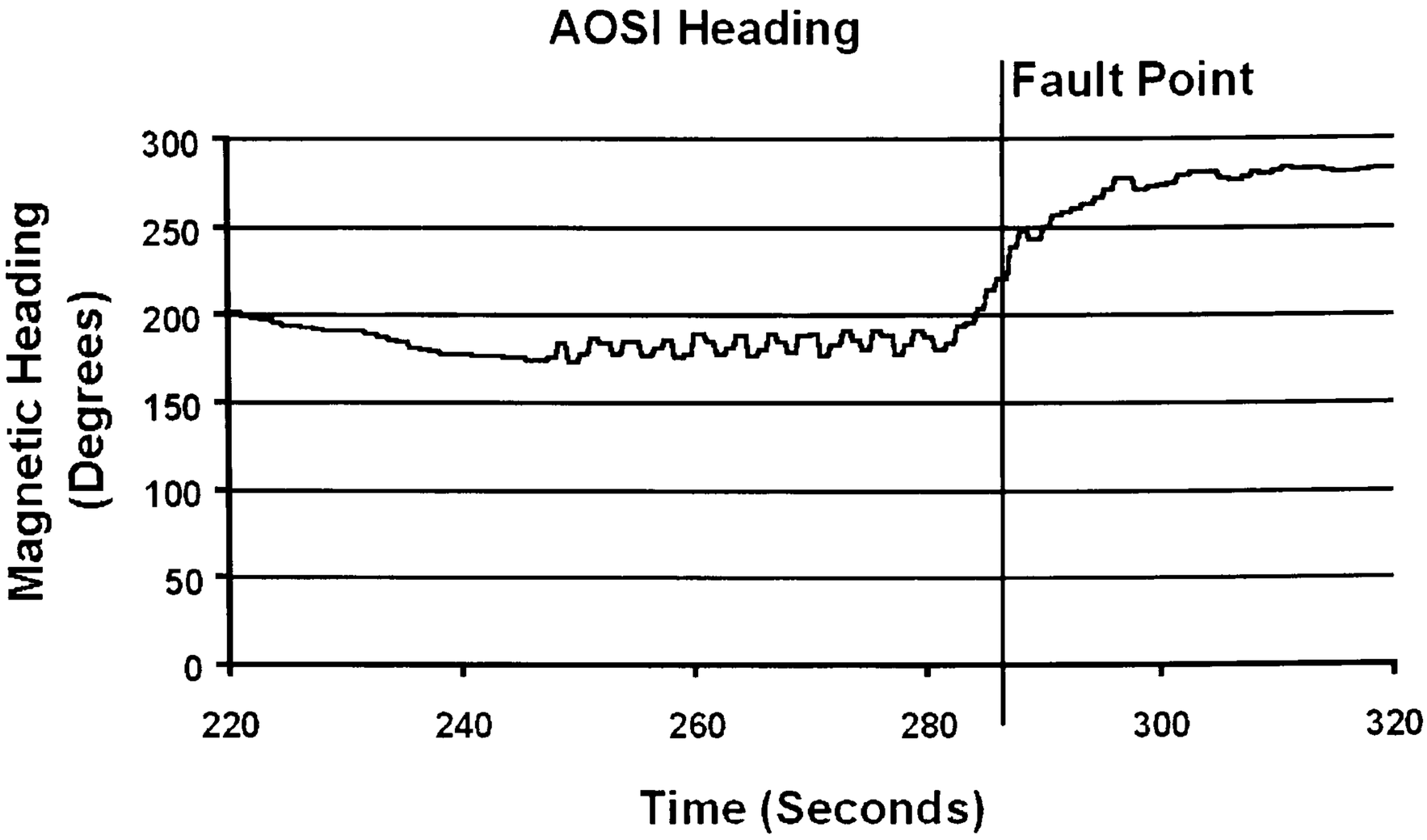


Figure 7.5: Environmental Disturbance: Magnetic Heading

(none)

It has found that the level of roll could be produced by either of the lift thrusters malfunctioning.

First Iteration: Suspicion Index

At the end of the first iteration the Suspicion Index is:

```
Suspicion Index at Mission Time: 286.985
Component Level Diagnostic Iteration: 0
AOSI_EZ3_COMPASS_SENSOR: 2
PORT_LIFT_THRUSTER: 1
STBD_LIFT_THRUSTER: 1
```

The lift thrusters have been nominated by the Model-Based Diagnosis Engine whilst the AOSI compass has two nominations, once for generating the discrepant parameter and once for the coincidental delta match between the roll and magnetic x parameters.

Seventh Iteration: Detection

Although the environmental event lasted for several seconds in total it only forced the roll parameter outside acceptable levels for short periods. RECOVERY runs over ten iterations which covers approximately two seconds. In this experiment the roll parameter was back in the acceptable window after the seventh iteration, meaning that there was no detection and hence no diagnostics for iterations eight to ten. For this reason the last iteration containing detection and diagnosis (seventh) is detailed here rather than the final (tenth) iteration.

Again, the only detection is provided by the residual between sensed and modelled roll.

```
Residual detected between
Modelled Node: ROLL_MODEL
and Sensed Node: AOSI_ROLL
Suspicion Index at Mission Time: 288.52
Component Level Diagnostic Iteration: 7
AOSI_EZ3_COMPASS_SENSOR: 15
PORT_LIFT_THRUSTER: 7
STBD_LIFT_THRUSTER: 7
```

Seventh Iteration: Domain Independent (Correlator)

The Correlator produced:

```
Correlator Invoked at Mission Time: 288.52
Delta_Correlations:
Bad_Node: AOSI_ROLL
Delta_Matches:
AOSI_MAGZ
Recent_Comp_Correlations:
(none)
```

The Correlator has again found a delta match between AOSI_ROLL and AOSI_MAGZ.

Seventh Iteration: Model-Based Diagnostics

The Model-Based Diagnosis Engine is again invoked only on the Roll model and finds that the discrepant roll could be produced by either lift-thruster malfunctioning:

```
Model-Based Diagnosis Engines invoked at: 288.52
Roll Engine Proposes:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER
Port Power Engine Proposes:
(none)
Stbd Power Engine Proposes:
(none)
```

Seventh Iteration: Suspicion Index

At the end of the seventh iteration the Suspicion Index is:

```
Suspicion Index at Mission Time: 288.52
Component Level Diagnostic Iteration: 7
AOSI_EZ3_COMPASS_SENSOR: 16
PORT_LIFT_THRUSTER: 8
STBD_LIFT_THRUSTER: 8
```

The lift-thrusters have been nominated by the Model-Based Diagnosis Engine. The AOSI compass has been repeatedly nominated for generating both the discrepant roll and magnetic x parameters.

Post Iteration: Domain Independent (Hierarchy)

As iterations eight to ten were not active the post-iteration section is now detailed. The Domain-Independent Hierarchy Module produced:

```
Domain Independent Diagnostics (post iterations)
Nodal Power Supply Diagnosis:
PORT_BATTERY  Fault Rating: 0.
```

```
STBD_BATTERY  Fault Rating: 0.
PORT_NAV_POWER  Fault Rating: 0.5
STBD_NAV_POWER  Fault Rating: 0.5
TWELVE_PCONTROL  Fault Rating: 0.25
TWENTYFOUR_PCONTROL  Fault Rating: 0.
COMMS_POWER  Fault Rating: 0.
FIVE_PCONTROL  Fault Rating: 0.
FIFTEEN_PCONTROL  Fault Rating: 0.
MINUSFIFTEEN_PCONTROL  Fault Rating: 0.
```

The Port and Starboard Navigation power supplies have the highest Fault Rating as they only supply two thrusters, lift and main. As in both cases the lift thruster has been nominated this leads to the Fault Rating of 0.5. The TWELVE_PCONTROL (12Vc) power supply has the next highest rating as it supplies the AOSI compass.

None of the power supplies have been nominated as they do not have enough active supplied components diagnosed as faulty. This is a correct diagnosis.

Post Iteration: Suspicion Index

As no power supplies were nominated the Suspicion Index is unchanged from that of iteration seven:

```
Suspicion Index at Mission Time: 288.955
Component Level Diagnostic Iteration: 10
AOSI_EZ3_COMPASS_SENSOR: 16
PORT_LIFT_THRUSTER: 8
STBD_LIFT_THRUSTER: 8
```

Component-Specific Diagnostic Level

RECOVERY now enters component-specific diagnostics, where the component-specific information attached to each component via the Relational Model is evaluated. Before this the Suspicion Index must have the sub-components (components that may only be diagnosed using component-specific information) added. In this case the

Port and Starboard Lift-Thrusters have five sub-components and so these are inserted after the relevant component:

```
Component-Specific Diagnostic Level Entered
Adding Related Sub-Components to Suspicion Index:
Final Suspicion Index at Mission Time: 288.955
Sub-Component Diagnostics Level
AOSI_EZ3_COMPASS_SENSOR: 16
PORT_LIFT_THRUSTER: 8
PORT_LIFT_THRUST_CONTROL: 0
PORT_LIFT_MOTOR: 0
PORT_LIFT_THRUSTER_BRUSH: 0
PORT_LIFT_GBOX: 0
PORT_LIFT_PROPELLER: 0
STBD_LIFT_THRUSTER: 8
STBD_LIFT_THRUST_CONTROL: 0
STBD_LIFT_MOTOR: 0
STBD_LIFT_THRUSTER_BRUSH: 0
STBD_LIFT_GBOX: 0
STBD_LIFT_PROPELLER: 0
```

The sub-components were not within the scope of the component-level diagnostics and so they have no nominations. The sub-component diagnostic stage steps through the list from the top to the bottom and so the zero rating of the sub-components does not matter.

The information attached to each suspicious component is now evaluated. At present the only information attached to the components consists of dynamic rules, or rules that specify system behaviour over time:

```
Dynamic Rule Testing of Component:
AOSI_EZ3_COMPASS_SENSOR
PORT_LIFT_THRUSTER Started
PORT_LIFT_THRUST_CONTROL Started
```

```
PORT_LIFT_MOTOR Started
PORT_LIFT_THRUSTER_BRUSH Started
PORT_LIFT_GBOX Started
PORT_LIFT_PROPELLER Started
STBD_LIFT_THRUSTER Started
STBD_LIFT_THRUST_CONTROL Started
STBD_LIFT_MOTOR Started
STBD_LIFT_THRUSTER_BRUSH Started
STBD_LIFT_GBOX Started
STBD_LIFT_PROPELLER Started
Scan finished (no match found)
Exiting diagnostics
```

In this case there is no information relevant to an environmental effect and so no relevant diagnosis is found. The final diagnosis is the final Suspicion Index (given above), in which the AOSI Compass sensor had the highest Suspicion Index due to repeatedly sensing the event. This is a reasonable result as the current implementation of RECOVERY is totally focussed on systemic problems. This experiment illustrates the need for diagnostic tools that can diagnose environmental problems.

7.2.6 Faulty Lift-Thruster Experiments

At the beginning of Goal 4 (return to surface) a Lift-Thruster fault is forced which drops the thrust from the relevant Lift-Thruster to zero. This causes a severe roll as the other Lift-Thruster is still operating normally, as shown in Figure 7.6. The resultant power loss and increased roll does not match modelled normal operation behaviour so invoking the RECOVERY diagnostic system.

This fault is equivalent to a tripped thruster controller due to wire squeeze/short, as occurred during early RAUVER operations and shown in the RAUVER fault log (Section 6.2.1. In these experiments the wires are made to 'short out' when the vehicle moves below a certain depth, which will squeeze the pressure hull and so reduce the amount of room available for components and wires.

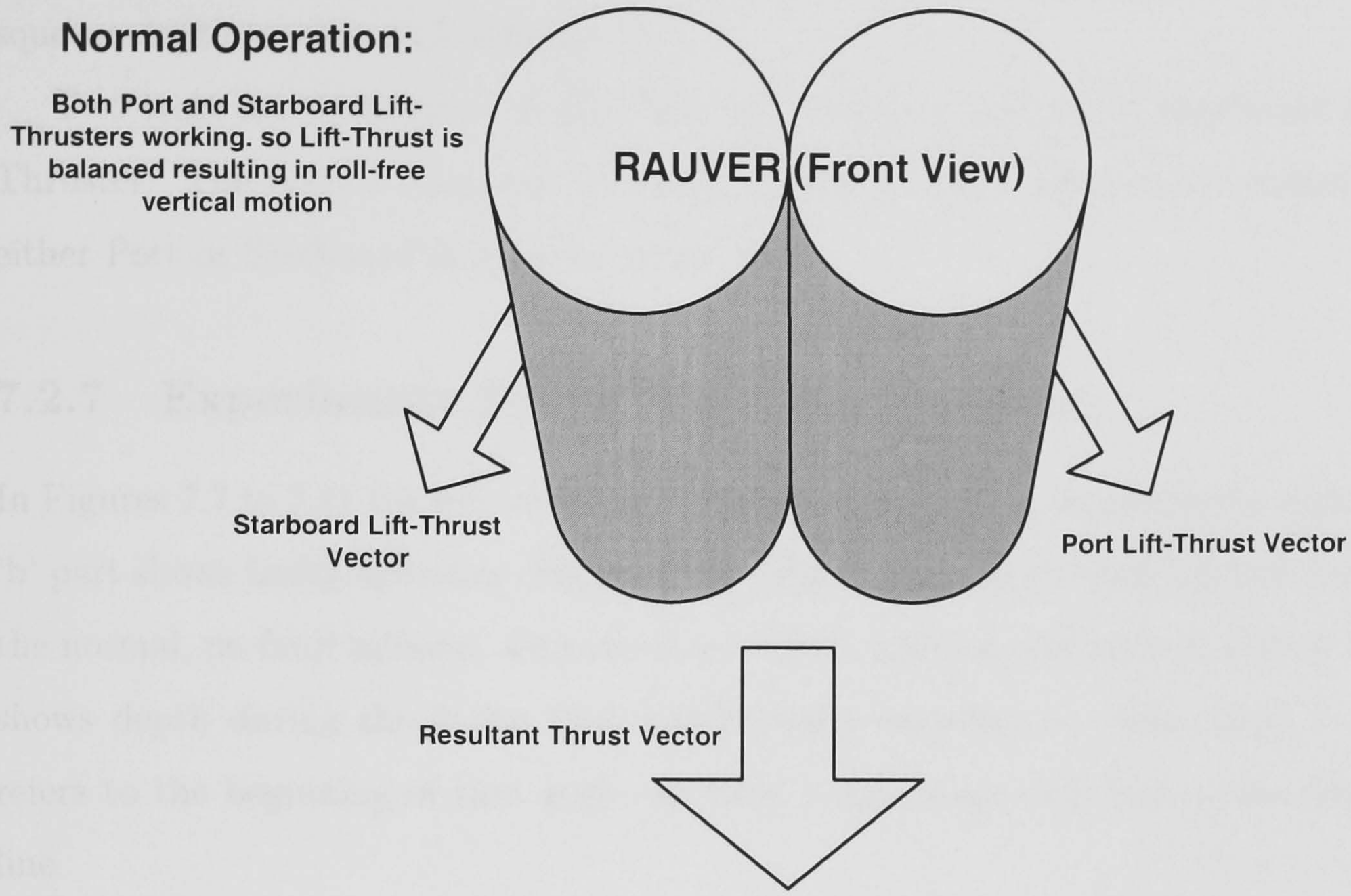


Figure 7.6: RAUVER During Normal and Faulty Ascent

This fault is forced using RECOVERY's utility software a short time after the vehicle has moved below the 'squeeze' depth. As it is a known fault it was attached to each thruster's component-specific information in the form of a dynamic rule. As the evaluation location (Crosswood Reservoir) is relatively shallow at 8 metres the squeeze depth was set to 5 metres.

Two tests are performed: faulty Port Lift-Thruster and faulty Starboard Lift-Thruster. The correct diagnosis for these tests is a faulty lift-thruster controller, either Port or Starboard depending on the test.

7.2.7 Experiment: Faulty Port Lift-Thruster

In Figures 7.7 to 7.11 the left, or 'a' part shows normal operation whilst the right, or 'b' part shows faulty operaion. Figure 7.7a (Depth) shows RAUVER's depth during the normal, no-fault mission, with the start of each mission goal shown. Figure 7.7b shows depth during the faulty Port Lift-Thruster experiment. The 'Goal 1' line refers to the beginning of that goal - so Goal 1 runs from that line to the Goal 2 line.

The depth profile hardly changes between missions because RAUVER can still ascend at near-normal speed using only one thruster due to their vectored nature. RAUVER's pitch (Figure 7.8 (Pitch)) is also nearly identical during both normal and faulty operation as the lift thrusters are near the centre of the vehicle.

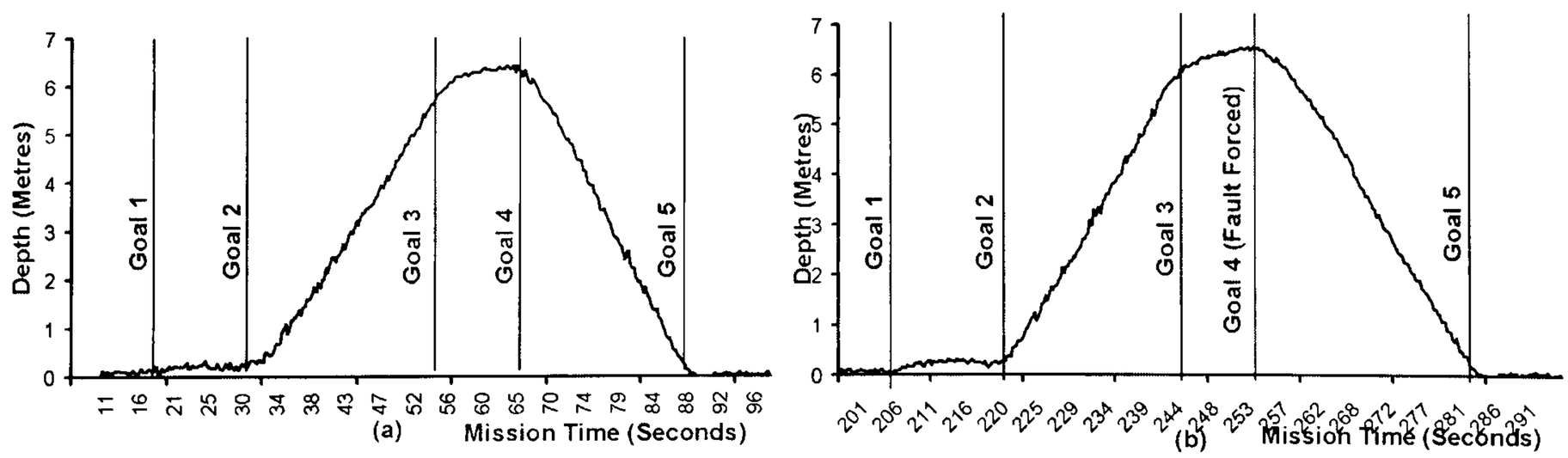


Figure 7.7: Faulty Port-Lift Thruster: Depth

In Figure 7.9b (Roll) the large change in roll caused by the failure of the Port Lift-Thruster can be seen during Goal 4 as the vehicle tries to ascend to the surface. The fault is also apparent in Figure 7.10b (Port Navigation Power) as the power

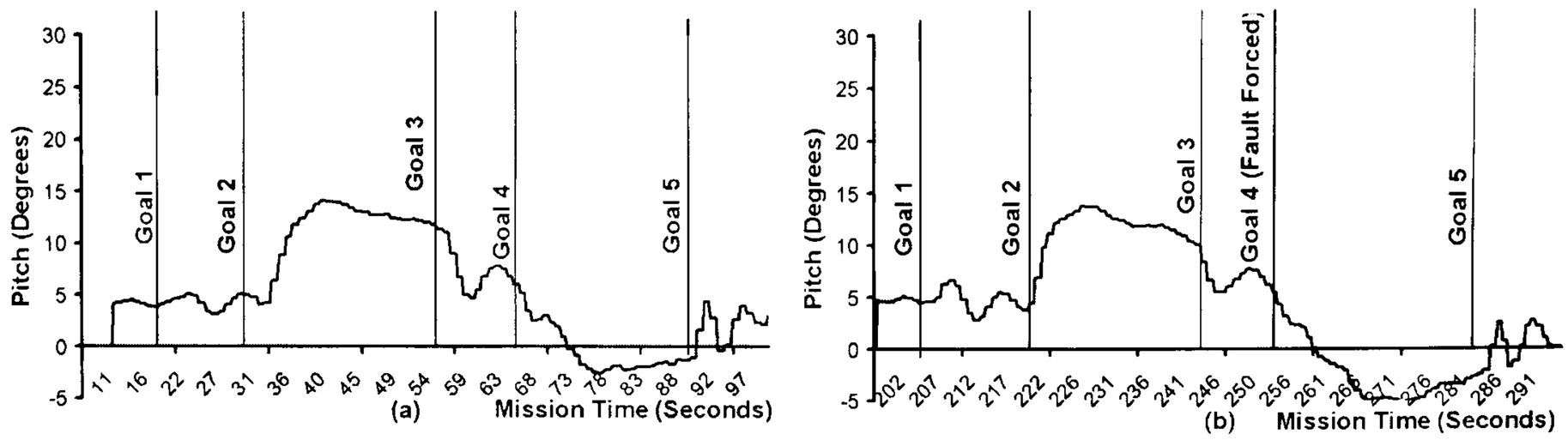


Figure 7.8: Faulty Port-Lift Thruster: Pitch

consumed is far below that normally consumed by the thruster, but Figure 7.11b (Starboard Navigation Power) is operating at near-normal figures.

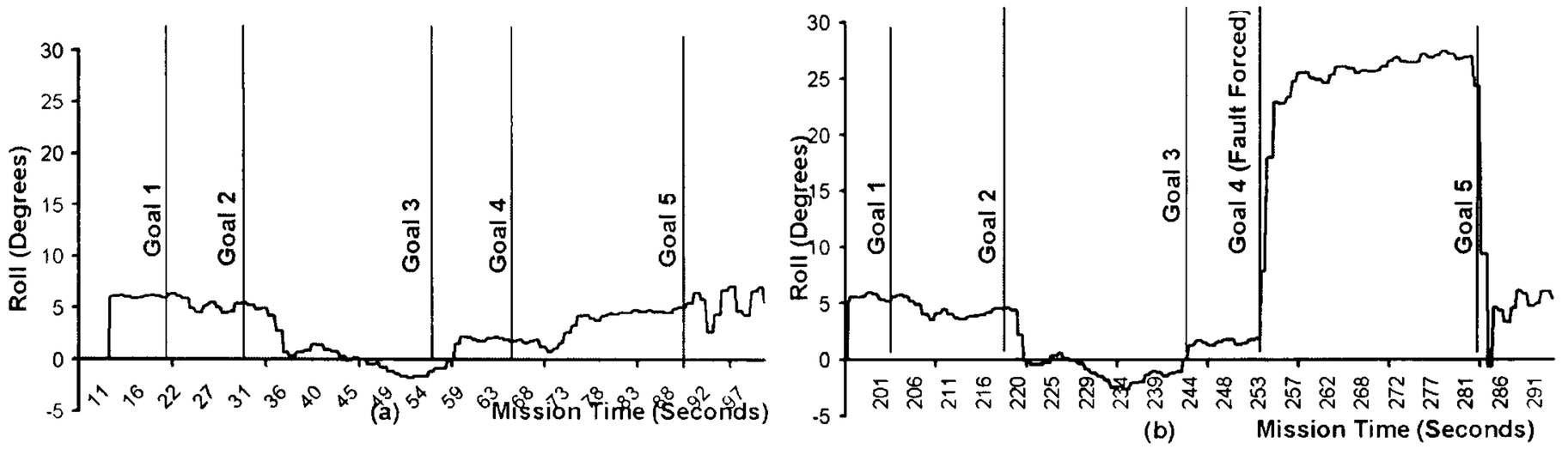


Figure 7.9: Faulty Port-Lift Thruster: Roll

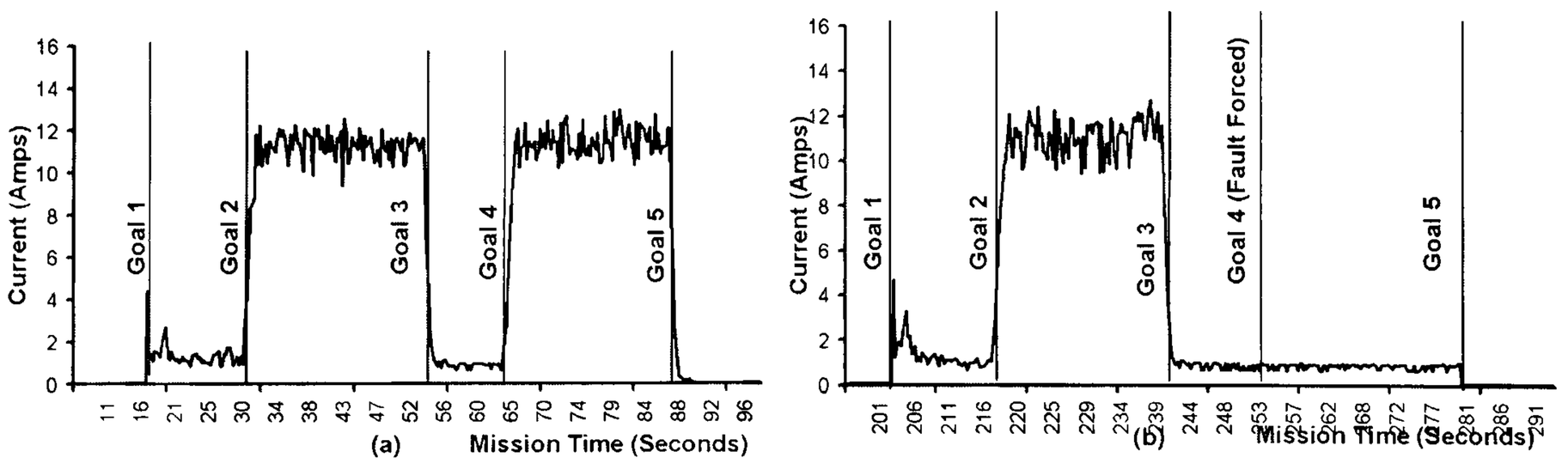


Figure 7.10: Faulty Port-Lift Thruster: Port Navigation Power Consumption

Because of the slow physical response of the vehicle the fault first becomes apparent in the power dimension. A residual is detected between the sensed and modelled power consumption of the Port Navigation power supply and so the diagnostic stage is invoked:

```
Residual detected between
```

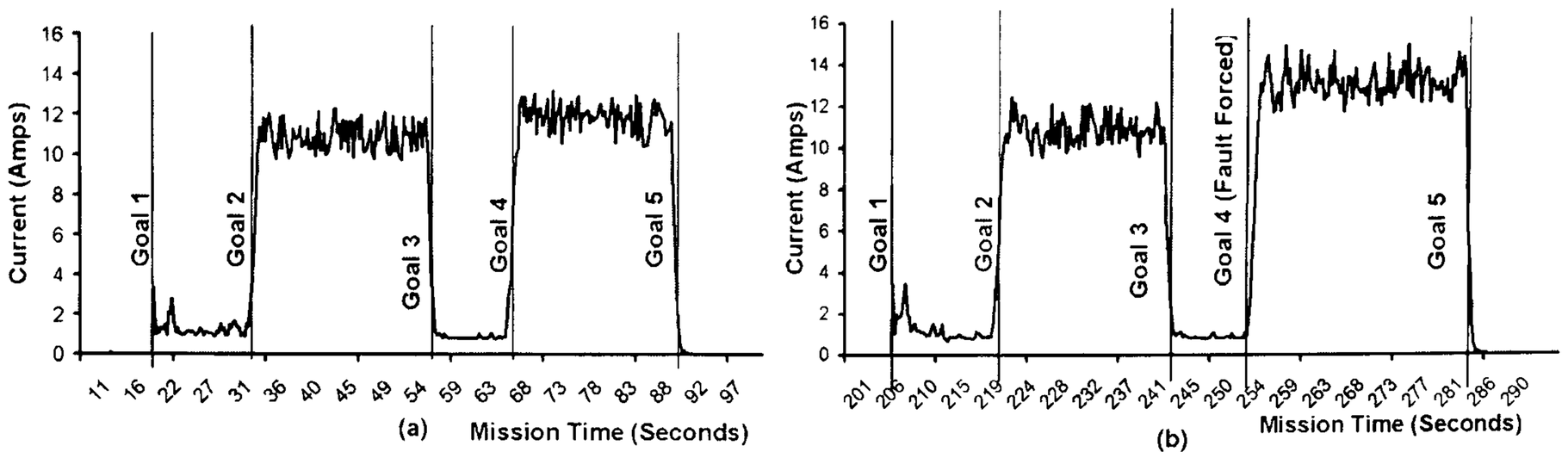


Figure 7.11: Faulty Port-Lift Thruster: Starboard Navigation Power Consumption

Component	*Detect*	Correlator		Models			*Total*
		Delta	*Recent*	*Roll*	*PNP*	*SNP*	
PORT_LIFT_THRUSTER		1			1		2
STBD_LIFT_THRUSTER		1					1
PORT_LAMP_CONTROL					1		1
PORT_LEM_SENSOR	1						1

Table 7.1: Faulty Port Lift-Thruster: First Iteration Nominations

```
Modelled Node: PORT_NAV_POWER_MODEL
and Sensed Node: PORT_LEM
```

Table 7.1 details the nominations produced during the first diagnostic iteration. The Port Lem Sensor is nominated once during detection as it generated the residual parameter. The Correlator's Delta Search then finds that both lift-thrusters have recently been activated and so nominates them. The Model-Based Diagnosis Engine, running the Port Navigation power model (and no others as they are not connected to the residual) parameter finds that the sensed power consumption can be matched with either the Port Lift-Thruster or the Port Lamp Controller.

The total number of nominations for each component is shown at the far right of the table (7.1).

Approximately a second after the fault is detected by the power model it becomes apparent in the movement dimension and is picked up by the roll model, this is repeated until the final iteration. During the final iteration the fault is still generating two residuals:

Component	*Detect*	Correlator		Models			*Total*
		Delta	*Recent*	*Roll*	*PNP*	*SNP*	
PORT_LIFT_THRUSTER				1	1		19
PORT_LAMP_CONTROL					1		14
PORT_LEM_SENSOR	1						10
STBD_LIFT_THRUSTER				1			9
PITCH_GYRO_SENSOR		1					9
AOSI_COMPASS_SENSOR	1						5
STBD_LAMP_CONTROL							4
PORT_TEMP_SENSOR							4

Table 7.2: Faulty Port Lift-Thruster: Final Iteration Nominations

```
Residual detected between
Modelled Node: PORT_NAV_POWER_MODEL
and Sensed Node: PORT_LEM
Residual detected between
Modelled Node: ROLL_MODEL
and Sensed Node: AOSI_ROLL
```

Table 7.2 details the nominations produced during the final diagnostic iteration. The total on the far right of the table shows the total number of nominations received over this and all preceding iterations. The Port Lem and AOSI Compass Sensor have both been nominated during detection for producing residuals.

The Correlator's Delta Search has matched the AOSI Roll and Gyro Pitch parameters and so the Pitch Gyro Sensor has been nominated once. As the Port and Starboard Lift-Thrusters have now been active for longer than the Correlator's Recent Components search will allow they are no longer nominated.

The Model-Based Diagnosis Engine nominates the Port Lift-Thruster using both the Roll and Port Navigation Power models, the latter of which also nominates the Port Lamp Controller.

As all ten iterations are complete RECOVERY now invokes the Domain-Independent Hierarchy Module to check for common source connections between nominated com-

ponents. The Port and Starboard Navigation power supplies both have all their supplied components nominated and so the power supplies have a fault rating of 1. This exceeds the nomination threshold and so they are both nominated.

```
Domain Independent Diagnostics (post iterations)
Nodal Power Supply Diagnosis:
PORT_NAV_POWER  Fault Rating: 1. (nominated)
STBD_NAV_POWER  Fault Rating: 1. (nominated)
```

If a power supply is nominated the supplied components are all denominated as it is assumed that it is the power supply which is at fault. This changes the Suspicion Index to:

```
Suspicion Index at Mission Time: 254.565
Component Level Diagnostic Iteration: 10
PORT_LEM_SENSOR: 10
PITCH_GYRO_SENSOR: 9
AOSI_EZ3_COMPASS_SENSOR: 5
PORT_POWERSTACK_TEMP_SENSOR: 4
PORT_NAV_POWER: 1
STBD_NAV_POWER: 1
```

The Port and Starboard Lift-Thrusters and Lamp Controllers have all been denominated and replaced with the Navigation power supplies. This denomination technique worked well when the Hierarchy Module was tested in isolation in Chapter 6. Unfortunately, when tested in the full system it has exonerated the very component that is at fault.

The Component-Specific Diagnostic Level is now entered. The relevant sub-components would now be added, but none of the nominated components have any sub-components and so the Suspicion Index is unchanged.

Evaluation of component-specific information attached to each nominated component proceeds:

```
Dynamic Rule Testing of Component:
```

```
PORT_LEM_SENSOR Started
PITCH_GYRO_SENSOR Started
AOSI_EZ3_COMPASS_SENSOR Started
PORT_POWERSTACK_TEMP_SENSOR Started
PORT_NAV_POWER Started
STBD_NAV_POWER Started
Scan finished (no match found)
```

RECOVERY only scans the components that have been nominated. The information (a static rule) detailing the wire-squeeze fault is attached to a non-nominated component and so it is not evaluated. This means that the known-fault of wire-squeeze is not found.

Due to the denomination effect of the domain-independent Hierarchy Module the faulty Port Lift-Thruster component is not even on the suspicious list anymore - it has been exonerated. The output to the mission controller consists of only the Suspicion Index when no exact match is found and so the exoneration of the faulty component is a serious issue. This shows that although denomination is an attractive idea the risks far outweigh the benefits.

Justification of Removal of Denomination

Following the above findings the denomination method was disabled for the remainder of the experiments. No other changes were made to the RECOVERY system during the evaluation of this research.

It was felt necessary and justifiable to remove the denomination for several reasons. Exoneration is not a fundamental concept of RECOVERY, it is a possible bonus that has been experimentally shown not to be worthwhile. Keeping it in would have removed other, more fundamental and important benefits such as Search Space Reduction (evaluated later in this chapter).

Although some other aspects of RECOVERY were found to be wanting they were retained, as they were fundamental concepts (such as domain-independent diagnostics). In these cases poor performance contributed to the research, as it highlighted areas of further work. Denomination is not fundamental and keeping

it would have contributed nothing to the research apart from distracting from the important, fundamental RECOVERY issues.

The identification of denomination as reducing overall RECOVERY performance shows the crucial need to evaluate systems in the real world on real data. Although much research, thought and discussion went into the design of RECOVERY, denomination was found to be a performance reducer very quickly once it was applied to real data. It also shows the need for flexibility of design, for even the best thought-out systems will miss aspects of real-world operation.

Note: The remainder of the experiments in this chapter will be performed without denomination. No other RECOVERY system changes were made during the evaluation of this research.

7.2.8 Experiment: Faulty Port Lift-Thruster Without Hierarchical Denomination

The faulty Port Lift-Thruster experiment was repeated with the denomination technique disabled. The Navigation power supplies are still nominated, but without denomination the Lift-Thrusters and Lamp Controllers are still in the Suspicion Index:

```
Suspicion Index at Mission Time: 254.565
Component Level Diagnostic Iteration: 10
PORT_LIFT_THRUSTER: 19
PORT_LAMP_CONTROL: 14
PORT_LEM_SENSOR: 10
STBD_LIFT_THRUSTER: 9
PITCH_GYRO_SENSOR: 9
AOSI_EZ3_COMPASS_SENSOR: 5
STBD_LAMP_CONTROL: 4
PORT_POWERSTACK_TEMP_SENSOR: 4
PORT_NAV_POWER: 1
STBD_NAV_POWER: 1
```

It is notable that the Port Lift Thruster, which is actually faulty, is at the top of the Suspicion Index. This is achieved even though so far only Domain-Independent tools and crude models have been used, none of which explicitly state the fault in the same way as a Rulebase.

The combination of RECOVERY's Domain-Independent Diagnostic Tools and the Model-Based Diagnosis Engine has resulted in the correct diagnosis of a faulty component without that fault being explicitly foreseen and encoded.

The Component-Specific Diagnostic Level is again entered and the relevant sub-components attached to the Suspicion Index, in this case to the Port and Starboard Lift-Thrusters (they are the only nominated components with sub-components):

```
Component-Specific Diagnostic Level Entered
Adding Related Sub-Components to Suspicion Index:
Final Suspicion Index at Mission Time: 254.565
Sub-Component Diagnostics Level
PORT_LIFT_THRUSTER: 19
PORT_LIFT_THRUST_CONTROL: 0
PORT_LIFT_MOTOR: 0
PORT_LIFT_THRUSTER_BRUSH: 0
PORT_LIFT_GBOX: 0
PORT_LIFT_PROPELLER: 0
PORT_LAMP_CONTROL: 14
PORT_LEM_SENSOR: 10
STBD_LIFT_THRUSTER: 9
STBD_LIFT_THRUST_CONTROL: 0
STBD_LIFT_MOTOR: 0
STBD_LIFT_THRUSTER_BRUSH: 0
STBD_LIFT_GBOX: 0
STBD_LIFT_PROPELLER: 0
PITCH_GYRO_SENSOR: 9
AOSI_EZ3_COMPASS_SENSOR: 5
STBD_LAMP_CONTROL: 4
```

```
PORT_POWERSTACK_TEMP_SENSOR: 4
PORT_NAV_POWER: 1
STBD_NAV_POWER: 1
```

The Suspicion Index, and hence the amount of component-specific information to be processed, is far longer than that produced when denomination was active. The crucial point is that this time it contains the faulty component.

RECOVERY now starts analysing the component-specific information attached to each of the components and sub-components. The information is passed to the relevant diagnostic tool, in this case the Dynamic Pattern Matcher. The component with the highest Suspicion Index is analysed first, followed by the next highest and so on:

```
Dynamic Rule Testing of Component:
PORT_LIFT_THRUSTER Started
PORT_LIFT_THRUST_CONTROL Started
Exact Match Found
Exiting diagnostics
```

This time RECOVERY has determined a match between the known 'wire squeeze' fault (represented with a dynamic rule) contained in the Port Lift Thruster Control sub-component. This sub-component is determined to be faulty. The knowledge attached to the component could also have contained detailed algorithmic models of the thruster, or other detailed information, but developmental constraints prevented this.

Even without the component-specific information being evaluated the overall Port Lift-Thruster component had the highest Suspicion Index and in the event of no exact match being found would still have had the highest probability of being faulty. This is a correct diagnosis.

7.2.9 Experiment: Faulty Starboard Lift-Thruster

This is a repeat of the faulty Port Lift-Thruster experiment performed in Section 7.2.8 but instead with the Starboard Lift-Thruster failing mid-mission. The experimental setup is identical.

Figure 7.12 (Depth) displays depth during both normal (Figure 7.12a) and faulty (7.12b) operation. As in the faulty Port Lift-Thruster experiment the change in the depth profile due to lift-thruster failure is minimal due to the vectored nature of the lift-thrusters. Figure 7.13 also shows that the pitch is, again, relatively unchanged as the lift-thrusters are positioned approximately in the centre of the vehicle.

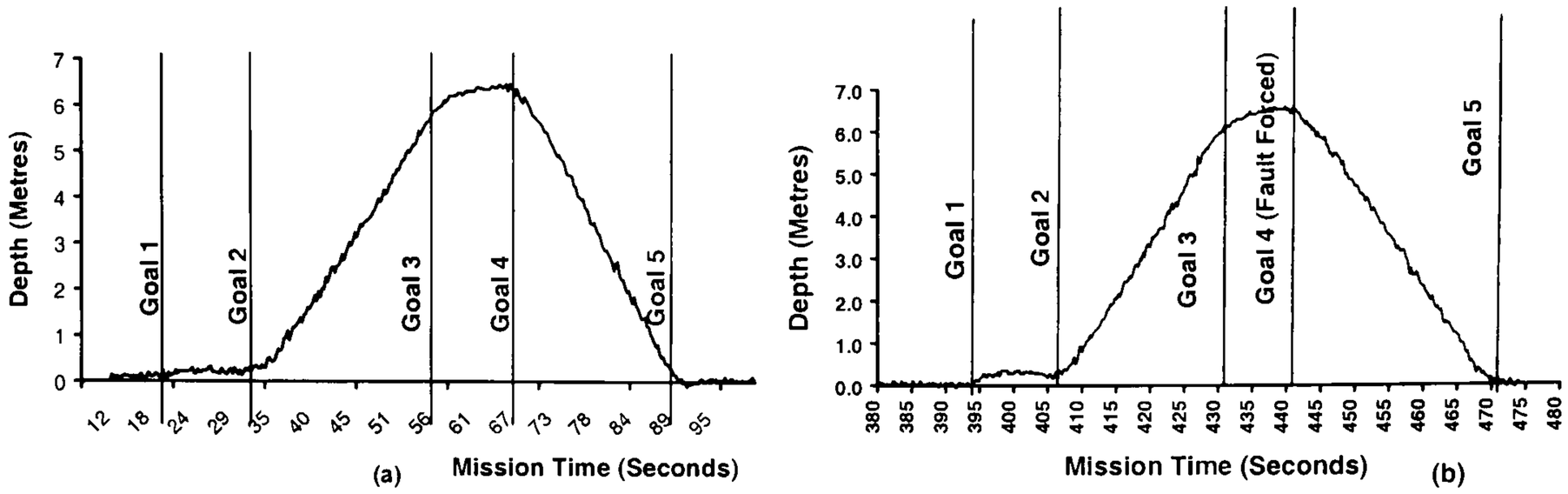


Figure 7.12: Faulty Starboard Lift-Thruster: Depth

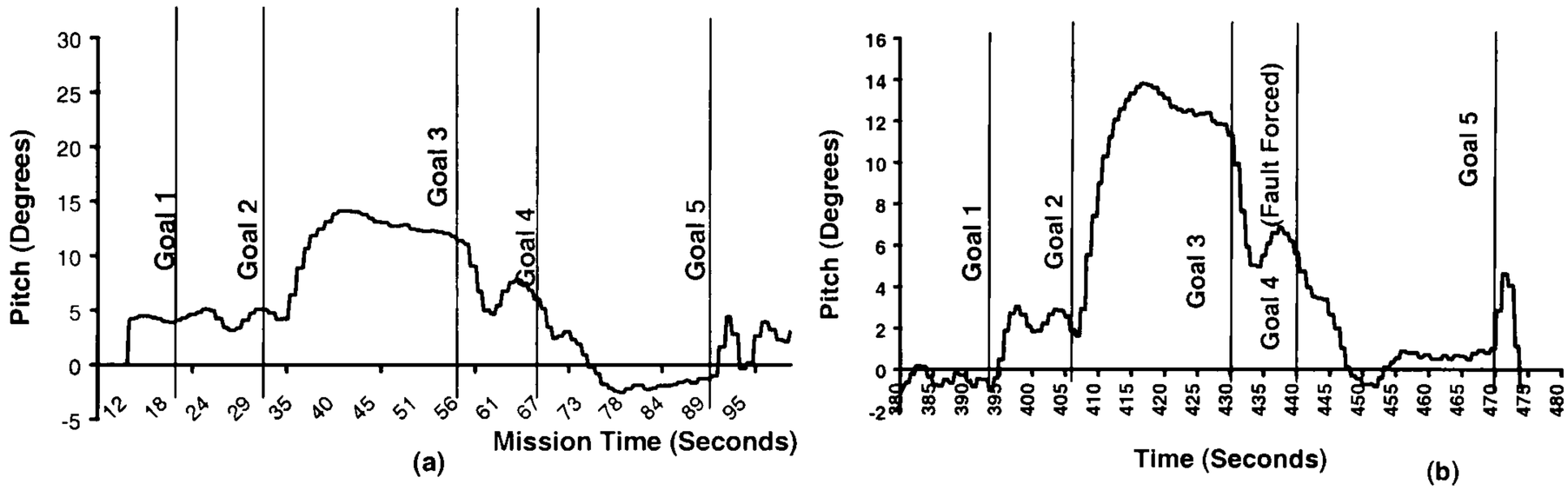


Figure 7.13: Faulty Starboard Lift-Thruster: Pitch

Figure 7.14 shows the large negative roll produced by the failure of the Starboard Lift-Thruster. (The failure of the Port Lift-Thruster produced a positive roll.)

This time Port Navigation power consumption (Figure 7.15) shows no loss in power whereas Starboard Navigation power consumption (Figure 7.16) has a clear discrepancy between normal and faulty operation.

Because of the slow physical movement of RAUVER the fault first becomes apparent in the power dimension, with a residual being generated between the sensed and modelled Starboard Navigation power supply consumption:

```
Residual detected between
```

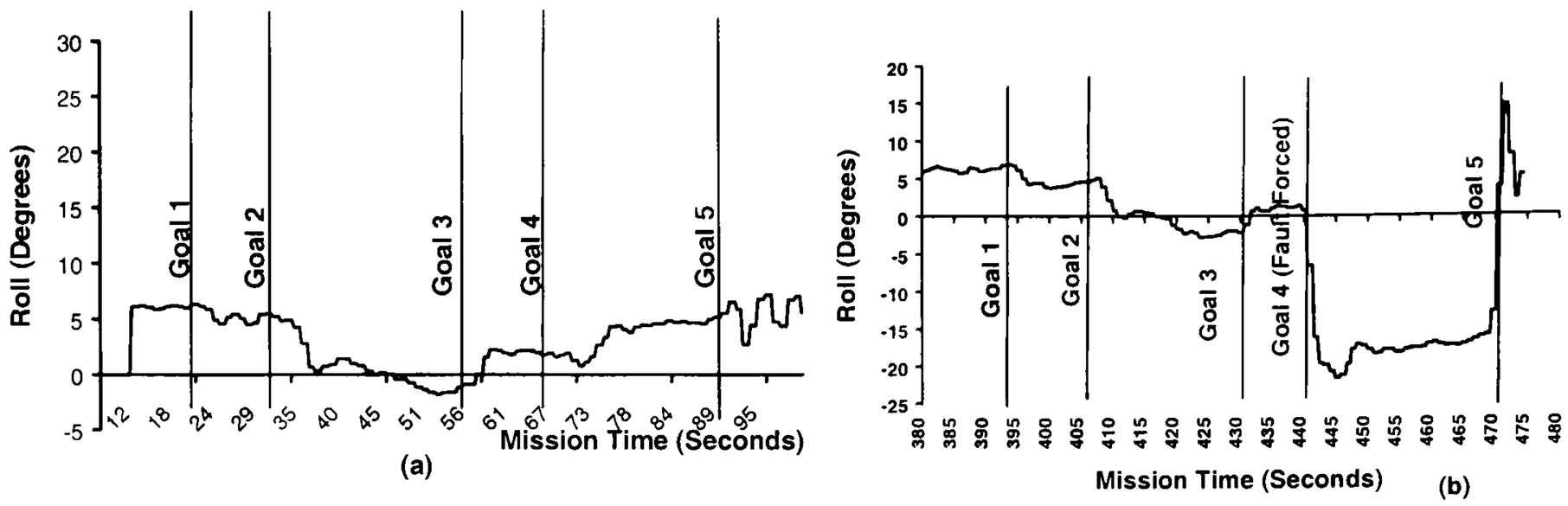


Figure 7.14: Faulty Starboard Lift-Thruster: Roll

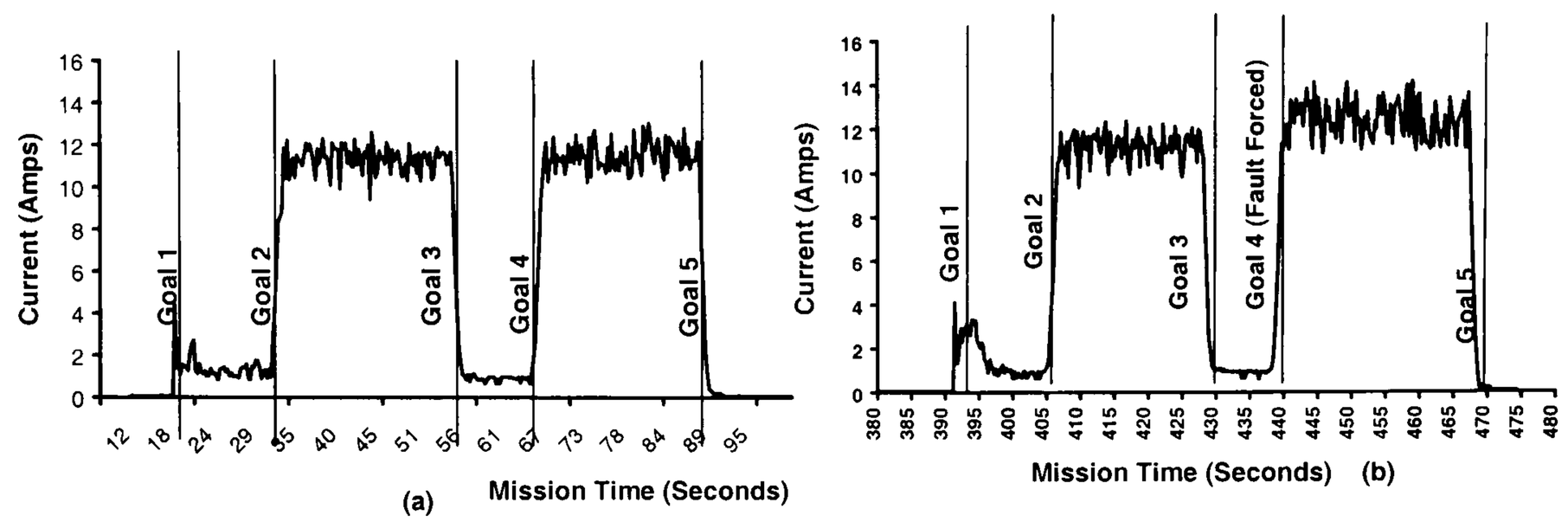


Figure 7.15: Faulty Starboard Lift-Thruster: Port Navigation Power Consumption

```
Modelled Node: STBD_NAV_POWER_MODEL
and Sensed Node: STBD_LEM
```

Table 7.3 details the nominations during first-iteration detection and diagnostics. The Starboard Lem is nominated once at detection for producing the Starboard Navigation Power residual.

The Correlator's Delta Search has matched both Port and Starboard Lift-Thruster demands with the Starboard Lem (power consumption) parameter. This is highly relevant as during this mission it is only the lift-thrusters that have been drawing significant amounts of power from the navigation supplies. Both lift-thrusters have been found by the Delta Search as they are set to identical values during this mission. The Delta Search has also matched the AOSI Magnetic X parameter, which is coincidental. The AOSI Compass is nominated as it generates this parameter.

The Correlator's Recent Components Search has found that both Port and Starboard Lift-Thrusters have been activated within 3 seconds (as detailed in the ex-

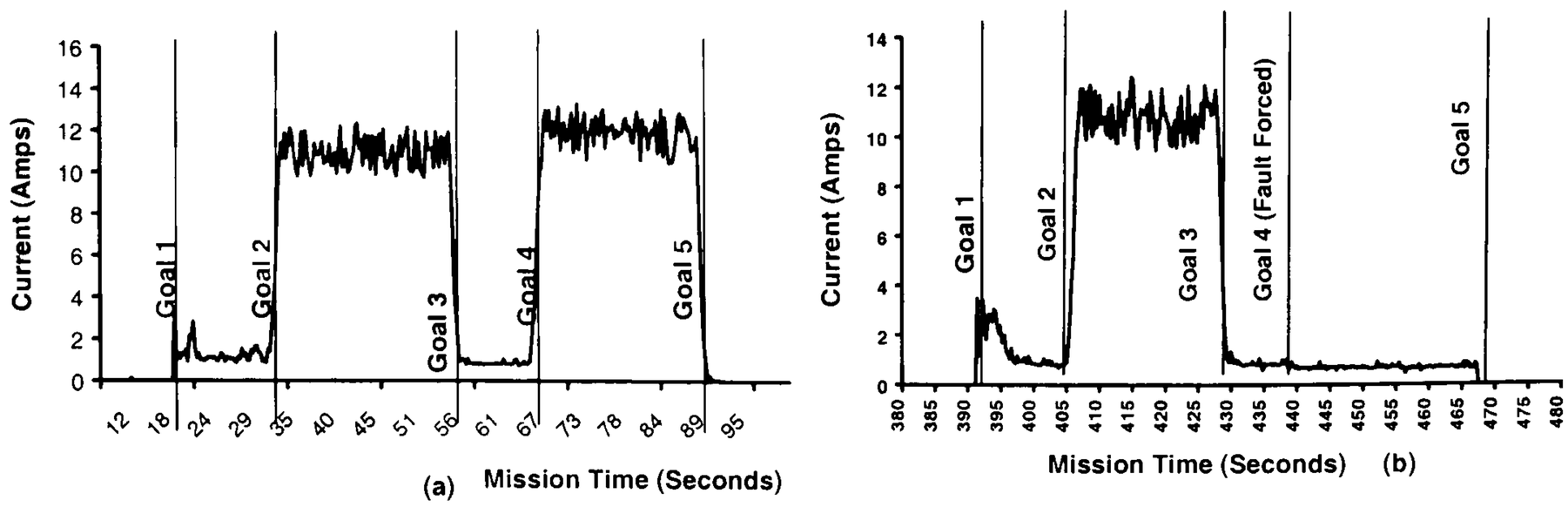


Figure 7.16: Faulty Starboard Lift-Thruster: Starboard Navigation Power Consumption

Component	*Detect*	Correlator		Models			*Total*
		Delta	*Recent*	*Roll*	*PNP*	*SNP*	
STBD_LIFT_THRUSTER		1	1			1	3
PORT_LIFT_THRUSTER		1	1				2
STBD_LAMP_CONTROL						1	1
STBD_LEM_SENSOR	1						1
AOSI_COMPASS_SENSOR		1					1

Table 7.3: Faulty Starboard Lift-Thruster: First Iteration Nominations

perimental setup in Section 7.2.1). This is another highly relevant discovery.

The Model-Based Diagnosis Engine has found that the faulty level of power consumption could be produced by either the Starboard Lift-Thruster or the Starboard Lamp Controller.

Approximately a second after the fault was first detected in the power dimension it became apparent in the movement dimension due to the extreme roll of the vehicle. During the final iteration the fault is still producing two residuals, one between sensed and modelled Starboard Navigation Power and AOSI Roll:

```
Residual detected between
Modelled Node: STBD_NAV_POWER_MODEL
and Sensed Node: STBD_LEM
Residual detected between
Modelled Node: ROLL_MODEL
```

Component	*Detect*	Correlator		Models			*Total*
		Delta	*Recent*	*Roll*	*PNP*	*SNP*	
AOSI_COMPASS_SENSOR	1	3					33
STBD_LIFT_THRUSTER		1		1		1	32
PORT_LIFT_THRUSTER		1		1			22
STBD_LAMP_CONTROL		1				1	17
STBD_LEM_SENSOR	1						10
PORT_LAMP_CONTROL		1					7
PITCH_GYRO_SENSOR		1					4

Table 7.4: Faulty Starboard Lift-Thruster: Final Iteration Nominations

and Sensed Node: AOSI_ROLL

Table 7.4 details the nominations generated during the final component-level diagnostic iteration. The totals at the far right of the table show total nominations for this and all preceding iterations. The Starboard Lem and AOSI Compass Sensors have both been nominated during detection for generating residuals.

The Correlator's Delta Search has matched the Starboard Lem parameter with both Port and Starboard Lift-Thruster demand parameters and the AOSI Magnetic X parameters. The AOSI Roll parameter has been matched with the Port and Starboard Lamp Control demand parameters (identical throughout the mission), the AOSI Temperature and Magnetic X Parameters (coincidental) and the Gyro Pitch parameter (also coincidental).

The Correlator's Recent Component Search has not found any matches as no components were activated close to this final iteration.

The Model-Based Diagnosis Engine has nominated both Port and Starboard Lift-Thrusters using the Roll model as the malfunctioning of either thruster could have caused the vehicle's roll. The Starboard Navigation power model has nominated the Starboard Lift-Thruster and the Starboard Lamp Controller for similar reasons.

Unlike the faulty Port Lift-Thruster experiment the faulty Starboard Lift-Thruster component does not have the highest Suspicion Index at this point. This is due to the high number of coincidental nominations by the Correlator of AOSI generated

parameters. The AOSI Compass component generates seven parameters, the highest ratio of any component on RAUVER. This means that it has a high likelihood of being nominated by irrelevant Correlator matches.

The faulty Starboard Lift-Thruster is high on the list with the second highest Suspicion Index. This still represents a significant achievement.

As the component-level diagnostic iterations are finished the post-iteration Domain-Independent Hierarchy Module is invoked to check for common connections between nominated components:

```
Domain Independent Diagnostics (post iterations)
Nodal Power Supply Diagnosis:
PORT_NAV_POWER  Fault Rating: 1. (nominated)
STBD_NAV_POWER  Fault Rating: 1. (nominated)
```

As with the faulty Port Lift-Thruster experiment both Navigation power supplies have been nominated as all their active supplied components have been nominated. Denomination has been deactivated and so they are simply added to the Suspicion Index. Related sub-components are then also added, producing a final component-level Suspicion Index of:

```
Component-Specific Diagnostic Level Entered
Adding Related Sub-Components to Suspicion Index:
Final Suspicion Index at Mission Time: 442.135
Sub-Component Diagnostics Level
AOSI_EZ3_COMPASS_SENSOR: 33
STBD_LIFT_THRUSTER: 32
STBD_LIFT_THRUST_CONTROL: 0
STBD_LIFT_MOTOR: 0
STBD_LIFT_THRUSTER_BRUSH: 0
STBD_LIFT_GBOX: 0
STBD_LIFT_PROPELLER: 0
PORT_LIFT_THRUSTER: 22
PORT_LIFT_THRUST_CONTROL: 0
```

```
PORT_LIFT_MOTOR: 0
PORT_LIFT_THRUSTER_BRUSH: 0
PORT_LIFT_GBOX: 0
PORT_LIFT_PROPELLER: 0
STBD_LAMP_CONTROL: 17
STBD_LEM_SENSOR: 10
PORT_LAMP_CONTROL: 7
PITCH_GYRO_SENSOR: 4
PORT_NAV_POWER: 1
STBD_NAV_POWER: 1
```

The component-specific diagnostic level is now entered and RECOVERY starts analysing the component-specific information attached to each component via the Relational Model. The information is passed to the relevant diagnostic tool, in this case the Dynamic Pattern Matcher, starting with the component with the highest Suspicion Index:

```
Dynamic Rule Testing of Component:
AOSI_EZ3_COMPASS_SENSOR Started
STBD_LIFT_THRUSTER Started
STBD_LIFT_THRUST_CONTROL Started
Exact Match Found
Exiting diagnostics
```

An exact match has been found with the known 'wire squeeze' fault attached to the Starboard Lift-Thruster's Control sub-component. This is a correct diagnosis, confirming that a highly suspicious component is actually faulty. This information would then be provided to the mission controller allowing it to accurately replan the mission.

7.2.10 Discussion of Full System Evaluation

The Relational Model proved to be a successful and versatile method for integrating different diagnostic tools and knowledge, whilst also providing an underlying

methodology for the novel domain-independent diagnostic tools. It has proved to be a powerful and versatile device which shows a great deal of promise.

The use of multiple dimensions, tied together by the Relational Model, also proved successful, with faults successfully being detected and diagnosed by different models. Combining the different diagnostic outputs across the dimensions resulted in the faulty component almost always having a higher suspicion index than other components.

Using different techniques for fault detection and diagnosis works well, with the use of simple rules and equational models being greatly enhanced when used in conjunction with the Relational Model. Unfortunately, the paucity of sensor information resulted in some faults being undetectable in some conditions. Some faults also appeared to be identical and so were misdiagnosed. Without any method of diagnosing environmental effects the overall performance of the system is also limited.

The novel domain-independent diagnostic tools worked well in conjunction with the conventional tools, often providing useful information and increasing the suspicion index of the faulty component. The use of the Hierarchy Module's denomination ability was found to be too high risk in real conditions and so was discarded. The Correlator was found to produce both relevant and irrelevant information. When the Correlator output was combined with the other diagnostic tool outputs the relevance aided overall system performance while the irrelevant simply contributed to the background 'noise'. The Correlator was found to occasionally miss important correlations due to having an overly narrow match window, although when this window was slightly widened the Correlator detected the relevant match at the cost of increased irrelevant nominations. But overall the novel domain-independent tools worked well under real conditions, particularly when the extremely simplistic implementation is taken into account.

The realisation that a specific fault may become apparent at different times in different dimensions came during the early stages of testing, leading to the development of the multiple iterations now used by RECOVERY. This was shown to work well in practice with faults being detected in different dimensions.

The breakdown of the components into component and sub-component types was

successful, reducing the need for the system to trawl through masses of component-specific information until a certain component could be picked out.

Overall, the full RECOVERY system works extremely well under real conditions. It successfully integrates the outputs of different diagnostic tools, including the novel domain-independent techniques developed during this research. It also successfully integrates heterogeneous diagnostic knowledge as used by the different diagnostic tools.

7.3 Evaluation of Search Space Reduction

This section evaluates RECOVERY's Search Space Reduction methods and compares the results and operation with a classical Dynamic Rulebase.

RECOVERY contains a heterogeneous knowledge base partitioned into component-specific regions. RECOVERY links these component-specific regions to specific components using links in the Relational Model so that when a component is nominated as suspicious then only the information attached to that component needs to be evaluated. This provides a method for reducing the amount of search space, or the amount of knowledge to be analysed, compared with analysing all the knowledge in the database.

A classical Rulebase does not have Search Space Reduction, it simply runs through the knowledge base evaluating one rule at a time until it either finds a match or runs out of rules. The time to find the correct rule will depend on the position of the rule in the knowledge base: it will take less time to find a rule at the beginning than the end.

A real mission log file as used in the Faulty Starboard Lift-Thruster experiment earlier in this chapter (Section 7.2.9) is used as the basis for the test. RECOVERY detection algorithms are run as standard until a fault is detected, at which point either the full RECOVERY system or a simple Rulebase are invoked.

In these tests RECOVERY's component-specific knowledge base is limited to dynamic rules and the same Dynamic Pattern Matcher Tool is used for both RECOVERY and Rulebase evaluation. A single 'correct' rule is inserted into the knowledge base at various locations. Each test is repeated three times to show repeatability and

to determine any inconsistencies due to the non real-time Windows NT4.0 operating system.

The metric for these tests is the time taken to find the correct rule.

7.3.1 Experiment Setup

Size of Rulebase: 62 Dynamic Rules

Rule at Start: Position 1

Rule at End: Position 62

RECOVERY: Settings are as detailed in Section 7.2.1 at the beginning of this chapter.

Denomination Disabled

Mission Log File: Full-System Faulty Starboard Lift-Thruster

7.3.2 Experiment: RECOVERY Versus Rulebase with Correct Rule at Start and End of Knowledge Base

These tests were performed on the Faulty Starboard Lift-Thruster Full System Evaluation mission log file, during which the component attached to the correct rule (Starboard Lift-Thruster Controller) was rated third in the suspicion index (see Section 7.2.9) and so was the third rule to be evaluated by RECOVERY. Improved times would be gained with a more accurate diagnosis.

Table 7.5 shows the time taken for a Classical Rulebase and the full RECOVERY system (incorporating Search Space Reduction) to find the correct dynamic rule. When the rule is at the start of the knowledge base the Rulebase is faster than RECOVERY due to RECOVERY's overhead. When the rule is at the end of the knowledge base the Classical Rulebase takes far longer to run as it must simply run each rule until it finds the correct one. In this case RECOVERY's slight overhead is grossly offset by the reduction in the size of the search space and so the amount of time taken for diagnosis.

The use of RECOVERY's search space reduction methods brings the diagnostic time down from 51 seconds with a classic rulebase to 8 seconds with RECOVERY.

System	*Rule Position*	*Run1 (Seconds)*	*Run2 (Seconds)*	*Run3 (Seconds)*
Rulebase	Start	2	2	1
RECOVERY	Start	9	8	8
Rulebase	End	51	51	51
RECOVERY	End	8	8	8

Table 7.5: Time in Seconds to Match a Dynamic Rule using a Simple Rulebase versus RECOVERY's Search Space Reduction

7.3.3 Discussion of Search Space Reduction

RECOVERY's Search Space Reduction method has proven to be extremely useful. The amount of information to be evaluated, and so both the time and power required for diagnosis has been substantially reduced over conventional methods.

Justification of Experiment

A classical Rulebase evaluating dynamic rules was used to compare with the RECOVERY method. When a classical Rulebase evaluates static rules the time taken to evaluate each rule is generally very low as the amount of information to be evaluated is small - it is an information poor technique. A dynamic rule takes a lot longer than a static rule to evaluate as the entire mission file must be scanned for each rule - it is an information rich technique.

It would have been possible to scan the entire mission file once and evaluate each dynamic rule in turn at each step, this would have been faster than scanning the entire mission file for each dynamic rule in turn. This technique was not used for two reasons:

1. A classical Rulebase, as used in most industrial or real-world situations, works by scanning each rule in turn and so this method was used for the comparison Rulebase.

2. RECOVERY's Search Space Reduction technique has a large overhead when compared with, say, a classical static Rulebase. It is accepted that RECOVERY's Search Space Reduction (with the current diagnostic tools) cannot match the speed

of a static rulebase - the overhead is not worth it.

But RECOVERY does have an advantage when working with large amounts of information. To show this a form of information was needed that took an appreciable amount of time to evaluate. Ideally, detailed equational models would have been used, but models take time to develop and developmental time was extremely constrained. Instead, dynamic rules were used because they take very little time to develop but take an appreciable amount of time to evaluate.

In effect, RECOVERY is being compared with a non search-space-reduction technique working in an information rich environment. The key point is to show that Search Space Reduction has great advantages in information-rich environments - the form the information takes is not important.

Advantages and Disadvantages

If looked at from a purely time-constrained point of view then if the search space is small, or if the knowledge available is information poor, then RECOVERY's overhead is generally not justified (although RECOVERY does provide other advantages, as discussed elsewhere).

But as the size of the search space increases, and as the knowledge becomes information rich, then RECOVERY's overhead becomes increasingly justified as it provides a means to drastically reduce the search space.

It should be noted that in the above experiment the search space is drastically reduced, from 62 dynamic rules down to 2, even though RECOVERY has not placed the correct faulty component at the top of the list. (The above experiment is based on the Faulty Starboard Lift-Thruster Experiment, which produced the final Suspicion Index shown in Section 7.2.9).

The evaluation of RECOVERY's Search Space Reduction method can be summarised as follows:

The larger the search space the greater the advantage of RECOVERY's Search Space Reduction method.

7.4 Discussion: The Limits of RECOVERY

The evaluation of RECOVERY has shown some limits and advantages, these will be discussed in this section. They break down into two forms: developmental and philosophical.

7.4.1 Developmental Limits

The current implementation of RECOVERY is research orientated and as such needs more development to overcome limits that are not inherent in the RECOVERY architecture.

The use of memory has not been optimised and the code is not written in the most efficient manner. There is no coherent time management strategy, which would be vital for use as a real-time online system. Although the system is modular it is not fully plug and play; if a new diagnostic tool is to be added then the code must be recompiled, although new items of supported information may be added at will. More types of both diagnostic knowledge and diagnostic tools also need to be supported.

The use of integer increments for the Suspicion Index is also overly primitive, particularly for interfacing probabilistic to other types of diagnostic tools.

7.4.2 Philosophical Limits

The RECOVERY architecture has some limits implicit in its design. The Relational Model can show only a limited number of relationships between nodes. The overall effectiveness of the system depends on the accuracy and scope of the Relational Model, which is unlikely to be totally accurate.

The Relational Model is presently unable to encompass the environment outside the robot, which is perhaps the biggest constraint on its effectiveness. Although the Relational Model is quite effective when representing a structured, man-made system such as a robot it cannot contain the complexity of the outside world.

Even if the number of nodes and types of relationships are extended it is unlikely to be able to reflect the true complexity of the real world, but it is still useful for

representing the most useful types of diagnostic relationships. The discrete, decoupled dimensional approach to multidimensionality also limits the overall usefulness of the system. It would be far more useful if the dimensions were coupled in a more realistic manner.

7.5 Summary of Chapter

This chapter presented the evaluation of both the full RECOVERY system and its method for Search Space Reduction. Both experiments showed strengths and weaknesses which were discussed after the experimental results were presented. This was followed by a short discussion on the limits imposed on the RECOVERY implementation by the constrained development time, and also the limits implicit in the philosophy of RECOVERY's architecture.

When presented with an environmental fault the various systemically-orientated diagnostic tools most often diagnosed the relevant sensor as being faulty, this is acceptable as none of the tools or models encompass the environment.

The use of denomination (or exoneration) was shown to be a high-risk strategy when presented with real failures, this practice was discontinued for the remainder of the experiments. Overall the full RECOVERY system was shown to work well on systemic faults, with the novel domain-independent diagnostic tools producing useful outputs on real faults.

RECOVERY's Focus of Suspicion method for search space reduction was shown to be extremely useful, with the advantages increasing with the size of the search space.

The next and final chapter draws conclusions about this research. Further work and ways to move the research forward are then described, followed by an overall summary of achievements.

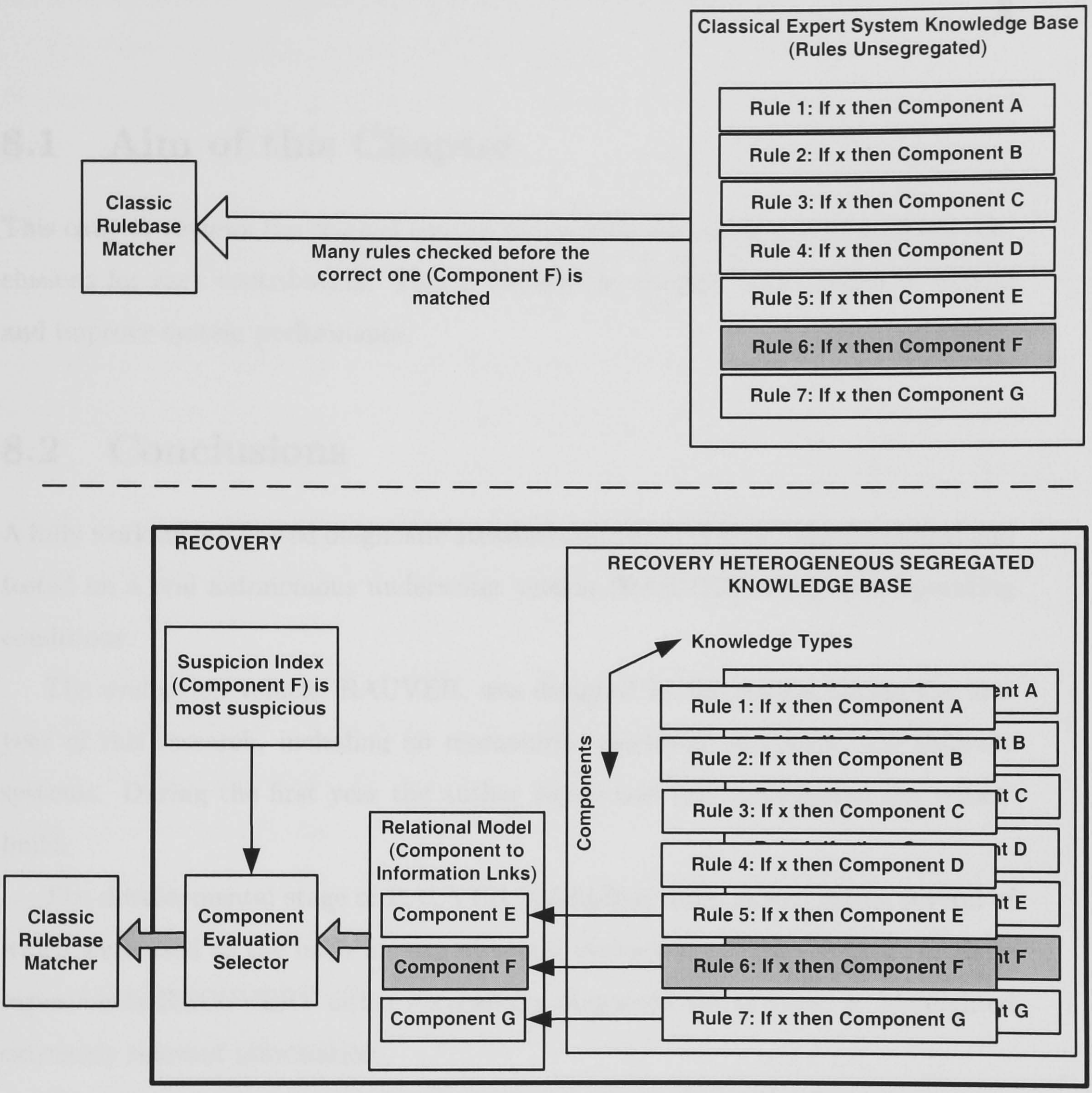


Figure 7.17: RECOVERY'S segmented knowledge database and a classical rule knowledge base showing that using RECOVERY's focus of suspicion method can reduce the search space.

Chapter 8

Conclusions & Further Work

8.1 Aim of this Chapter

This chapter reviews the original contributions from this research and presents conclusions for each contribution. This is followed by further work needed to extend and improve system performance.

8.2 Conclusions

A fully working integrated diagnostic architecture (RECOVERY) was developed and tested on a real autonomous underwater vehicle (RAUVER) under real operating conditions.

The evaluation vehicle, RAUVER, was designed by the author during the first year of this research, including all mechanical, electrical, electronic and software systems. During the first year the author supervised and coordinated the vehicle build.

The developmental stage of RAUVER highlighted some design faults, several of which were used as the basis for experimental evaluation of RECOVERY. In these experiments RECOVERY either successfully diagnosed the problem or highlighted extremely relevant information.

The adoption of a practically orientated development technique, together with the crude, simplest form of diagnostic tools and information forced by the time limit, has led to a successful working system of immediate practical benefit.

Several companies and other organisations have expressed interest in developing and commercialising RECOVERY.

8.2.1 Conclusions on the use of a Real Autonomous Robotic Vehicle (RAUVER) for Evaluation

The use of a real autonomous vehicle and real mission data rather than a simulation led directly to some alterations, discoveries and constraints:

The slow physical response of RAUVER led to the faults becoming apparent in different dimensions at different points in time. This led to the extension of RECOVERY from a single diagnostic iteration to several iterations followed by a post-iteration segment, a technique that worked well.

The need to keep processing power to a minimum became apparent when specifying the onboard processor and power systems for RAUVER. The tight constraints on physical space inside the pressure hulls has led to the only realistic choice being a PC104+ industrial computing card, the maximum speed available at the time of writing is a Pentium 300MHz. This card consumes 8 Watts of power. The low battery powers available at present mean that every Watt saved is significant, leading to some advanced vehicles such as the Autonomous Systems Institute 'SAUV' vehicle reverting to x486 processors to conserve power.

Because of these practical constraints it is not possible to simply throw more powerful processors at the problem. RECOVERY code was optimised during development to bring diagnostic times down to within 10 seconds from initial detection to diagnosis using a Pentium 200MHz processor.

The use of real mission data meant that there was no possibility of adjusting mission parameters to show the system in a favourable light. In some cases the noisy data caused RECOVERY to misdiagnose, forcing system alterations to limit the amount of irrelevant correlations produced, which later had the effect of missing crucial data.

RAUVER has a limited number of sensors onboard, as dictated by both development time and available money, a common situation when developing a vehicle. This meant that sensors could not simply be added into a simulation, RECOVERY development and evaluation had to make do with what was available. This led to the realisation of just how 'blind' a robotic vehicle really is and so to the adoption of the procedure of maximising available information for diagnosis.

Some of the sensors also proved to be noisy, leading to problems with model accuracy and the degradation of detection performance.

All these points assisted in the practical mindset adopted during RECOVERY development.

The use of extremely crude models for detection and diagnosis has shown that even these simple models can be extremely useful in a real, practical situation.

The use of RAUVER introduced some constraints on evaluation. Initially the Domain-Independent Hierarchy Module was to be evaluated by installing switches and selectively disabling combinations of power supplies. It became clear that this was not going to be possible due to the extremely limited development time available, the limited space and tight wiring constraints imposed by RAUVER construction (no room for the switches), and the unwillingness to repeatedly open RAUVER pressure hulls due to the possibility of a leak. It was hoped that the RAUVER maintenance overhaul period would fall well before RECOVERY evaluation and so allow at least some evaluation hardware to be installed, together with a GPS module for enhanced missions, but this was not possible due to time constraints on other Ocean Systems Personnel.

This was offset by the opportunity to use some real faults for RECOVERY evaluation.

8.2.2 Conclusions on the Literature Review

The literature review showed that the use of automated diagnostics for autonomous missions is strongly supported by NASA for both unmanned and manned missions, especially for missions that have a high chance of failure. Integrated diagnostics is a field of study that is meriting research by large, cutting-edge organisations such as NASA, General Electric and the United States military. These and other organisations conducted the Open Systems Approach to Integrated Diagnostics Demonstration (OSAIDD) study. This study was undertaken during the first year of this research, with results published during the second year.

RECOVERY is effectively a fully working and extended implementation of the vague conceptual specification produced by the OSAIDD study. RECOVERY's

Relational Model was shown to fulfill and extend the specification laid down in the OSAIDD study for a central informational model.

8.2.3 Conclusions on the Full RECOVERY System Evaluation

RECOVERY was shown to successfully integrate different types of diagnostic tools and diagnostic knowledge. RECOVERY successfully diagnosed faults using a Model-Based Diagnosis Engine in conjunction with novel Domain-Independent Diagnostics acting on RECOVERY's Relational Model.

The diagnostic capability (and so mission robustness) of an AUV can be increased by adding cheap, internal sensors to provide additional diagnostic viewpoints.

Clear benefits are attainable by using different types of diagnosis and multi-dimensional viewpoints together with a mixture of domain-specific and domain-independent information. With these methods it is possible to diagnose specific faults using general information.

It is essential for some method to be found to evaluate performance of tools under different conditions so that some measure of relevance can be found. This information should be part of the 'plug' for the tools: 'invoke me under these conditions but not others'. It should also apply to diagnostic knowledge, for instance the roll model is only useful when the vehicle is submerged as it does not take into account wave action.

Much useful heterogeneous information is generated as part of the usual AUV design process, most of which is not generally used for automated diagnosis. For instance, RAUVER documentation consists mainly of paper copies of circuit diagrams and component data sheets, none of which would normally be used for diagnostics. Any system that tries to make use of this information needs to be modular to cope with the wide variety of configurations available to AUVs.

The use of embedded diagnostics is also useful. If no diagnostic system had been present in RAUVER during these faults then the only fault detection would have been alarms and violated mission constraints. The diagnosis, such as it was, would be limited to 'roll too high' with no ability to determine that it was actually

a thruster at fault, let alone a particular thruster.

Using a simple systems such as a classical Rulebase the programmer would have had to foresee many specific events and encode them as rules, such as 'if vehicle rolls to port when vehicle moving vertically then port lift thruster is faulty'. Whilst this is useful for the more common, foreseeable faults it is of limited use for real autonomy as every situation must be predicted. RECOVERY showed that it could diagnose specific faults using only general system information.

The ability to diagnose specific faults using non fault-specific commonly available information, coupled with the ability to significantly reduce the search space, shows that the design and run-time overhead of the RECOVERY system is worthwhile. When coupled with an autonomous mission controller such as a planner, or even a mission script, it has the potential to enhance robotic autonomy by providing enhanced fault information.

8.2.4 Conclusions on RECOVERY's Search Space Reduction Method

RECOVERY proved able to provide a high Suspicion Index for the faulty component under most conditions.

RECOVERY's Focus of Suspicion method for Search Space Reduction was shown to be able to greatly reduce the diagnostic search space, and so the amount of information to be analysed. This can greatly reduce the time (from 51 seconds down to 8 with a reasonable sized search space), number of processor operations and electrical power required to arrive at a diagnosis, even if the information consists only of rules.

8.2.5 Conclusions on the use of Domain-Independent Diagnostics

Domain-Independent Diagnostics work in at least two areas: General Correlations and Hierarchical Diagnostics. These work on both the behaviour of faulty parameters and the outputs of different types of diagnostic tool. The Relational Model

should enable other types of Domain-Independent Diagnostics to be developed.

The use of denomination is risky and should be avoided in order to prevent the exclusion of a faulty component from the diagnostic search space.

To increase the amount of diagnostic knowledge and to enable Domain-Independent Hierarchical diagnostics to work properly every parameter should have detection information attached to it, even if this information is simple maximum/minimum limits.

If a source component, such as a power supply, does not have direct sensing of voltage then subcomponents must be active and detectable at the time of the fault in order to diagnose the failure of the power supply.

Overall the domain-independent diagnostic tools work very well in conjunction with RECOVERY's Relational Model.

8.3 Further Work

This section discusses ideas for enhancing and extending RECOVERY's performance.

8.3.1 Practical Work

The system must be tuned to reduce processor overheads, memory requirements and run-time in order to minimise power consumption and diagnostic time. This will make it more attractive to heavily constrained autonomous robots.

8.3.2 Diagnostic Performance Metrics

This thesis does not provide figures of merit for RECOVERY performance, apart from the time reduction of the Search Space Reduction methods. This is because RECOVERY is at a very early stage of development and so it is preferable to talk through each success and failure, explaining the issues involved and highlighted. A raw figure of merit would not show this.

But figures of merit, and other diagnostic performance metrics, are essential if RECOVERY is to be compared to other systems in the future. They are particularly

important for convincing other organisations of the benefits of a particular system as they provide objective summaries of system performance.

Standard diagnostic metrics exist for conventional systems, these are the rates of Fault Detection, Fault Isolation and False Alarms. These are insufficient in scope for evaluating the next generation of Integrated Diagnostics, particularly embedded (or fielded) systems such as RECOVERY. James Bohr [9] addresses these issues and provides and extended metric set. David Doel [34] proposes a systems-engineering approach.

8.3.3 Mission Control

The mission controller should be linked into the Relational Model so that the components used are the same throughout both diagnostic and mission control areas. Extra links could be added to show which goals use which components. This would aid in detection, diagnosis and mission replanning.

8.3.4 Fault Accommodation

RECOVERY's operation should be extended to encompass fault accommodation as well as detection and diagnosis. Although it is the mission controller's responsibility to determine whether to replan or abort the mission, the diagnostic system needs to be able to cope with degraded vehicle performance in case of mission continuation.

This means being able to alter the models used for detection and diagnosis so that RECOVERY is not in a continuous state of diagnosis during a degraded mission. Various techniques exist for automatically generating and modifying models (as discussed in Chapter 3) these could be added to RECOVERY in a 'post-diagnostic accommodation' stage.

The models currently used for detection and diagnosis could also be used for vehicle control. For instance, the models could be used to calculate the thrust required to lift the vehicle at a particular rate (currently done with a simple PID controller). When the models are degraded the performance of the vehicle is automatically compensated for. In the 'dead lift-thruster' example used for evaluation the models could be used to solve the constrained movement problem of lifting within a certain

amount of roll.

8.3.5 Integrating Diagnostic Tools

Far more research is needed on the subject of integrating, particularly as to how different diagnostic tools should be weighted, how secondary diagnostic information should be weighted, and how diagnostic tool outputs should be weighted under different operating circumstances.

One promising method of interfacing tools with different characteristics is context-dependent voting. For instance, some tools work very well when the system is under steady state conditions but badly when the system is in flux. Other tools have the opposite characteristics. By combining information on the how the performance of tools changes with system context, together with information on the current context of the system, diagnostic performance should be much improved.

A 'cheap' but powerful solution to this problem is to use fuzzy context weighting. Homayoun Seraji *et al* of NASA/JPL use this method in a forthcoming paper to adjust the weighting of different sensors on a 'landing site judger' for a Mars lander. For instance, radar works best when at low altitude, video cameras work very poorly during dust storms, etc. By using context-specific fuzzy weighting of these sensors they have greatly improved the performance of the system responsible for judging the suitability of landing sites.

Some form of formal metric needs to be developed that can determine the relevance of the different integrated diagnostic tools under different conditions.

Goebel, Krok and Sutherland [44] cite the difficulty of integrating probabilistic diagnostic methods, such as Neural Networks, with non-probabilistic, such as Model-Based Diagnosis methods. They propose that the output of all tools should be constrained to a non-linear 'belief' value, ranging from 0 to 1. This system could easily be adopted on RECOVERY, but questions remain on interfacing. In the meantime the use of Suspicion Index nomination weighting should be investigated.

There is a great deal of commonality between integrating (fusing) diagnostic tools and sensor fusion. RECOVERY can be seen as fusion architecture, fusing sensors, diagnostic tools, and diagnostic knowledge. Sensor fusion is currently a

major research area and the techniques and knowledge gained should be applied to the fusion of diagnostic tools and knowledge.

8.3.6 Machine Learning

The Correlator can effectively generate new relational links between nodes although it does not actually do this at present. This is a powerful ability which should be further investigated as it raises the possibility of being able to learn from previous faults.

8.3.7 Multiple Robots

There is a strong research push towards using multiple cooperating robots. This has interesting implications for automated diagnosis as it provides the possibility of using multiple, external points of view. This should greatly enhance the diagnostic capability by providing a much-expanded observation space. RECOVERY's Relational Model could be extended to encompass observations from outside the vehicle.

This provides a method of diagnosing problems in the environment. For instance, if several vehicles in group are suffering from drift then it is likely to be an environmental effect such as a water current and not a sensor fault in a particular vehicle.

8.3.8 Distributed Modularity

The use of multiple robots, together with the ongoing push towards 'intelligent' components running their own processors, means that in the future diagnostic systems such as RECOVERY will be needed that can cope with distributed modularity. This will be made more difficult by different robots and systems running at different speeds and different times, some on real-time and some on other operating systems.

8.3.9 Active Testing

RECOVERY currently acts as a passive observer, unable to control the vehicle. Many faults require active diagnosis to solve them, such as providing an input to a suspect component and observing the output. For instance, if it is thought that a given thruster is faulty then tests could be used to discount possibilities such as the vehicle being entangled in a rope, etc.

8.3.10 Extending the Relational Model

The Relational Model's links currently only state that there is a relationship of a certain type between two nodes, without giving any more information. Information should be added to each link to describe its attributes. For instance, a link between a power supply and a supplied component could contain information on the resistance of the lead (an issue in high-power circuits), further increasing the information available for diagnosis.

At present the Relational Model breakdown is based on least replaceable units, but for diagnosis, as opposed to fixing by a human operator, this may not be the best technique. Other techniques should be investigated to determine the best level of breakdown into components and subcomponents.

The types of nodes and links in the Relational Model should also be extended to reflect a more diverse selection of components, parameters and relationships.

8.3.11 Extending Diagnostic Knowledge

RECOVERY should support more types of diagnostic knowledge. At present it supports algorithmic models, static rules, dynamic rules and Domain-Independent rules. Other types of available knowledge are: training sets for neural networks, Mean Time Between Failures, characteristics graphs (as in most semiconductor datasheets) and so on.

For instance, RAUVER has a problem with its water detectors decaying and providing a false alarm every few months, as shown in the RAUVER Fault Log (Section 6.2.1. This information is clearly useful for diagnosis but is not currently implemented in RECOVERY.

The present knowledge space is segmented into two layers: system-specific (roll model) and component-specific (a thruster model). Another layer may be type-specific knowledge as several components of the same type are often used within a system (RAUVER has two sets of identical thrusters). Further research into the most efficient segmentation of the knowledge space should be conducted.

Due to developmental time constraints RECOVERY currently supports only static and dynamic rules at the component-specific stage. This was found to be more than enough for demonstrating the effectiveness of RECOVERY's Search Space Reduction method. The use of equational models at this stage would have amplified the effectiveness of the Search Space Reduction method and would also have allowed the use of hierarchical-detail diagnostic operation.

8.3.12 Extending Diagnostic Tools

Differing types of Diagnosis Tool, such as Bayesian Belief Networks or Neural Networks should be added to study how the system is enhanced with these powerful methods.

8.3.13 Introducing a Real-Time Diagnostic Strategy

In order to meet real-time constraints a coherent time-management strategy should be implemented. Aldea [2] demonstrated a successful system using a hierarchy of models, with the most detailed models taking the longest to run. The system could decide which model to run depending on how much time was available for diagnosis. This could be extended to cover all the information stored in the knowledge space, not just models, so allowing RECOVERY to meet deadlines.

8.3.14 Extending Domain-Independent Diagnostics

The Domain-Independent Hierarchy Module should be extended to run over several iterations, enabling it to move up more than one step in the Relational Model source/sink hierarchy. For instance, at present the Hierarchy Module can only move one step up the hierarchy, from faulty alarms to nominating the 5V power supply (which supplies them). If the batteries were faulty and so more than one power

supply seemed faulty the Hierarchy Module would not be able to move another step up the hierarchy to the batteries (which supply all the power supplies).

Some limit would be needed on the number of iterations, a provisional limit would be to stop when no more components are being nominated. If a fixed number of iterations was used there would be a risk that hierarchical diagnostics would be terminated before reaching all the way up the fault chain.

Further work is also needed on the technique of denomination. Denominating exonerated components could further reduce the search space, but this is currently too high a risk due to the crudity of the Domain-Independent diagnostic techniques.

Currently only a few items of Domain-Independent diagnostic information are used. This should be increased to cover other equivalent relationships. For instance, a resistor in an electrical circuit is equivalent to a constriction in a hydraulic pipe or to friction in a gearbox.

8.4 Overall Summary of Research

This research has attained a number of achievements, as listed below.

- The design of a novel integrated diagnostic architecture and its implementation as a fully working system known as RECOVERY.

- As far as is known RECOVERY is ahead of any other system for integrated diagnostics. The fact that the multi-body OSAIDD study, encompassing NASA, US DoD, General Electric, etc, only made vague conceptual recommendations in 1999 strongly supports this.

- RECOVERY is a full working system that has been successfully implemented on a real AUV, used to diagnose real faults discovered in real-world conditions.

- The implementation of RECOVERY highlighted previously unknown possibilities for integrated diagnostics, such as search space reduction, domain-independent diagnostics and the ability to diagnose non directly-sensed components.

- RECOVERY was shown to be more than a method of overlaying diagnostic tool outputs. It is also structured to allow the use of novel domain-independent diagnostic tools and successful techniques for search space reduction.

- The ability to use cheap commonly available (and commonly discarded) design knowledge, such as hydrodynamic models, for fault diagnosis was shown.

- RECOVERY was shown to be able to successfully diagnose real problems without fault-specific information.

- RECOVERY's architecture allows diagnosis of components that are not directly sensed.

- RECOVERY's method of search space reduction shows clear benefits over conventional systems, with the benefit increasing with the size of the search space.

- RECOVERY is of immediate practical benefit to the autonomous robot community.

- Embedded integrated diagnostics can enhance mission controller performance by providing enhanced information on unforeseen problems.

- Further academic research based on RECOVERY is already being proposed.

- The commercialisation and research prospects show that RECOVERY has much potential.

Appendix A

Glossary of Terms

A.1 Diagnostic Terms

- AUV: Autonomous Underwater Vehicle. An underwater robot that can act without outside intervention. Most commonly a torpedo shaped survey submarine at present, but AUVs capable of interacting with submerged objects (intervention) are in development.

- Central Information Model: Proposed by the OSAIDD study as the key component in an underlying architecture for Integrated Diagnostics.

- Diagnostic Information Fusion: A branch of Integrated Diagnostics which concentrates on providing underlying methods for combining the outputs of different diagnostics systems.

- Diagnostic Tool: A diagnostic system.

- Dimension: An area in which a fault is apparent, for instance a fault that causes an aeroplane to roll is apparent in the movement dimension.

- Dynamic: A fault that can only be diagnosed by looking at the state of a system over time.

- FDDR: Fault Detection, Diagnosis and Recovery. Also known as FDDA, or Fault Detection, Diagnosis and Accommodation. The overall diagnostic process where a fault is detected, diagnosed and recovered from.

- FDI: Fault Detection and Isolation. A diagnostic system that can diagnose a fault but not recover from it.

- Finite State Machine: Any program (or other device) that has a finite set of states. In the context of this thesis it means a pre-programmed mission in which the mission programmer has tried to anticipate all possible outcomes.

- Generality: The ability to generalise from specific examples.

- Heterogeneous: Encompassing different types of diagnostic tool.

- Homogeneous: Encompassing only a single type of diagnostic tool.

- Incipient: A slowly developing fault.

- Integrated Diagnostics: A process by which large organisations, particularly the US Military, aim to reduce life cycle costs of their systems by reducing maintenance through improved diagnostics. It has a logistical focus at present.

- MetOcean: A branch of meteorological forecasting focussed on the state of the ocean.

- Model-Based Diagnosis: A form of diagnostic system that uses models of the system under investigation.

- Monodimensional: A model or process that deals with only a single aspect of a system, for instance movement, or power consumption.

- Multidimensional: A model or process that deals with more than one aspect of a system.

- Observability: Price [84] defines observability as *"There should be sufficient observation points that inconsistent results to tests can be detected ... and that sufficient data can be gathered to be able to perform diagnosis to the most accurate level desired..."*. Diagnostic systems generally concentrate on observing (sensing) system parameters, it is essential that there are enough observations for these systems to work properly.

- OSAIDD: The Open Systems Approach to Integrated Diagnostics Demonstration Study. A far reaching multi-organisational study that determined the way forwards for Integrated Diagnostics.

- Prognosis: The expected outcome of a fault.
- ROV: Remotely Operated Vehicle: Human controlled underwater robot, usually supplied with power and control from the surface via a cable or tether.
- Rulebase: A form of expert system that uses a knowledge base of rules in an 'if-then' form.
- Static: A fault that can be diagnosed from looking at the state of a system at a single instant in time (a snapshot).
- Topology: The relationship between nodes in a network.
- Waypoint: A common method of specifying a route that an agent (AUV, etc) must follow.

A.2 RECOVERY-Specific Terms

- Activity Status: A data slot in the Nodal Information Database that store a flag denoting whether that particular component (node) is active at that time.
- Activity Status Generator: The module that determines the activity of each component using information provided in that component's information file.
- Alarm Watcher: The module that scans all alarm parameters to see if they are activated. Also scans all parameters that have max/min limits associated with them to see if they are outside their limits.
- Compiler: The module that takes the Component Information Files and compiles the information contained into the Relational Model and the Segregated Diagnostic Knowledge Database.
- Component-Specific Information: Information relevant to a particular component.
- Component Information File: Information file containing diagnostic data relevant to a specific component. Also used to store system information, such as the relationships between components.

- Constraint Watcher: The module that watches for information passed from the mission controller about any parameters that have exceeded goal-specific limits.

- Correlator: A RECOVERY diagnostic tool that searches for correlations between faulty and other parameters.

- Data & dt Values: Data slots in the Nodal Information Database that store parameter values and their generated differentials.

- Data Copier: The module that copies data from the robotic vehicle's data ring and stores it in the appropriate Nodal Information Database data slot.

- Data Generator: The module that automatically generates differentials of all data copied from the vehicle's data ring. Also generates Activity Status of each component.

- Data Manager: Manages the retrieval of information from the Relational Model and Segregated Diagnostic Knowledge Database. Also passes the information to any module that request it.

- Domain-Independent Diagnostics: Diagnostic tools that exploit the equivalence between domains and so are not limited to a single domain. For instance, resistance in the electrical domain is equivalent to a constriction in a hydraulic system. RECOVERY provides methods for domain-independent diagnostics.

- Fault Detection: The suite of detection tools that detect faults.

- Final List Maker: Produces a list of suspicious components ranked in order of suspicion.

- Focus of Suspicion: Method by which the diagnostics search space is reduced.

- Hierarchy Module: A RECOVERY diagnostic tool that searches for common connections between nominated components.

- Invocation Manager: Determines which module should be invoked, and invokes them.

- Iteration Counter: Counts the diagnostic iterations and provides this information to the Invocation Manager.

- List of Diagnostic Tools: Stored in the Invocation Manager to allow it to invoke the diagnostic tools.

- List of Links: Master list of links, containing endpoint identifiers (Node IDs) for each link.

- Master List of Nodes: Master list of Node Identifiers, used to index and access information in the Segregated Diagnostic Knowledge Database.

- Mission Controller: The module responsible for determining the actions of the robot.

- Nodal Information Database: Part of the Relational Model. Contains multiple data slots for each node.

- Node Type: A data slot in the Nodal Information Database that stores the type of that node, such as component, parameter, etc.

- Nominator: The module that increments the Suspicion Index data slot of nominated components (nodes).

- RECOVERY: The architecture (and also the implementation) developed during the course of this research to provide an underlying architecture for Integrated Diagnostics. It is a more detailed, fully working independently generated example of the model proposed by the OSAIDD study.

- Relational Model: The central and key component of the RECOVERY architecture, similar to the Central Information Model proposed by the OSAIDD study.

- Residual Watcher: The module that checks for residuals between modelled and sensed parameters. These are linked in the Relational Model.

- Search Space Reduction: The method by which the diagnostic search space is reduced.

- Segregated Diagnostic Knowledge Database: Stores all diagnostic knowledge, such as rules or equational models. Segregated by component to allow for Search Space Reduction.

- Status Generation: The group of modules responsible for copying and generating vehicle status data.

- Suspicion Index: The key data slot in the Nodal Information Database, it provides a way to reduce the search space.

- System-Specific Information: Information that describes the system, such as hydrodynamic models of the vehicle and the relationships between components.

- Thruster: A component used for movement and positioning of an AUV. The RAUVER vehicle used in this research has 2 main-thrusters (for forward movement and yaw control) and 2 lift-thrusters (for vertical movement and roll control).

- Vehicle Data Ring (Sensor Data): The robotic vehicle's data store, often a network ring.

Appendix B

Full Relational Model of RAUVER

B.1 Graphical Representation of the Relational Model

This section shows a graphical representation of the full Relational Model of RAUVER used to evaluate this research. The relevant text input to the compiler is shown at the bottom of each figure, the diagrams are manually generated from that information.

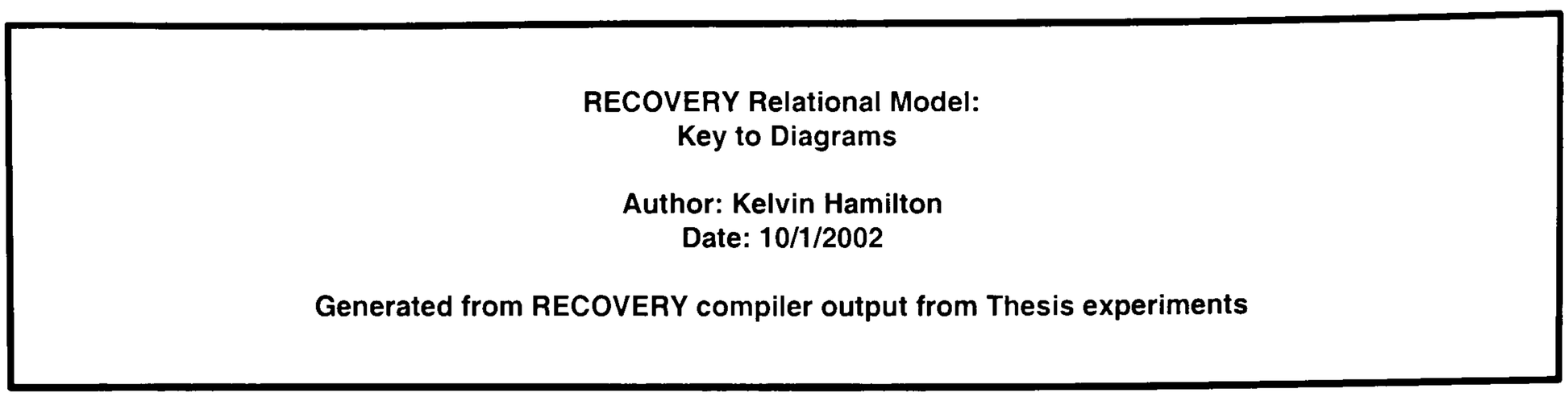
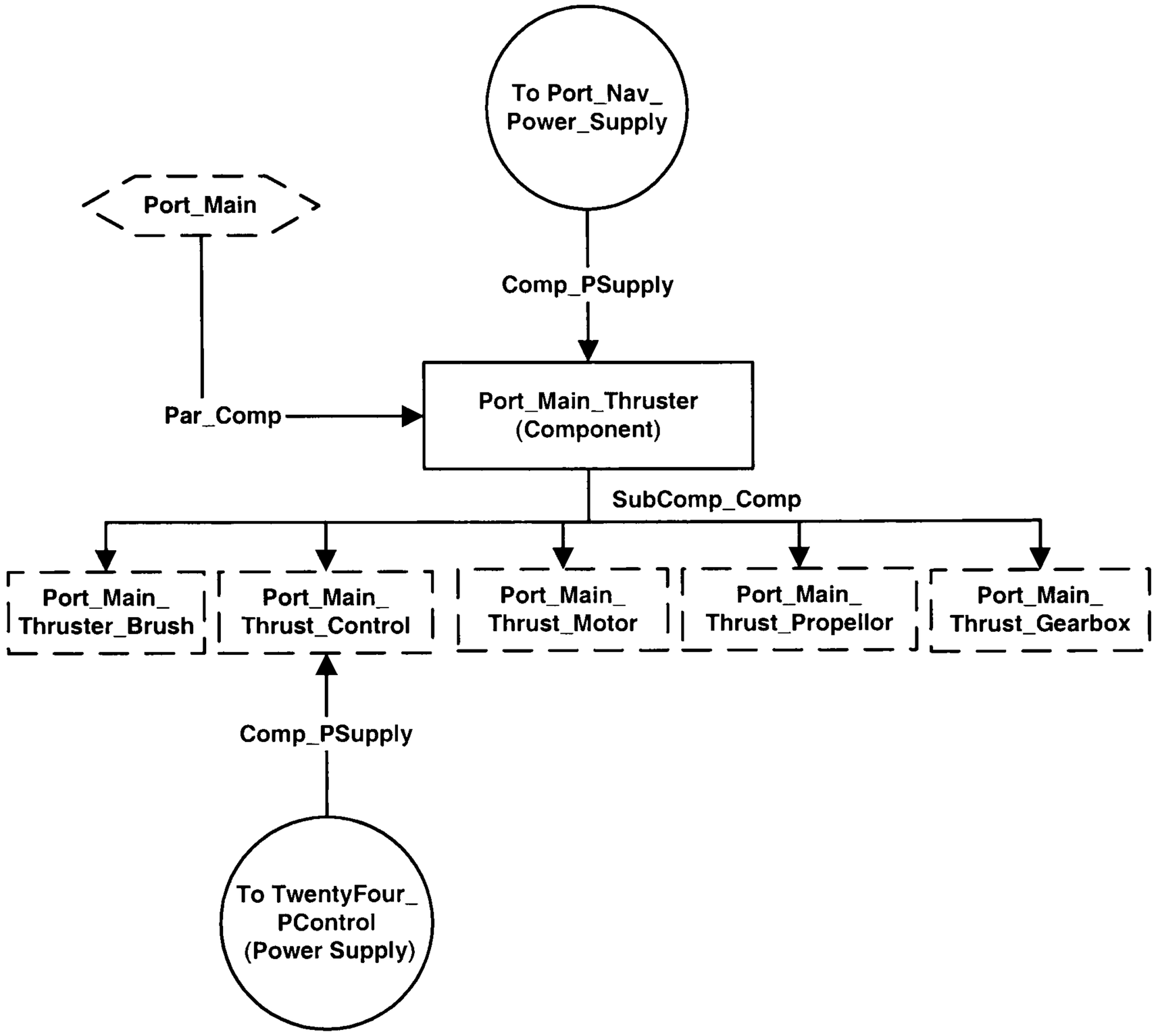


Figure B.1: Key to Relational Model Diagrams

RECOVERY Relational Model:
Port Main-Thruster

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



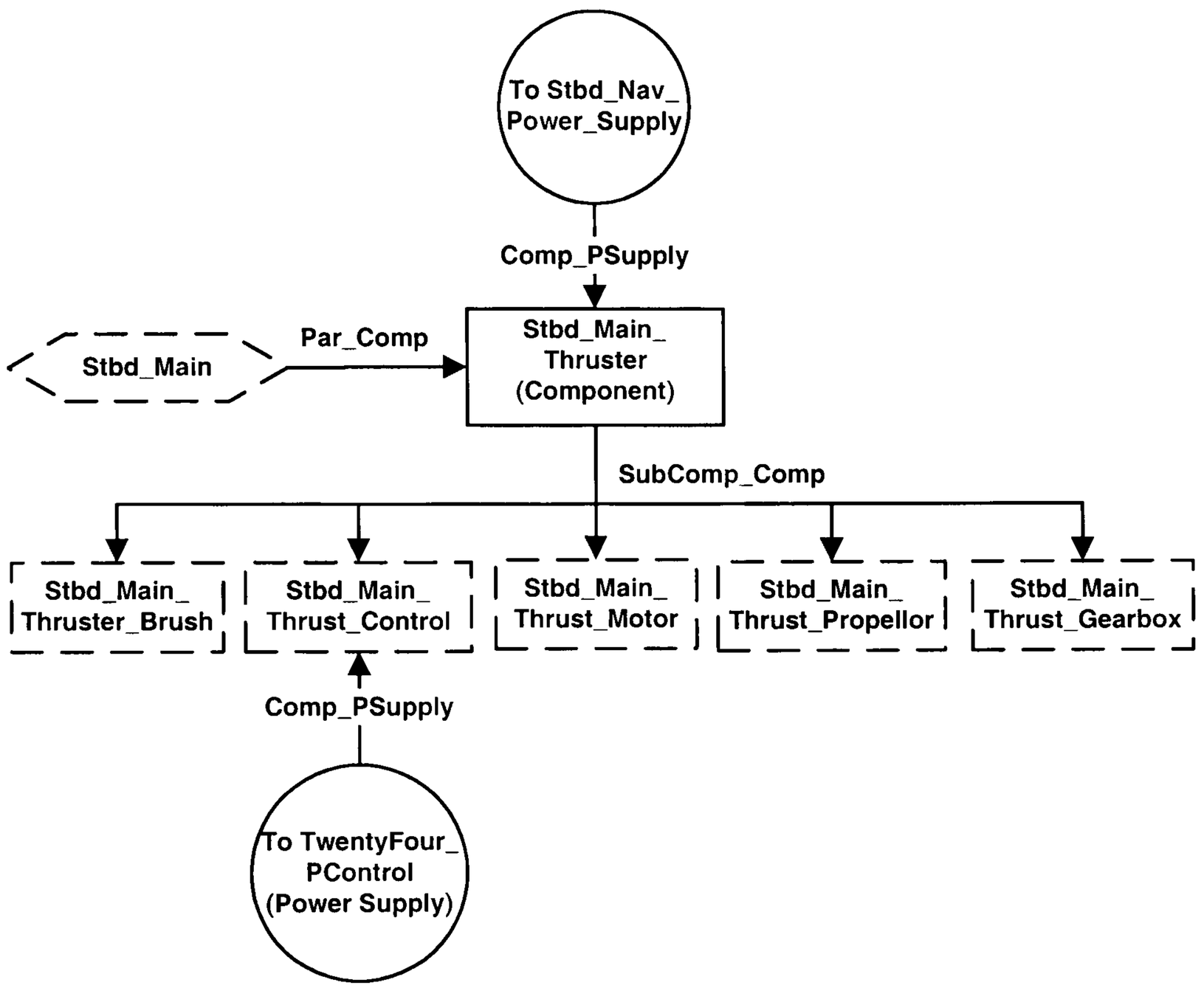

```
LINK SUBCOMP_COMP PORT_MAIN_THRUST_CONTROL PORT_MAIN_THRUSTER;
LINK SUBCOMP_COMP PORT_MAIN_MOTOR PORT_MAIN_THRUSTER;
LINK SUBCOMP_COMP PORT_MAIN_THRUSTER_BRUSH PORT_MAIN_THRUSTER;
LINK SUBCOMP_COMP PORT_MAIN_GBOX PORT_MAIN_THRUSTER;
LINK SUBCOMP_COMP PORT_MAIN_PROPELLER PORT_MAIN_THRUSTER;

// parameters to components
LINK PAR_COMP PORT_MAIN PORT_MAIN_THRUSTER;

// components to power supplies
LINK COMP_PSUPPLY PORT_MAIN_THRUSTER PORT_NAV_POWER;
LINK COMP_PSUPPLY PORT_MAIN_THRUST_CONTROL
TWENTYFOUR_PCONTROL;
```

Figure B.2: Port Main-Thruster Module

RECOVERY Relational Model:
Stbd Main-Thruster

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



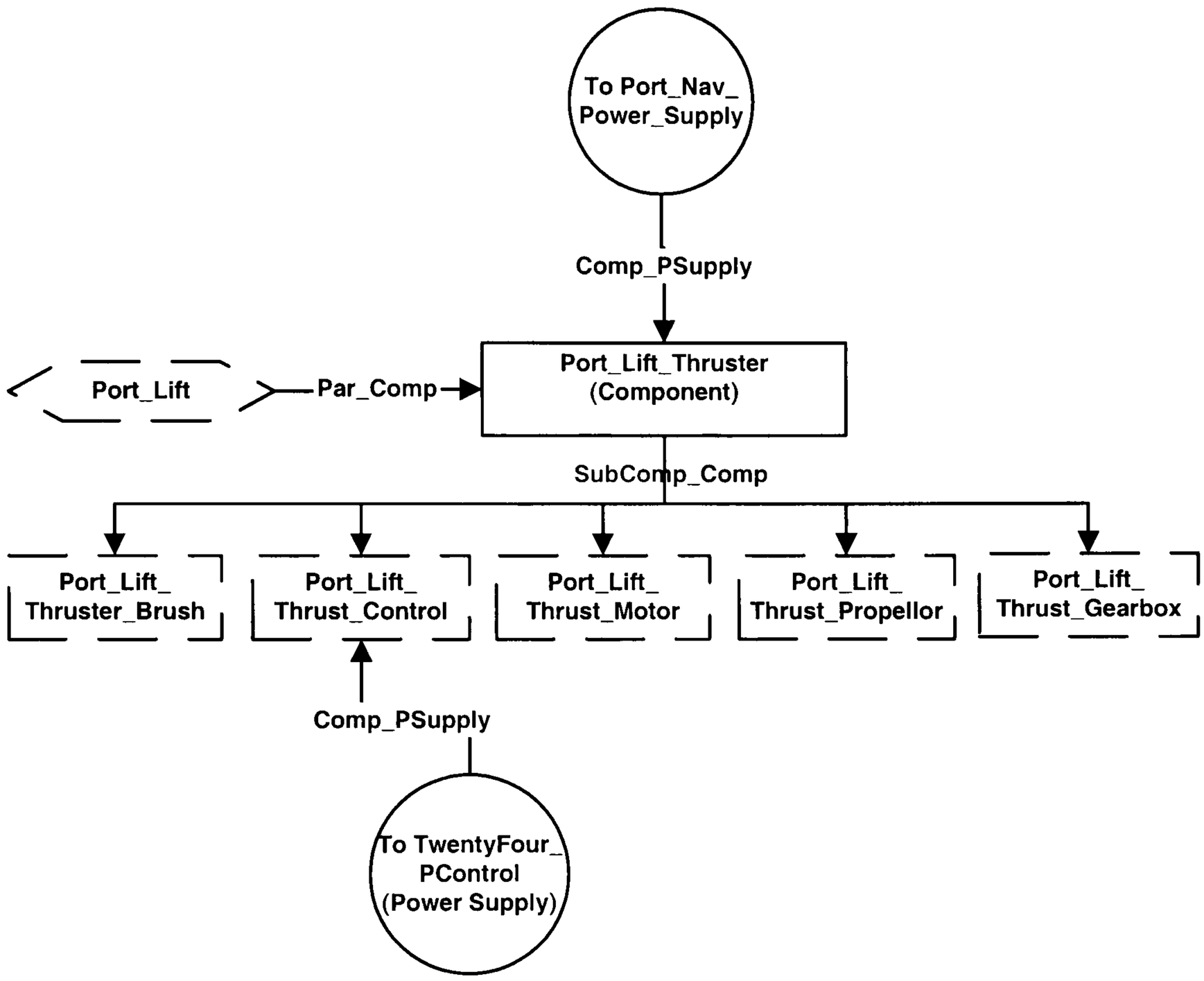

```
// sub components to components
LINK SUBCOMP_COMP STBD_MAIN_THRUST_CONTROL STBD_MAIN_THRUSTER;
LINK SUBCOMP_COMP STBD_MAIN_MOTOR STBD_MAIN_THRUSTER;
LINK SUBCOMP_COMP STBD_MAIN_THRUSTER_BRUSH STBD_MAIN_THRUSTER;
LINK SUBCOMP_COMP STBD_MAIN_GBOX STBD_MAIN_THRUSTER;
LINK SUBCOMP_COMP STBD_MAIN_PROPELLER STBD_MAIN_THRUSTER;

// parameters to components
LINK PAR_COMP STBD_MAIN STBD_MAIN_THRUSTER;

// components to power supplies
LINK COMP_PSUPPLY STBD_MAIN_THRUSTER STBD_NAV_POWER;
LINK COMP_PSUPPLY STBD_MAIN_THRUST_CONTROL
    TWENTYFOUR_PCONTROL;
```

Figure B.3: Starboard Main-Thruster Module

RECOVERY Relational Model:
Port Lift-Thruster

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



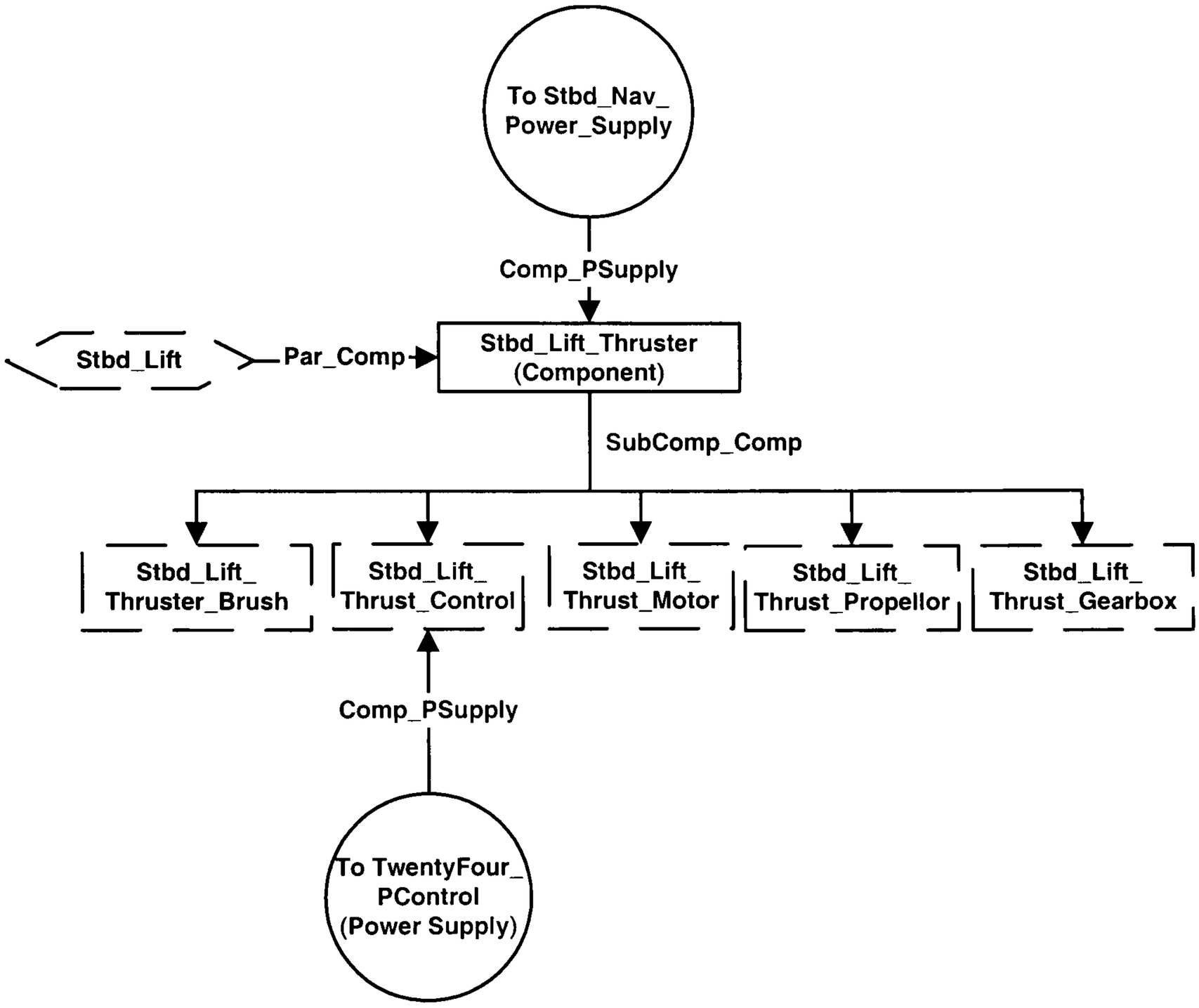

```
// sub components to components
LINK SUBCOMP_COMP PORT_LIFT_THRUST_CONTROL PORT_LIFT_THRUSTER;
LINK SUBCOMP_COMP PORT_LIFT_MOTOR PORT_LIFT_THRUSTER;
LINK SUBCOMP_COMP PORT_LIFT_THRUSTER_BRUSH PORT_LIFT_THRUSTER;
LINK SUBCOMP_COMP PORT_LIFT_GBOX PORT_LIFT_THRUSTER;
LINK SUBCOMP_COMP PORT_LIFT_PROPELLER PORT_LIFT_THRUSTER;

// parameters to components
LINK PAR_COMP PORT_LIFT PORT_LIFT_THRUSTER;

// components to power supplies
LINK COMP_PSUPPLY PORT_LIFT_THRUSTER PORT_NAV_POWER;
LINK COMP_PSUPPLY PORT_LIFT_THRUST_CONTROL
TWENTYFOUR_PCONTROL;
```

Figure B.4: Port Lift-Thruster Module

RECOVERY Relational Model:
Stbd Lift-Thruster

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



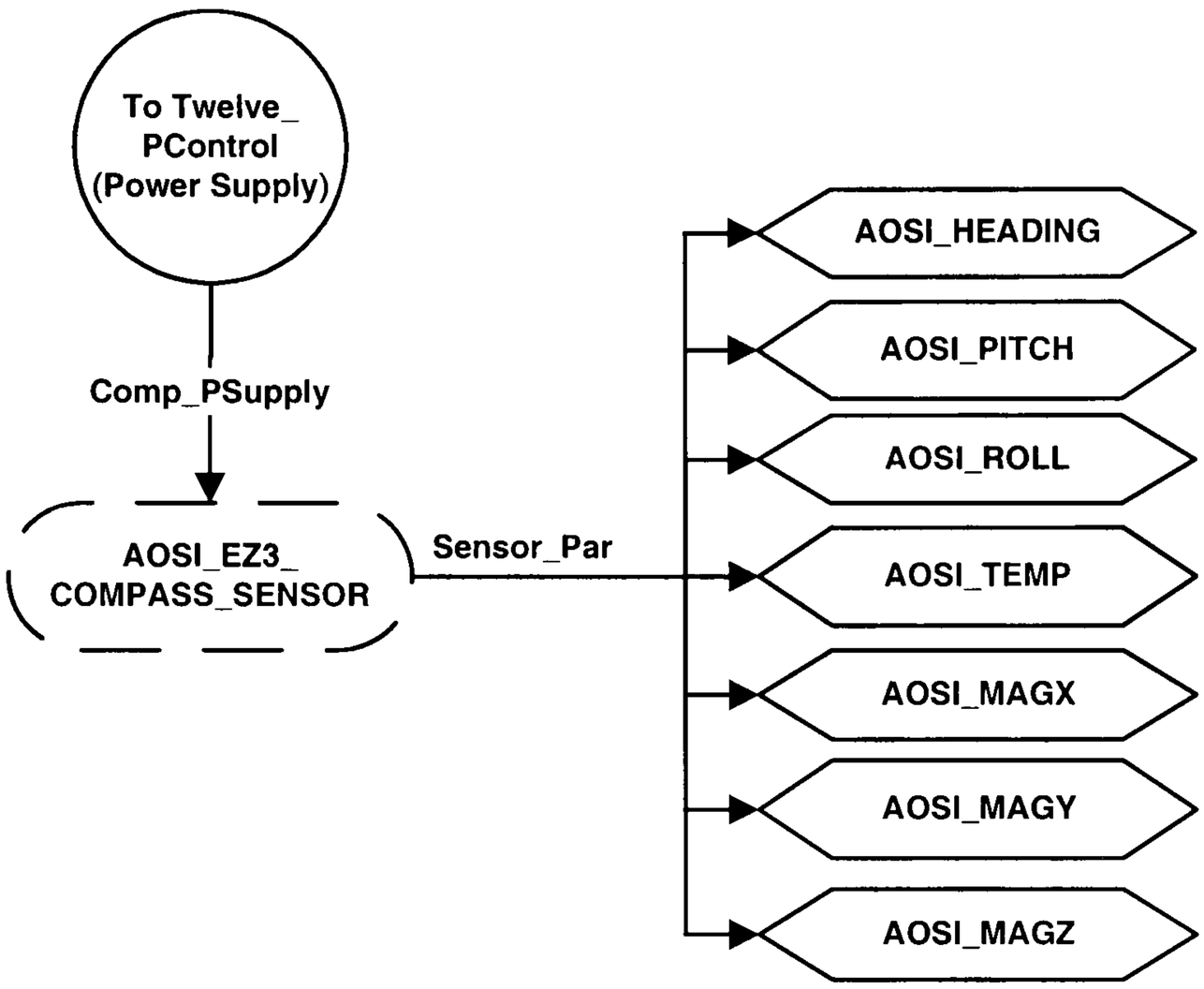

```
// sub components to components
LINK SUBCOMP_COMP STBD_LIFT_THRUST_CONTROL STBD_LIFT_THRUSTER;
LINK SUBCOMP_COMP STBD_LIFT_MOTOR STBD_LIFT_THRUSTER;
LINK SUBCOMP_COMP STBD_LIFT_THRUSTER_BRUSH STBD_LIFT_THRUSTER;
LINK SUBCOMP_COMP STBD_LIFT_GBOX STBD_LIFT_THRUSTER;
LINK SUBCOMP_COMP STBD_LIFT_PROPELLER STBD_LIFT_THRUSTER;

// parameters to components
LINK PAR_COMP STBD_LIFT STBD_LIFT_THRUSTER;

// components to power supplies
LINK COMP_PSUPPLY STBD_LIFT_THRUSTER STBD_NAV_POWER;
LINK COMP_PSUPPLY STBD_LIFT_THRUST_CONTROL
    TWENTYFOUR_PCONTROL;
```

Figure B.5: Starboard Lift-Thruster Module

RECOVERY Relational Model:
AOSI EZ-Compass-3 Compass/Attitude Sensor Module

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



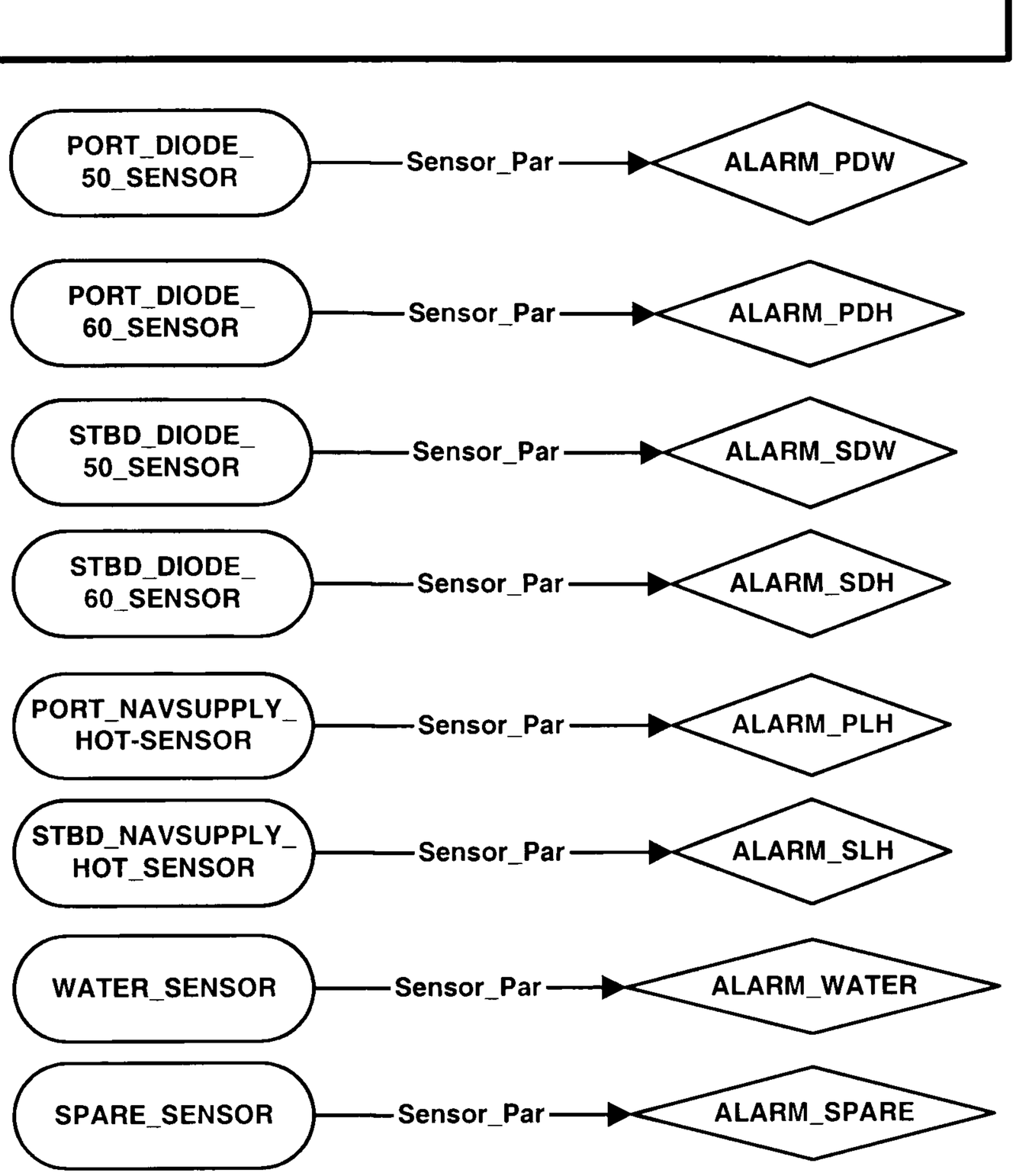

```
// *************** Links: ***************
                    //     SENSOR_PAR,
LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_HEADING;
  LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_PITCH;
  LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_ROLL;
  LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_TEMP;
  LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_MAGX;
  LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_MAGY;
  LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_MAGZ;

                    //     COMP_PSUPPLY,
        LINK COMP_PSUPPLY AOSI_EZ3_COMPASS_SENSOR
                    TWELVE_PCONTROL;
```

Figure B.6: AOSI EZ-Compass-3 Compass/Attidute Sensor Module

RECOVERY Relational Model:
Individual Sensors to Alarm Parameters

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



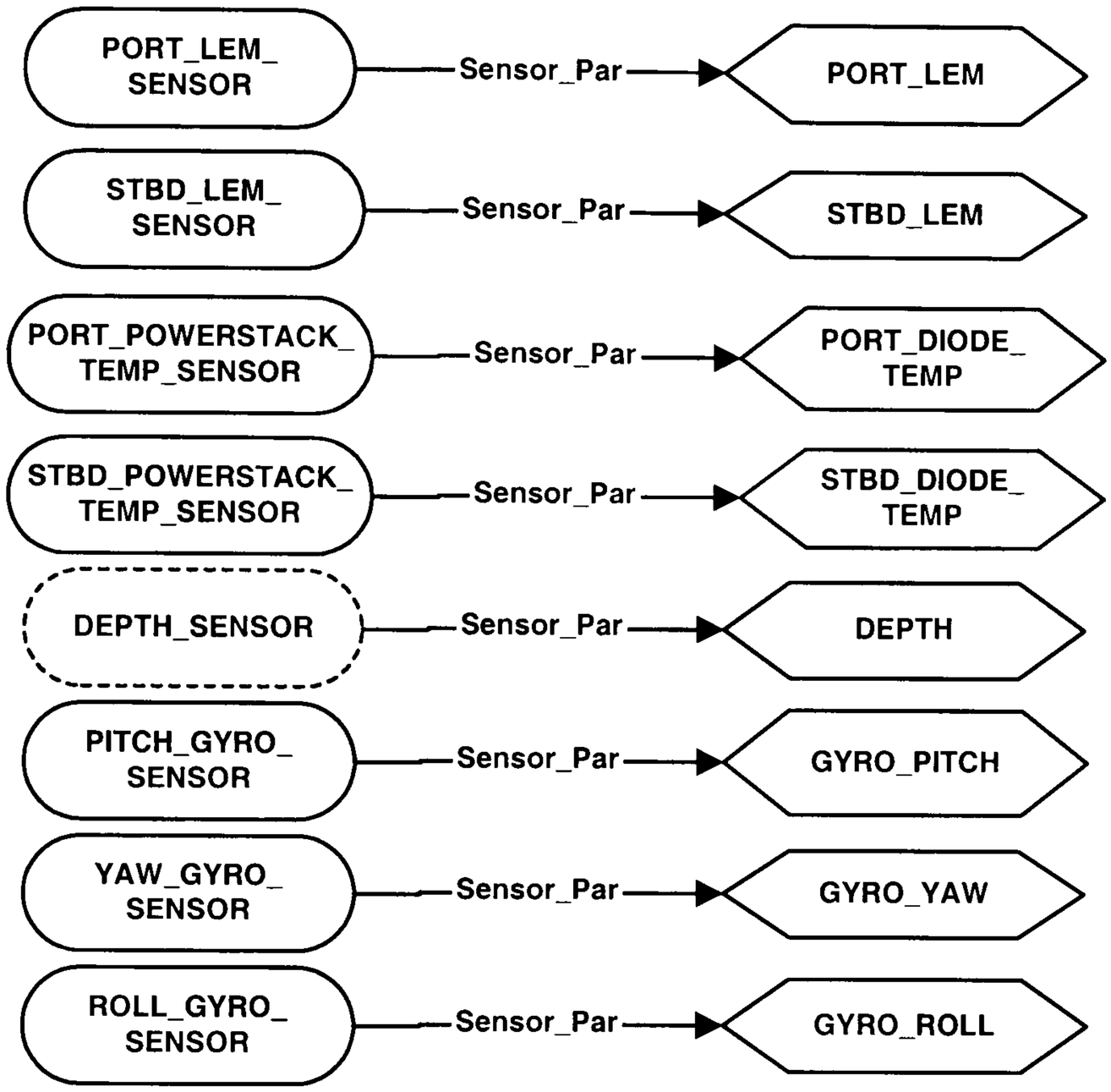

```
// alarms
LINK SENSOR_PAR PORT_DIODE_50_SENSOR ALARM_PDW;
LINK SENSOR_PAR PORT_DIODE_60_SENSOR ALARM_PDH;
LINK SENSOR_PAR PORT_NAVSUPPLY_HOT_SENSOR ALARM_PLH;
LINK SENSOR_PAR STBD_DIODE_50_SENSOR ALARM_SDW;
LINK SENSOR_PAR STBD_DIODE_60_SENSOR ALARM_SDH;
LINK SENSOR_PAR STBD_NAVSUPPLY_HOT_SENSOR ALARM_SLH;
LINK SENSOR_PAR WATER_SENSOR ALARM_WATER;
LINK SENSOR_PAR SPARE_DIGITAL_SENSOR ALARM_SPARE;
```

Figure B.7: Sensor Components to Alarm Parameters

RECOVERY Relational Model:
Individual Sensors to Sensed Parameters

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



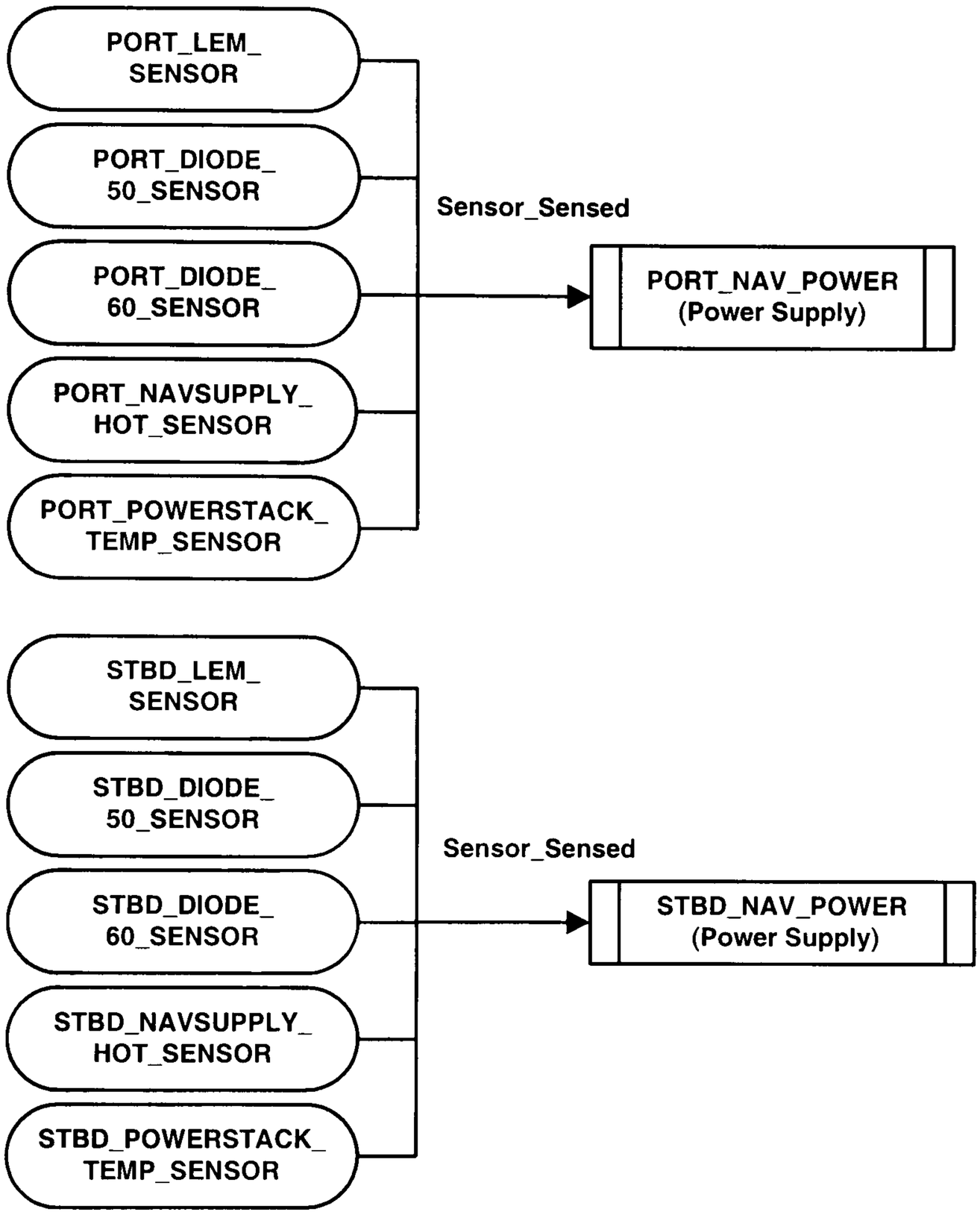

```
// sensor to par for sensed paramers
// tells you which component does the sensing
LINK SENSOR_PAR PORT_LEM_SENSOR PORT_LEM;
LINK SENSOR_PAR STBD_LEM_SENSOR STBD_LEM;

LINK SENSOR_PAR PORT_POWERSTACK_TEMP_SENSOR
    PORT_DIODE_TEMP;
LINK SENSOR_PAR STBD_POWERSTACK_TEMP_SENSOR
    STBD_DIODE_TEMP;

LINK SENSOR_PAR DEPTH_SENSOR DEPTH;
LINK SENSOR_PAR YAW_GYRO_SENSOR GYRO_YAW;
LINK SENSOR_PAR PITCH_GYRO_SENSOR GYRO_PITCH;
LINK SENSOR_PAR ROLL_GYRO_SENSOR GYRO_ROLL;
```

Figure B.8: Sensor Components to Sensed Parameters

RECOVERY Relational Model:
Individual Sensors to Sensed Components

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



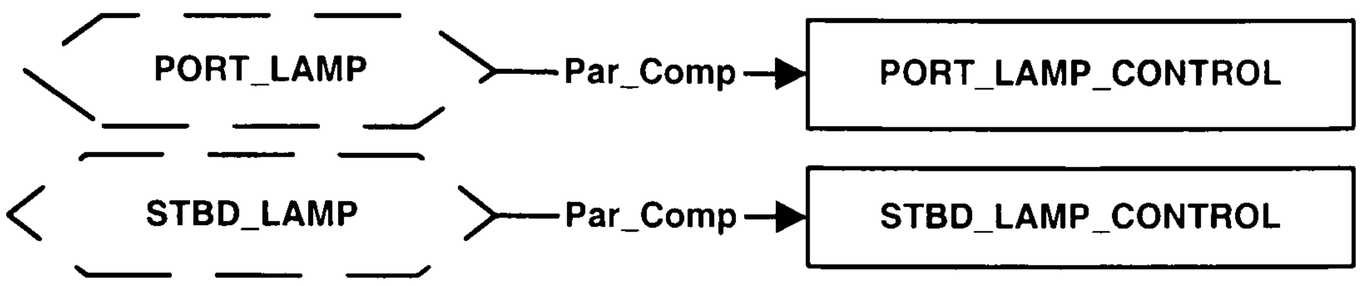

```
// sensor to sensed component
LINK SENSOR_SENSED PORT_LEM_SENSOR PORT_NAV_POWER;
LINK SENSOR_SENSED PORT_DIODE_50_SENSOR PORT_NAV_POWER;
LINK SENSOR_SENSED PORT_DIODE_60_SENSOR PORT_NAV_POWER;
LINK SENSOR_SENSED PORT_NAVSUPPLY_HOT_SENSOR PORT_NAV_POWER;
LINK SENSOR_SENSED PORT_POWERSTACK_TEMP_SENSOR
                   PORT_NAV_POWER;

LINK SENSOR_SENSED STBD_LEM_SENSOR STBD_NAV_POWER;
LINK SENSOR_SENSED STBD_DIODE_50_SENSOR STBD_NAV_POWER;
LINK SENSOR_SENSED STBD_DIODE_60_SENSOR STBD_NAV_POWER;
LINK SENSOR_SENSED STBD_NAVSUPPLY_HOT_SENSOR STBD_NAV_POWER;
```

Figure B.9: Sensor Components to Sensed Components

RECOVERY Relational Model:
Unsensed Parameters to Components

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



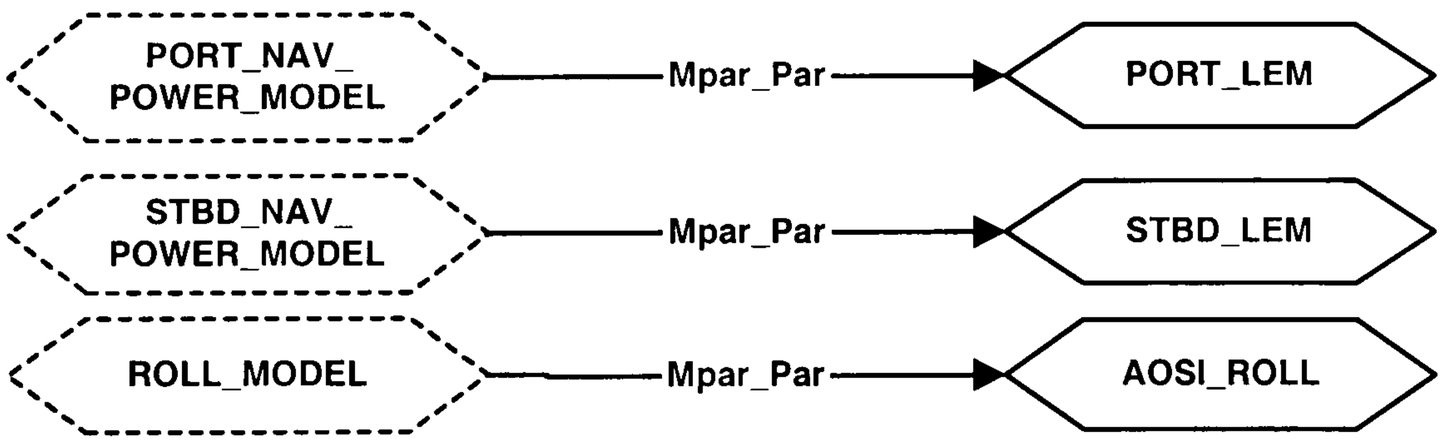

```
// parameter to component for unsensed parameters
LINK PAR_COMP PORT_LAMP PORT_LAMP_CONTROL;
LINK PAR_COMP STBD_LAMP STBD_LAMP_CONTROL;
```

Figure B.10: Unsensed Parameters to Components

RECOVERY Relational Model:
Modelled Parameters to Sensed Parameters

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



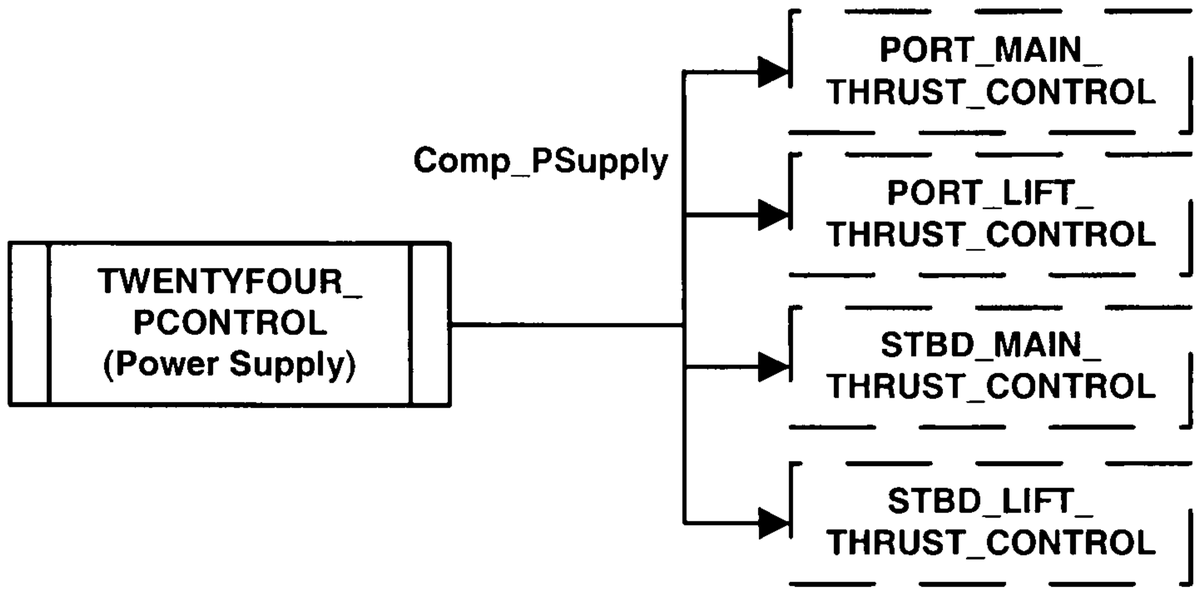

```
// modelled to sensed  parameter links
LINK MPAR_PAR PORT_NAV_POWER_MODEL PORT_LEM;
LINK MPAR_PAR STBD_NAV_POWER_MODEL STBD_LEM;
LINK MPAR_PAR ROLL_MODEL AOSI_ROLL;
```

Figure B.11: Modelled Parameters to Sensed Parameters

RECOVERY Relational Model:
Twelve and Twenty Four Volt Control Power Supplies to Components

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments



LINK COMP_PSUPPLY DEPTH_SENSOR TWELVE_PCONTROL;
LINK COMP_PSUPPLY PORT_POWERSTACK_TEMP_SENSOR TWELVE_PCONTROL;
LINK COMP_PSUPPLY STBD_POWERSTACK_TEMP_SENSOR TWELVE_PCONTROL;
LINK COMP_PSUPPLY AOS_EZ3_COMPASS_SENSOR TWELVE_PCONTROL

LINK COMP_PSUPPLY PORT_MAIN_THRUST_CONTROL TWENTYFOUR_PCONTROL;
LINK COMP_PSUPPLY PORT_LIFT_THRUST_CONTROL TWENTYFOUR_PCONTROL;
LINK COMP_PSUPPLY STBD_MAIN_THRUST_CONTROL TWENTYFOUR_PCONTROL;
LINK COMP_PSUPPLY STBD_LIFT_THRUST_CONTROL TWENTYFOUR_PCONTROL;

Figure B.12: Twelve and Twenty Four Volt Power Supplies to Supplied Components

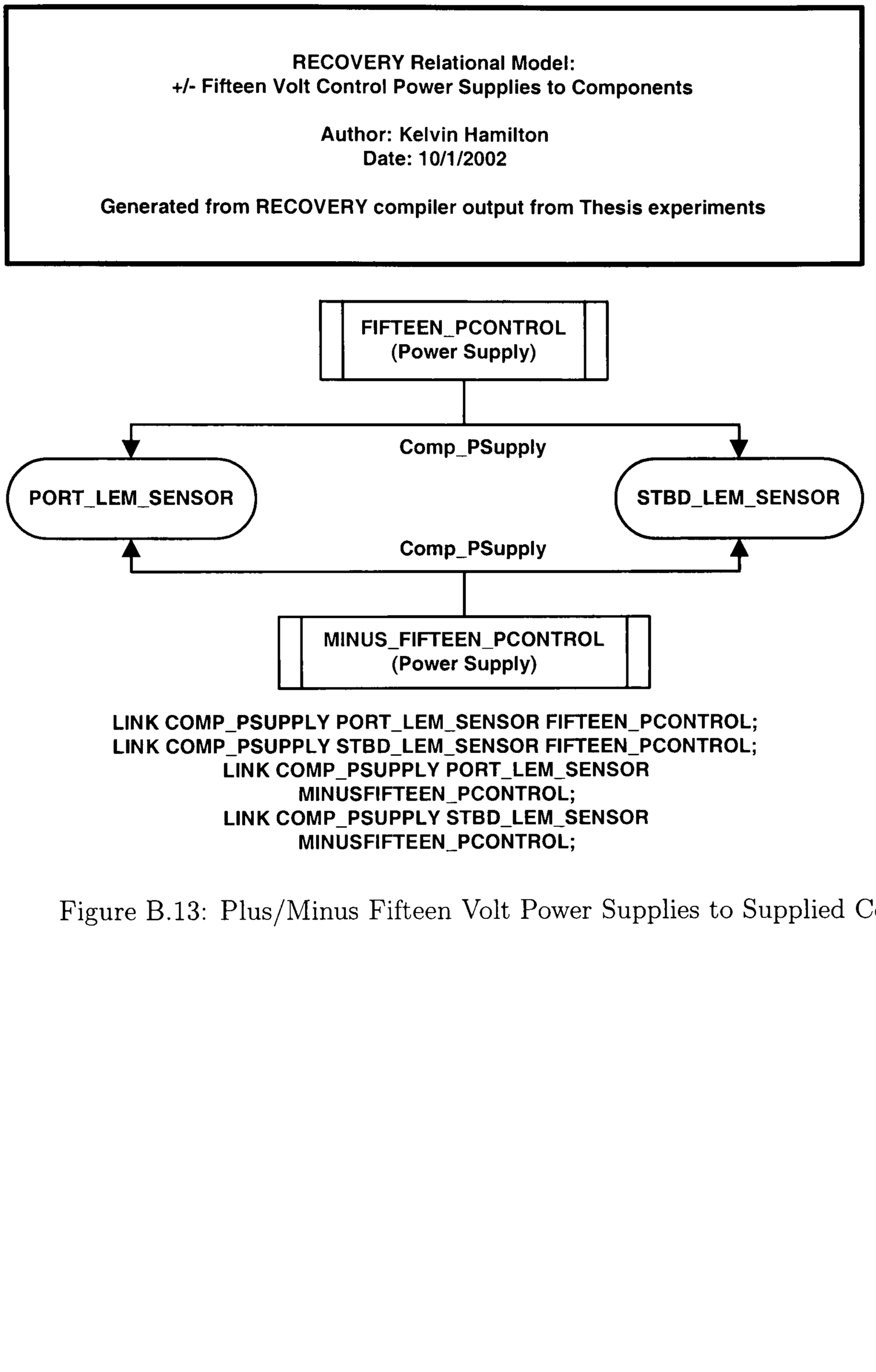
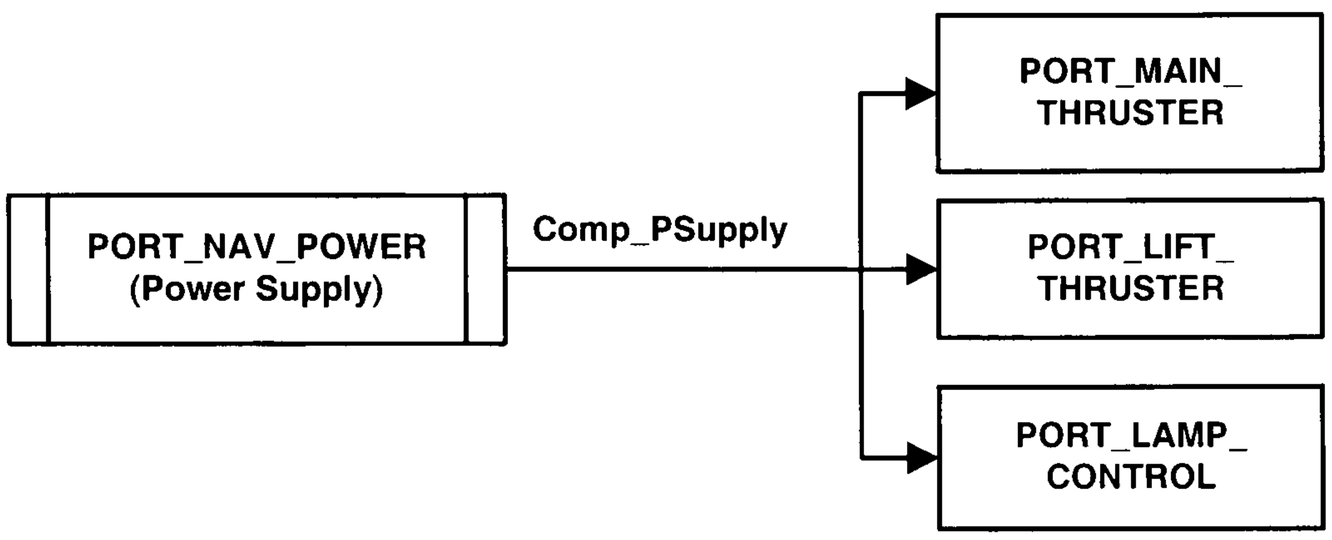


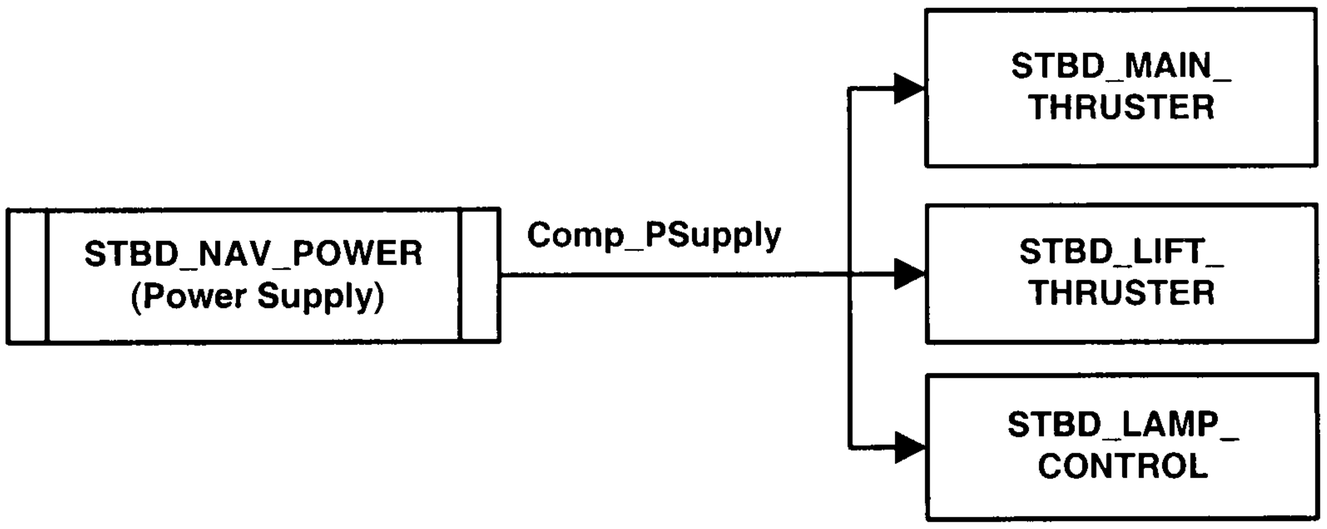
Figure B.13: Plus/Minus Fifteen Volt Power Supplies to Supplied Components

RECOVERY Relational Model:
Navigation Power Supplies to Components

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments





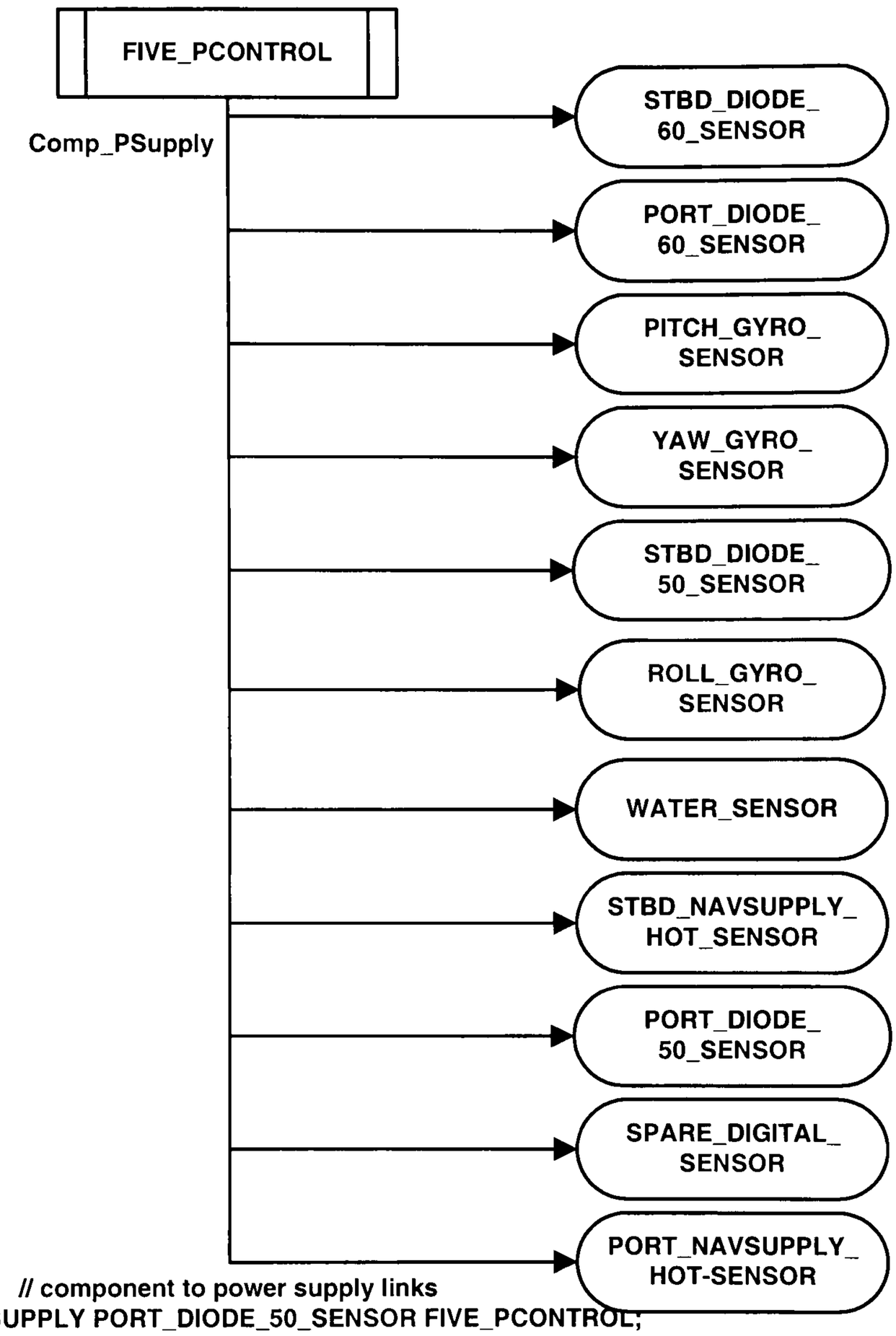

```
LINK COMP_PSUPPLY PORT_MAIN_THRUSTER PORT_NAV_POWER;
LINK COMP_PSUPPLY PORT_LIFT_THRUSTER PORT_NAV_POWER;
LINK COMP_PSUPPLY PORT_LAMP_CONTROL PORT_NAV_POWER;

LINK COMP_PSUPPLY STBD_MAIN_THRUSTER STBD_NAV_POWER;
LINK COMP_PSUPPLY STBD_LIFT_THRUSTER STBD_NAV_POWER;
LINK COMP_PSUPPLY STBD_LAMP_CONTROL STBD_NAV_POWER;
```

Figure B.14: Navigation Power Supplies to Supplied Components

RECOVERY Relational Model:
Five Volt Power Supply (Control) to Components

Author: Kelvin Hamilton
Date: 10/1/2002

Generated from RECOVERY compiler output from Thesis experiments




```
// component to power supply links
    LINK COMP_PSUPPLY PORT_DIODE_50_SENSOR FIVE_PCONTROL;
    LINK COMP_PSUPPLY STBD_DIODE_50_SENSOR FIVE_PCONTROL;
    LINK COMP_PSUPPLY PORT_DIODE_60_SENSOR FIVE_PCONTROL;
    LINK COMP_PSUPPLY STBD_DIODE_60_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY PORT_NAVSUPPLY_HOT_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY STBD_NAVSUPPLY_HOT_SENSOR FIVE_PCONTROL;
        LINK COMP_PSUPPLY WATER_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY SPARE_DIGITAL_SENSOR FIVE_PCONTROL;

LINK COMP_PSUPPLY YAW_GYRO_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY PITCH_GYRO_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY ROLL_GYRO_SENSOR FIVE_PCONTROL;
```

Figure B.15: Five Volt Power Supply to Supplied Components

B.2 Text Input to the RECOVERY Relational Model Compiler

This section shows the raw text input to the RECOVERY Relational Model compiler.

Full RECOVERY C++ code is included on the attached cd-rom.

B.2.1 Master List of Nodes

This is the master list of nodes. All types of nodes are included here, with the type of node followed by the individual node identifier. The SECTION_START and SECTION_END identifiers are used to segregate the types of nodes.

```
// RECOVERY file
// Construction model nodes (components and parameters)

// Adding a node here?
// Add generation info into
// construction model functions

NULL_NODE SECTION_START;

UNSENSED_PAR RUNTIME;

NULL_NODE FIRST_PARAMETER;

NULL_NODE UNKNOWN_PARAMETER;

UNSENSED_PAR PORT_MAIN;
UNSENSED_PAR STBD_MAIN;
UNSENSED_PAR PORT_LIFT;
UNSENSED_PAR STBD_LIFT;

UNSENSED_PAR PORT_LAMP;
UNSENSED_PAR STBD_LAMP;

SENSED_PAR DEPTH;
```

```
SENSED_PAR AOSI_HEADING;
SENSED_PAR AOSI_PITCH;
SENSED_PAR AOSI_ROLL;
SENSED_PAR AOSI_TEMP;
SENSED_PAR AOSI_MAGX;
SENSED_PAR AOSI_MAGY;
SENSED_PAR AOSI_MAGZ;

SENSED_PAR GPS_STATUS;
SENSED_PAR GPS_LAT_DEGREE;
SENSED_PAR GPS_LAT_MINUTE;
SENSED_PAR GPS_LAT_SECOND;
SENSED_PAR GPS_LAT_HEMISPHERE;
SENSED_PAR GPS_LONG_DEGREE;
SENSED_PAR GPS_LONG_MINUTE;
SENSED_PAR GPS_LONG_SECOND;
SENSED_PAR GPS_LONG_HEMISPHERE;
SENSED_PAR GPS_HEADING;
SENSED_PAR GPS_VELOCITY;
SENSED_PAR GPS_MAG_VARIATION;
SENSED_PAR GPS_MAG_VARIATION_DIRECTION;

SENSED_PAR GYRO_YAW;
SENSED_PAR GYRO_PITCH;
SENSED_PAR GYRO_ROLL;

SENSED_PAR PORT_DIODE_TEMP;
SENSED_PAR STBD_DIODE_TEMP;

SENSED_PAR PORT_LEM;
SENSED_PAR STBD_LEM;

SENSED_PAR ALTITUDE;


// vehicle alarms
NULL_NODE FIRST_ALARM;
```

```
ALARM_PAR ALARM_WATER;
ALARM_PAR ALARM_PDH;
ALARM_PAR ALARM_SDH;
ALARM_PAR ALARM_PDW;
ALARM_PAR ALARM_SDW;
ALARM_PAR ALARM_PLH;
ALARM_PAR ALARM_SLH;
ALARM_PAR ALARM_SPARE;
NULL_NODE LAST_ALARM;

// modelled parameters
NULL_NODE FIRST_MODELLED_PARAMETER;
MODELLED_PAR PORT_NAV_POWER_MODEL;
MODELLED_PAR STBD_NAV_POWER_MODEL;
MODELLED_PAR ROLL_MODEL;
MODELLED_PAR PITCH_MODEL;
MODELLED_PAR DEPTH_VELOCITY_MODEL;
MODELLED_PAR YAW_VELOCITY_MODEL;
MODELLED_PAR HEADING_MODEL;
MODELLED_PAR DEPTH_MODEL;
NULL_NODE LAST_MODELLED_PARAMETER;


NULL_NODE LAST_PARAMETER;


// ************** components *************
NULL_NODE FIRST_COMPONENT;

NULL_NODE UNKNOWN_COMPONENT;

// ****** first test position ***
// thrusters
// - comp followed by subcomp
COMP PORT_MAIN_THRUSTER;
SUB_COMP PORT_MAIN_THRUST_CONTROL;
SUB_COMP PORT_MAIN_MOTOR;
```

```
SUB_COMP PORT_MAIN_THRUSTER_BRUSH;
SUB_COMP PORT_MAIN_GBOX;
SUB_COMP PORT_MAIN_PROPELLER;

COMP STBD_MAIN_THRUSTER;
SUB_COMP STBD_MAIN_THRUST_CONTROL;
SUB_COMP STBD_MAIN_MOTOR;
SUB_COMP STBD_MAIN_THRUSTER_BRUSH;
SUB_COMP STBD_MAIN_GBOX;
SUB_COMP STBD_MAIN_PROPELLER;

COMP PORT_LIFT_THRUSTER;
SUB_COMP PORT_LIFT_THRUST_CONTROL;
SUB_COMP PORT_LIFT_MOTOR;
SUB_COMP PORT_LIFT_THRUSTER_BRUSH;
SUB_COMP PORT_LIFT_GBOX;
SUB_COMP PORT_LIFT_PROPELLER;


// this is the test node for rulebase
COMP STBD_LIFT_THRUSTER;
SUB_COMP STBD_LIFT_THRUST_CONTROL;
SUB_COMP STBD_LIFT_MOTOR;
SUB_COMP STBD_LIFT_THRUSTER_BRUSH;
SUB_COMP STBD_LIFT_GBOX;
SUB_COMP STBD_LIFT_PROPELLER;

// lamp components
COMP PORT_LAMP_CONTROL;
COMP STBD_LAMP_CONTROL;


// power supplies
PSUPPLY_COMP PORT_BATTERY;
PSUPPLY_COMP STBD_BATTERY;
PSUPPLY_COMP PORT_NAV_POWER;
PSUPPLY_COMP STBD_NAV_POWER;
```

```
// ******** middle position of components


PSUPPLY_COMP TWELVE_PCONTROL;
PSUPPLY_COMP TWENTYFOUR_PCONTROL;
PSUPPLY_COMP COMMS_POWER;
PSUPPLY_COMP FIVE_PCONTROL;
PSUPPLY_COMP FIFTEEN_PCONTROL;
PSUPPLY_COMP MINUSFIFTEEN_PCONTROL;

// comms
COMP SURFACE_COMMS_UNIT;

// sensors
// all sensors here; including simple thermostats and trips
// analogue sensor section
SYSTEM_SENSOR_COMP PORT_LEM_SENSOR;
SYSTEM_SENSOR_COMP STBD_LEM_SENSOR;
SYSTEM_SENSOR_COMP PORT_POWERSTACK_TEMP_SENSOR;
SYSTEM_SENSOR_COMP STBD_POWERSTACK_TEMP_SENSOR;
SYSTEM_SENSOR_COMP YAW_GYRO_SENSOR;
SYSTEM_SENSOR_COMP PITCH_GYRO_SENSOR;
SYSTEM_SENSOR_COMP ROLL_GYRO_SENSOR;
SYSTEM_SENSOR_COMP GPS_SENSOR;

ENVIRO_SENSOR_COMP DEPTH_SENSOR;
ENVIRO_SENSOR_COMP AOSI_EZ3_COMPASS_SENSOR;
ENVIRO_SENSOR_COMP CTD_SENSOR;

// digital sensors
SYSTEM_SENSOR_COMP PORT_DIODE_50_SENSOR;
SYSTEM_SENSOR_COMP STBD_DIODE_50_SENSOR;
SYSTEM_SENSOR_COMP PORT_DIODE_60_SENSOR;
SYSTEM_SENSOR_COMP STBD_DIODE_60_SENSOR;
SYSTEM_SENSOR_COMP PORT_NAVSUPPLY_HOT_SENSOR;
SYSTEM_SENSOR_COMP STBD_NAVSUPPLY_HOT_SENSOR;
SYSTEM_SENSOR_COMP WATER_SENSOR;
SYSTEM_SENSOR_COMP SPARE_DIGITAL_SENSOR;
```

```
// circuit boards
COMP LAMP_CONVERTOR_PCB;
COMP DIGITAL_INPUT_PCB;
COMP DIGITAL_POT_PCB;
COMP ANALOGUE_IO_PCB;
COMP CONTROL_PCB;
```

```
// ******* last test position ***
// this is the test node for rulebase
COMP STBD_LIFT_THRUSTER;

NULL_NODE LAST_COMPONENT;

NULL_NODE SECTION_END;
```

B.2.2 Port Main-Thruster Module

This is an individual component file showing all the links for the Port Main-Thruster Module. The other module files follow the same format.

```
// RECOVERY knowledge file
// Rauver Vehicle, Ocean Systems Laboratory
// Port Main Thruster Semantic Model
// K.Hamilton 25/10/2000
//
// Defines the links between nodes
// Link type, endpoint1, endpoint2, proposed end1, proposed end2

// sub components to components
LINK SUBCOMP_COMP PORT_MAIN_THRUST_CONTROL PORT_MAIN_THRUSTER;
LINK SUBCOMP_COMP PORT_MAIN_MOTOR PORT_MAIN_THRUSTER;
LINK SUBCOMP_COMP PORT_MAIN_THRUSTER_BRUSH PORT_MAIN_THRUSTER;
LINK SUBCOMP_COMP PORT_MAIN_GBOX PORT_MAIN_THRUSTER;
LINK SUBCOMP_COMP PORT_MAIN_PROPELLER PORT_MAIN_THRUSTER;
```

```
// parameters to components
LINK PAR_COMP PORT_MAIN PORT_MAIN_THRUSTER;

// components to power supplies
LINK COMP_PSUPPLY PORT_MAIN_THRUSTER PORT_NAV_POWER;
LINK COMP_PSUPPLY PORT_MAIN_THRUST_CONTROL TWENTYFOUR_PCONTROL;

// known faults and dynamic models
// FaultCondition1 FaultCondition2 FaultCondition3 Candidate
// if thruster active and power lower than model then propeller off
// if thruster active and power higher than model then gbox jammed or prop stuck
```

B.2.3 Port Lift-Thruster Module

```
// RECOVERY knowledge file
// Rauver Vehicle, Ocean Systems Laboratory
// Port LIFT Thruster Semantic Model
// K.Hamilton 25/10/2000
//
// Defines the links between nodes
// Link type, endpoint1, endpoint2, proposed end1, proposed end2

// sub components to components
LINK SUBCOMP_COMP PORT_LIFT_THRUST_CONTROL PORT_LIFT_THRUSTER;
LINK SUBCOMP_COMP PORT_LIFT_MOTOR PORT_LIFT_THRUSTER;
LINK SUBCOMP_COMP PORT_LIFT_THRUSTER_BRUSH PORT_LIFT_THRUSTER;
LINK SUBCOMP_COMP PORT_LIFT_GBOX PORT_LIFT_THRUSTER;
LINK SUBCOMP_COMP PORT_LIFT_PROPELLER PORT_LIFT_THRUSTER;

// parameters to components
LINK PAR_COMP PORT_LIFT PORT_LIFT_THRUSTER;

// components to power supplies
LINK COMP_PSUPPLY PORT_LIFT_THRUSTER PORT_NAV_POWER;
LINK COMP_PSUPPLY PORT_LIFT_THRUST_CONTROL TWENTYFOUR_PCONTROL;
```

```
// known faults and dynamic models
```

B.2.4 Starboard Main-Thruster Module

```
// RECOVERY knowledge file
// Rauver Vehicle, Ocean Systems Laboratory
// STBD Main Thruster Semantic Model
// K.Hamilton 25/10/2000
//
// Defines the links between nodes
// Link type, endpoint1, endpoint2, proposed end1, proposed end2

// sub components to components
LINK SUBCOMP_COMP STBD_MAIN_THRUST_CONTROL STBD_MAIN_THRUSTER;
LINK SUBCOMP_COMP STBD_MAIN_MOTOR STBD_MAIN_THRUSTER;
LINK SUBCOMP_COMP STBD_MAIN_THRUSTER_BRUSH STBD_MAIN_THRUSTER;
LINK SUBCOMP_COMP STBD_MAIN_GBOX STBD_MAIN_THRUSTER;
LINK SUBCOMP_COMP STBD_MAIN_PROPELLER STBD_MAIN_THRUSTER;

// parameters to components
LINK PAR_COMP STBD_MAIN STBD_MAIN_THRUSTER;

// components to power supplies
LINK COMP_PSUPPLY STBD_MAIN_THRUSTER STBD_NAV_POWER;
LINK COMP_PSUPPLY STBD_MAIN_THRUST_CONTROL TWENTYFOUR_PCONTROL;

// known faults and dynamic models
```

B.2.5 Starboard Lift-Thruster Module

```
// RECOVERY knowledge file
// Rauver Vehicle, Ocean Systems Laboratory
// STBD Main Thruster Semantic Model
```

```
// K.Hamilton 25/10/2000
//
// Defines the links between nodes
// Link type, endpoint1, endpoint2, proposed end1, proposed end2

// sub components to components
LINK SUBCOMP_COMP STBD_LIFT_THRUST_CONTROL STBD_LIFT_THRUSTER;
LINK SUBCOMP_COMP STBD_LIFT_MOTOR STBD_LIFT_THRUSTER;
LINK SUBCOMP_COMP STBD_LIFT_THRUSTER_BRUSH STBD_LIFT_THRUSTER;
LINK SUBCOMP_COMP STBD_LIFT_GBOX STBD_LIFT_THRUSTER;
LINK SUBCOMP_COMP STBD_LIFT_PROPELLER STBD_LIFT_THRUSTER;

// parameters to components
LINK PAR_COMP STBD_LIFT STBD_LIFT_THRUSTER;

// components to power supplies
LINK COMP_PSUPPLY STBD_LIFT_THRUSTER STBD_NAV_POWER;
LINK COMP_PSUPPLY STBD_LIFT_THRUST_CONTROL TWENTYFOUR_PCONTROL;

// known faults and dynamic models
```

B.2.6 AOSI EZ Compass 3 Sensor Module

```
// RECOVERY knowledge file
// Rauver Vehicle, Ocean Systems Laboratory
// K.Hamilton 26/10/2000
//
// Defines the links between nodes
// Link type, endpoint1, endpoint2, proposed end1, proposed end2

// *************** Links: ***************
//      SENSOR_PAR,
LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_HEADING;
LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_PITCH;
LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_ROLL;
LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_TEMP;
LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_MAGX;
```

```
LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_MAGY;
LINK SENSOR_PAR AOSI_EZ3_COMPASS_SENSOR AOSI_MAGZ;

//      COMP_PSUPPLY,
LINK COMP_PSUPPLY AOSI_EZ3_COMPASS_SENSOR TWELVE_PCONTROL;
```

B.2.7 System-Level Links

Here are links that are not for any particular module.

```
// RECOVERY information file
// Model Links - RAUVER vehicle
// K.Hamilton 1/11/00

// modelled to sensed  parameter links
LINK MPAR_PAR PORT_NAV_POWER_MODEL PORT_LEM;
LINK MPAR_PAR STBD_NAV_POWER_MODEL STBD_LEM;
LINK MPAR_PAR ROLL_MODEL AOSI_ROLL;

// for correlator evaluation (null movement thermal) only
//LINK MPAR_PAR HEADING_MODEL AOSI_HEADING;

// component to power supply links
LINK COMP_PSUPPLY PORT_LAMP_CONTROL PORT_NAV_POWER;
LINK COMP_PSUPPLY STBD_LAMP_CONTROL STBD_NAV_POWER;

LINK COMP_PSUPPLY PORT_LEM_SENSOR FIFTEEN_PCONTROL;
LINK COMP_PSUPPLY STBD_LEM_SENSOR FIFTEEN_PCONTROL;
LINK COMP_PSUPPLY PORT_LEM_SENSOR MINUSFIFTEEN_PCONTROL;
LINK COMP_PSUPPLY STBD_LEM_SENSOR MINUSFIFTEEN_PCONTROL;

LINK COMP_PSUPPLY PORT_DIODE_50_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY STBD_DIODE_50_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY PORT_DIODE_60_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY STBD_DIODE_60_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY PORT_NAVSUPPLY_HOT_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY STBD_NAVSUPPLY_HOT_SENSOR FIVE_PCONTROL;
```

```
LINK COMP_PSUPPLY WATER_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY SPARE_DIGITAL_SENSOR FIVE_PCONTROL;

LINK COMP_PSUPPLY DEPTH_SENSOR TWELVE_PCONTROL;

LINK COMP_PSUPPLY PORT_POWERSTACK_TEMP_SENSOR TWELVE_PCONTROL;
LINK COMP_PSUPPLY STBD_POWERSTACK_TEMP_SENSOR TWELVE_PCONTROL;

LINK COMP_PSUPPLY YAW_GYRO_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY PITCH_GYRO_SENSOR FIVE_PCONTROL;
LINK COMP_PSUPPLY ROLL_GYRO_SENSOR FIVE_PCONTROL;

// parameter to component for unsensed parameters
LINK PAR_COMP PORT_LAMP PORT_LAMP_CONTROL;
LINK PAR_COMP STBD_LAMP STBD_LAMP_CONTROL;

// sensor to par for sensed paramers
// tells you which component does the sensing
LINK SENSOR_PAR PORT_LEM_SENSOR PORT_LEM;
LINK SENSOR_PAR STBD_LEM_SENSOR STBD_LEM;

LINK SENSOR_PAR PORT_POWERSTACK_TEMP_SENSOR PORT_DIODE_TEMP;
LINK SENSOR_PAR STBD_POWERSTACK_TEMP_SENSOR STBD_DIODE_TEMP;

LINK SENSOR_PAR DEPTH_SENSOR DEPTH;
LINK SENSOR_PAR YAW_GYRO_SENSOR GYRO_YAW;
LINK SENSOR_PAR PITCH_GYRO_SENSOR GYRO_PITCH;
LINK SENSOR_PAR ROLL_GYRO_SENSOR GYRO_ROLL;


// alarms - may need to add another link type
LINK SENSOR_PAR PORT_DIODE_50_SENSOR ALARM_PDW;
LINK SENSOR_PAR PORT_DIODE_60_SENSOR ALARM_PDH;
LINK SENSOR_PAR PORT_NAVSUPPLY_HOT_SENSOR ALARM_PLH;
LINK SENSOR_PAR STBD_DIODE_50_SENSOR ALARM_SDW;
LINK SENSOR_PAR STBD_DIODE_60_SENSOR ALARM_SDH;
LINK SENSOR_PAR STBD_NAVSUPPLY_HOT_SENSOR ALARM_SLH;
LINK SENSOR_PAR WATER_SENSOR ALARM_WATER;
```

```
LINK SENSOR_PAR SPARE_DIGITAL_SENSOR ALARM_SPARE;


// sensor to sensed component
LINK SENSOR_SENSED PORT_LEM_SENSOR PORT_NAV_POWER;
LINK SENSOR_SENSED PORT_DIODE_50_SENSOR PORT_NAV_POWER;
LINK SENSOR_SENSED PORT_DIODE_60_SENSOR PORT_NAV_POWER;
LINK SENSOR_SENSED PORT_NAVSUPPLY_HOT_SENSOR PORT_NAV_POWER;
LINK SENSOR_SENSED PORT_POWERSTACK_TEMP_SENSOR PORT_NAV_POWER;

LINK SENSOR_SENSED STBD_LEM_SENSOR STBD_NAV_POWER;
LINK SENSOR_SENSED STBD_DIODE_50_SENSOR STBD_NAV_POWER;
LINK SENSOR_SENSED STBD_DIODE_60_SENSOR STBD_NAV_POWER;
LINK SENSOR_SENSED STBD_NAVSUPPLY_HOT_SENSOR STBD_NAV_POWER;
```

B.2.8 Known Faults

Here are the known faults, represented as both static and dynamic rules, that were used for evaluation purposes.

```
// ********************* INFORMATION ******************
// STATIC RULES
// Usage:
// RULE component parameter min max equals;

// thrusters
// - comp followed by subcomp
RULE PORT_MAIN_THRUSTER DEPTH -10.0 50.0 -10.0;
RULE PORT_MAIN_THRUST_CONTROL DEPTH -10.0 50.0 -10.0;
RULE PORT_MAIN_MOTOR DEPTH -10.0 50.0 -10.0;
RULE PORT_MAIN_THRUSTER_BRUSH DEPTH -10.0 50.0 -10.0;
RULE PORT_MAIN_GBOX DEPTH -10.0 50.0 -10.0;
RULE PORT_MAIN_PROPELLER DEPTH -10.0 50.0 -10.0;

RULE STBD_MAIN_THRUSTER DEPTH -10.0 50.0 -10.0;
RULE STBD_MAIN_THRUST_CONTROL DEPTH -10.0 50.0 -10.0;
```

```
RULE STBD_MAIN_MOTOR DEPTH -10.0 50.0 -10.0;
RULE STBD_MAIN_THRUSTER_BRUSH DEPTH -10.0 50.0 -10.0;
RULE STBD_MAIN_GBOX DEPTH -10.0 50.0 -10.0;
RULE STBD_MAIN_PROPELLER DEPTH -10.0 50.0 -10.0;

RULE PORT_LIFT_THRUSTER DEPTH -10.0 50.0 -10.0;
RULE PORT_LIFT_THRUST_CONTROL DEPTH -10.0 50.0 -10.0;
RULE PORT_LIFT_MOTOR DEPTH -10.0 50.0 -10.0;
RULE PORT_LIFT_THRUSTER_BRUSH DEPTH -10.0 50.0 -10.0;
RULE PORT_LIFT_GBOX DEPTH -10.0 50.0 -10.0;
RULE PORT_LIFT_PROPELLER DEPTH -10.0 50.0 -10.0;

RULE STBD_LIFT_THRUSTER DEPTH -10.0 50.0 -10.0;
RULE STBD_LIFT_THRUST_CONTROL DEPTH -10.0 50.0 -10.0;
RULE STBD_LIFT_MOTOR DEPTH -10.0 50.0 -10.0;
RULE STBD_LIFT_THRUSTER_BRUSH DEPTH -10.0 50.0 -10.0;
RULE STBD_LIFT_GBOX DEPTH -10.0 50.0 -10.0;
RULE STBD_LIFT_PROPELLER DEPTH -10.0 50.0 -10.0;

// lamp components
RULE PORT_LAMP_CONTROL DEPTH -10.0 50.0 -10.0;
RULE STBD_LAMP_CONTROL DEPTH -10.0 50.0 -10.0;


// power supplies
RULE PORT_BATTERY DEPTH -10.0 50.0 -10.0;
RULE STBD_BATTERY DEPTH -10.0 50.0 -10.0;
RULE PORT_NAV_POWER DEPTH -10.0 50.0 -10.0;
RULE STBD_NAV_POWER DEPTH -10.0 50.0 -10.0;
RULE TWELVE_PCONTROL DEPTH -10.0 50.0 -10.0;
RULE TWENTYFOUR_PCONTROL DEPTH -10.0 50.0 -10.0;
RULE COMMS_POWER DEPTH -10.0 50.0 -10.0;
RULE FIVE_PCONTROL DEPTH -10.0 50.0 -10.0;
RULE FIFTEEN_PCONTROL DEPTH -10.0 50.0 -10.0;
RULE MINUSFIFTEEN_PCONTROL DEPTH -10.0 50.0 -10.0;

// comms
RULE SURFACE_COMMS_UNIT DEPTH -10.0 50.0 -10.0;
```

```
// sensors
// all sensors here DEPTH -10.0 50.0 -10.0; including simple thermostats and trips
// analogue sensor section
RULE PORT_LEM_SENSOR DEPTH -10.0 50.0 -10.0;
RULE STBD_LEM_SENSOR DEPTH -10.0 50.0 -10.0;
RULE PORT_POWERSTACK_TEMP_SENSOR DEPTH -10.0 50.0 -10.0;
RULE STBD_POWERSTACK_TEMP_SENSOR DEPTH -10.0 50.0 -10.0;
RULE YAW_GYRO_SENSOR DEPTH -10.0 50.0 -10.0;
RULE PITCH_GYRO_SENSOR DEPTH -10.0 50.0 -10.0;
RULE ROLL_GYRO_SENSOR DEPTH -10.0 50.0 -10.0;
RULE GPS_SENSOR DEPTH -10.0 50.0 -10.0;

RULE DEPTH_SENSOR DEPTH -10.0 50.0 -10.0;
RULE AOSI_EZ3_COMPASS_SENSOR DEPTH -10.0 50.0 -10.0;
RULE CTD_SENSOR DEPTH -10.0 50.0 -10.0;

// digital sensors
RULE PORT_DIODE_50_SENSOR DEPTH -10.0 50.0 -10.0;
RULE STBD_DIODE_50_SENSOR DEPTH -10.0 50.0 -10.0;
RULE PORT_DIODE_60_SENSOR DEPTH -10.0 50.0 -10.0;
RULE STBD_DIODE_60_SENSOR DEPTH -10.0 50.0 -10.0;
RULE PORT_NAVSUPPLY_HOT_SENSOR DEPTH -10.0 50.0 -10.0;
RULE STBD_NAVSUPPLY_HOT_SENSOR DEPTH -10.0 50.0 -10.0;
RULE WATER_SENSOR DEPTH -10.0 50.0 -10.0;
RULE SPARE_DIGITAL_SENSOR DEPTH -10.0 50.0 -10.0;

// circuit boards
RULE LAMP_CONVERTOR_PCB DEPTH -10.0 50.0 -10.0;
RULE DIGITAL_INPUT_PCB DEPTH -10.0 50.0 -10.0;
RULE DIGITAL_POT_PCB DEPTH -10.0 50.0 -10.0;
RULE ANALOGUE_IO_PCB DEPTH -10.0 50.0 -10.0;
RULE CONTROL_PCB DEPTH -10.0 50.0 -10.0;

// ****************** DYNAMIC RULES
// Usage:
// DYNAMIC_RULE component parameter min max equals;
```

```
// thrusters
// - comp followed by subcomp
DYNAMIC_RULE PORT_MAIN_THRUSTER DEPTH -10.0 50.0 -10.0;

// **** this component has a unique value
DYNAMIC_RULE PORT_MAIN_THRUST_CONTROL DEPTH -10.0 5.0 -10.0;
DYNAMIC_RULE PORT_MAIN_MOTOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_MAIN_THRUSTER_BRUSH DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_MAIN_GBOX DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_MAIN_PROPELLER DEPTH -10.0 50.0 -10.0;

DYNAMIC_RULE STBD_MAIN_THRUSTER DEPTH -10.0 50.0 -10.0;

// **** this component has a unique value
DYNAMIC_RULE STBD_MAIN_THRUST_CONTROL DEPTH -10.0 5.0 -10.0;
DYNAMIC_RULE STBD_MAIN_MOTOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_MAIN_THRUSTER_BRUSH DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_MAIN_GBOX DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_MAIN_PROPELLER DEPTH -10.0 50.0 -10.0;

DYNAMIC_RULE PORT_LIFT_THRUSTER DEPTH -10.0 50.0 -10.0;

// **** this component has a unique value
DYNAMIC_RULE PORT_LIFT_THRUST_CONTROL DEPTH -10.0 5.0 -10.0;
DYNAMIC_RULE PORT_LIFT_MOTOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_LIFT_THRUSTER_BRUSH DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_LIFT_GBOX DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_LIFT_PROPELLER DEPTH -10.0 50.0 -10.0;

DYNAMIC_RULE STBD_LIFT_THRUSTER DEPTH -10.0 50.0 -10.0;

// **** this component has a unique value
DYNAMIC_RULE STBD_LIFT_THRUST_CONTROL DEPTH -10.0 5.0 -10.0;
DYNAMIC_RULE STBD_LIFT_MOTOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_LIFT_THRUSTER_BRUSH DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_LIFT_GBOX DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_LIFT_PROPELLER DEPTH -10.0 50.0 -10.0;
```

```
// lamp components
DYNAMIC_RULE PORT_LAMP_CONTROL DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_LAMP_CONTROL DEPTH -10.0 50.0 -10.0;


// power supplies
DYNAMIC_RULE PORT_BATTERY DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_BATTERY DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_NAV_POWER DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_NAV_POWER DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE TWELVE_PCONTROL DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE TWENTYFOUR_PCONTROL DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE COMMS_POWER DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE FIVE_PCONTROL DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE FIFTEEN_PCONTROL DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE MINUSFIFTEEN_PCONTROL DEPTH -10.0 50.0 -10.0;

// comms
DYNAMIC_RULE SURFACE_COMMS_UNIT DEPTH -10.0 50.0 -10.0;

// sensors
// all sensors here DEPTH -10.0 50.0 -10.0; including simple thermostats and trips
// analogue sensor section
DYNAMIC_RULE PORT_LEM_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_LEM_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_POWERSTACK_TEMP_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_POWERSTACK_TEMP_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE YAW_GYRO_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PITCH_GYRO_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE ROLL_GYRO_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE GPS_SENSOR DEPTH -10.0 50.0 -10.0;

DYNAMIC_RULE DEPTH_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE AOSI_EZ3_COMPASS_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE CTD_SENSOR DEPTH -10.0 50.0 -10.0;

// digital sensors
DYNAMIC_RULE PORT_DIODE_50_SENSOR DEPTH -10.0 50.0 -10.0;
```

```
DYNAMIC_RULE STBD_DIODE_50_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_DIODE_60_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_DIODE_60_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE PORT_NAVSUPPLY_HOT_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE STBD_NAVSUPPLY_HOT_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE WATER_SENSOR DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE SPARE_DIGITAL_SENSOR DEPTH -10.0 50.0 -10.0;

// circuit boards
DYNAMIC_RULE LAMP_CONVERTOR_PCB DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE DIGITAL_INPUT_PCB DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE DIGITAL_POT_PCB DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE ANALOGUE_IO_PCB DEPTH -10.0 50.0 -10.0;
DYNAMIC_RULE CONTROL_PCB DEPTH -10.0 50.0 -10.0;
```

Appendix C

RAUVER Information

C.1 RAUVER Specification

In ROV mode RAUVER uses a multicore tether to receive power and communicate with the surface via a standard industrial RS485 serial communications link, this is then converted to RS232 for use with a standard PC. All power conversion is performed on board RAUVER using low noise toroidal transformers to bring the voltages down to 12/24Vdc, this is then passed to the motor controllers and lighting. A separate power supply provides control system voltages, this split system maximises control system noise immunity from the PWM motor controllers and any other switching transients.

The control system comprises a Motorola MC68HC11-F1 QED microcontroller board, clocked at 16MHz. This receives commands from the surface via the RS485 link, sets thruster powers accordingly and sends RAUVER system status information back to the surface.

For AUV operation the power conversion module is simply removed and replaced with 12V batteries, these are connected to give 12/24Vdc as in ROV mode. In this way down time for changing between ROV and AUV modes is minimised, allowing more experiments to be carried out during expensive field trials.

In both modes an electrically isolated and screened payload module is available, comprising a cylindrical volume of approximately 200mm x 200mm situated at the front of the starboard hull. All control and power voltages are available for use by the payload.

RAUVER is very versatile and can easily be adapted for mission specific requirements. The specification below is for the standard RAUVER configuration, if you wish to discuss specific mission requirements please use the contact link at the bottom of the page.

Information on RAUVER is freely available, and includes a fully detailed 3d CAD model containing all internal and external components.

Specifications:

Maximum Diving Depth 70m (Sport Diver Limit)

Overall Length Approximately 2m

Overall Width Approximately 0.8m

Weight in Air 120kg Maximum

Weight in Water Approximately neutral, trimmed slightly buoyant.

Trim Positive and negative, position adjustable

Tether Length (ROV mode only) 90m

Forward Speed Approximately 3 knots

Lifting capacity Up to 10kg in standard configuration

Main-Thruster Units:

2 x Bosch dc motors,

12Vdc

200W

Coupled to 300mm diameter nozzled propeller via a 5:1 gearbox.

Each thruster provides in excess of 10kg thrust.

Ramp rates set to 2 seconds for zero to full speed

Lift-Thruster Units:

(Up to 4) Hydrovision 'Hyball' thrusters

24Vdc

380W

Coupled to 130mm diameter propeller in Kort nozzle via a 10.5:1 gearbox

Each thruster provides in excess of 13kg thrust.

Ramp rates set to 2 seconds for zero to full speed

Thruster Control:

Penny & Giles 'Solo 60' 24Vdc 60A programmable digital motor control units, soft-

ware limited to 22A.

Speed control signal derived from an Analog Devices AD8403AN10 digital potentiometer controlled by RAUVER's system MC68HC11F1.

On Board Power:

AUV mode has;

5Vdc regulated, 750mA

5Vdc regulated, 10mA, for analogue signals

12Vdc regulated, 2A

24Vdc regulated, 2A

+15Vdc regulated

-15Vdc regulated

12Vdc main battery

24Vdc main battery

ROV mode has;

5Vdc regulated, 750mA

5Vdc regulated, 10mA, for analogue signals

12Vdc regulated, 2A

24Vdc regulated, 2A

+15Vdc regulated

-15Vdc regulated

24Vdc unregulated

230Vac 'mains', 3A min. available

ROV Mode Power Conversion:

4 x 500VA 24V toroidal transformers

International Rectifier 35A Bridge Rectifiers

68,000uF smoothing capacitors

Lighting:

2 x 70W Tungsten Halogen Lamps situated at the front of each hull. Illumination

adjustable independently using Digilog dimmers

Fault Protection:

Electrical - Zone 2 Residual Current Circuit Breaker plus various fuses

Hull breach - Water detector at front of each hull and in junction box.

Thermal Protection - 50'C / 60'C / 90'C thermostats at relevant locations in each hull, plus analogue power/control-stack temperature signals

Impact Protection - Full stainless-steel 316L (non-magnetic) 25mm diameter cage

Cameras:

1 x 3 Lux Colour TV Camera, soon to be attached to a micro pan and tilt unit designed here in the Ocean Systems Laboratory

Sensors:

Sonar - 'SeaKing' Mechanically Scanned Forward Looking SONAR

Depth - Lucas Schaevitz 70m Absolute type, +- 0.05% Accuracy

Heading - AOSI EZ-Compass 3

Magneto-inductive/capacitive

Attitude compensated

0.5 degree resolution

Gyros - 3 x Murata 'Camcorder' type gyroscopes. Also investigating advanced fibre-optic gyroscopes.

x-y positioning - Under review as vehicle too slow for an Inertial Navigation System. Will possibly use a Doppler log with flow sensors used in the meantime.

Spare Tether Cores:

2 x twisted pair

1 co-ax

Appendix D

Sample RAUVER Evaluation Mission Log File

Shown below is a sample from the RAUVER mission log file generated and stored by RAUVER during the course of the Faulty Port Lift-Thruster Experiment (Section 7.2.7). Each line represents one sample of all status data. It loosely follows the NMEA format, with a start and finish character of $ and * respectively. Each data field is preceded by a unique identifier, for instance A196 shows 196 seconds since vehicle power-up, B415 shows the milliseconds of the current second.

Full copies of all mission files are included on the attached cd-rom.

D.1 Mission Log File from the Faulty Port Lift-Thruster Experiment

"Ocean Systems Laboratory, Heriot-Watt University, UK"

"(www.cee.hw.ac.uk/oceans)"

"Autonomous Underwater Vehicle Mission Log"

"Recorded from: RAUVER"

"Logging Started At: Fri Nov 24 11:25:29 2000

"

"Mission Location: "

"Mission Type: "

"Mission Objectives: "

"Comments: "

$jA196B415a128b128c128d128e255f255g255h597i0r2071s2010t2052u1240v0w5x0y0R5.17P4.80T4.7X-151.8Y263.0Z-401.3C83.1*

$jA197B160a128b128c128d128e255f255g255h595i0r2070s2011t2049u1241v0w5x0y0R5.62P4.61T4.6X-150.2Y263.7Z-401.0C83.0*

$jA197B355a128b128c128d128e255f255g255h596i0r2070s2010t2073u1240v0w5x0y0$R5.62P4.61T4.6X-150.2Y263.7Z-401.0C83.0*

$jA197B615a128b128c128d128e255f255g255h594i0r2069s1992t2047u1240v0w5x0y0$R5.55P4.54T4.6X-149.8Y263.0Z-401.6C83.0*

$jA197B825a128b128c128d128e255f255g255h597i0r2068s1986t2040u1240v0w5x0y0$R5.55P4.54T4.6X-149.8Y263.0Z-401.6C83.0*

$jA198B85a128b128c128d128e255f255g255h597i0r2068s2004t2082u1239v0w5x0y0$R5.55P4.54T4.6X-149.8Y263.0Z-401.6C83.0*

$jA198B365a128b128c128d128e255f255g255h593i0r2070s2011t2036u1240v0w5x0y0$R5.55P4.54T4.6X-149.8Y263.0Z-401.6C83.0*

$jA198B630a128b128c128d128e255f255g255h593i0r2072s1994t2050u1239v0w5x0y0$R5.59P4.59T4.7X-150.5Y262.4Z-401.9C83.0*

$jA198B825a128b128c128d128e255f255g255h593i0r2069s1994t2096u1240v0w5x0y0$R5.59P4.59T4.7X-150.5Y262.4Z-401.9C83.0*

... continues...

Appendix E

RECOVERY's Diagnostic Output Files (Full Text)

Shown below is a sample RECOVERY diagnostic log file generated and stored during the course of the Faulty Port Lift-Thruster Experiment (Section 7.2.7).

Full copies of all RECOVERY output is included on the attached cd-rom.

E.1 Bad Port Lift-Thruster, No Denomination

```
Residual detected between
Modelled Node: PORT_NAV_POWER_MODEL
 and Sensed Node: PORT_LEM at:
Tue Jul 31 14:13:15 2001
Plus milliseconds: 452

Number of Hits at Mission Time: 252.33
Component Level Diagnostic Iteration: 0
PORT_LEM_SENSOR: 1
Diagnostic Iteration 0 Started At:
Tue Jul 31 14:13:15 2001
Plus milliseconds: 452

Correlator Started at
Tue Jul 31 14:13:15 2001
Plus milliseconds: 452
Invoked at Mission Time: 252.33
Delta_Correlations:
Bad_Node: PORT_LEM
Delta_Matches:

Similar_Alarm_Correlations:
```

```
Recent_Comp_Correlations:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER

Correlator Finished at
Tue Jul 31 14:13:15 2001
Plus milliseconds: 462

Model Based Diagnosis Engines Started at
Tue Jul 31 14:13:15 2001
Plus milliseconds: 472

Engines invoked at: 252.33
Roll Engine Proposes:

Port Power Engine Proposes:
PORT_LIFT_THRUSTER
PORT_LAMP_CONTROL

Stbd Power Engine Proposes:
Model Based Diagnosis Engines Finished at
Tue Jul 31 14:13:15 2001
Plus milliseconds: 852

Number of Hits at Mission Time: 252.33
Component Level Diagnostic Iteration: 0
PORT_LIFT_THRUSTER: 2
STBD_LIFT_THRUSTER: 1
PORT_LAMP_CONTROL: 1
PORT_LEM_SENSOR: 1

Diagnostic Iteration 0 Finished At:
Tue Jul 31 14:13:15 2001
Plus milliseconds: 862

Exiting diagnostics at:
```

```
Tue Jul 31 14:13:15 2001
Plus milliseconds: 872

Residual detected between
Modelled Node: PORT_NAV_POWER_MODEL
 and Sensed Node: PORT_LEM at:
Tue Jul 31 14:13:15 2001
Plus milliseconds: 923

Number of Hits at Mission Time: 252.595
Component Level Diagnostic Iteration: 1
PORT_LIFT_THRUSTER: 2
PORT_LEM_SENSOR: 2
STBD_LIFT_THRUSTER: 1
PORT_LAMP_CONTROL: 1
Diagnostic Iteration 1 Started At:
Tue Jul 31 14:13:15 2001
Plus milliseconds: 933

Correlator Started at
Tue Jul 31 14:13:15 2001
Plus milliseconds: 933
Invoked at Mission Time: 252.595
Delta_Correlations:
Bad_Node: PORT_LEM
Delta_Matches:

Similar_Alarm_Correlations:

Recent_Comp_Correlations:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER

Correlator Finished at
Tue Jul 31 14:13:15 2001
Plus milliseconds: 943

Model Based Diagnosis Engines Started at
```

Tue Jul 31 14:13:15 2001

Plus milliseconds: 943

Engines invoked at: 252.595

Roll Engine Proposes:

Port Power Engine Proposes:

PORT_LIFT_THRUSTER

PORT_LAMP_CONTROL

Stbd Power Engine Proposes:

Model Based Diagnosis Engines Finished at

Tue Jul 31 14:13:16 2001

Plus milliseconds: 323

Number of Hits at Mission Time: 252.595

Component Level Diagnostic Iteration: 1

PORT_LIFT_THRUSTER: 4

STBD_LIFT_THRUSTER: 2

PORT_LAMP_CONTROL: 2

PORT_LEM_SENSOR: 2

Diagnostic Iteration 1 Finished At:

Tue Jul 31 14:13:16 2001

Plus milliseconds: 333

Exiting diagnostics at:

Tue Jul 31 14:13:16 2001

Plus milliseconds: 333

Residual detected between

Modelled Node: PORT_NAV_POWER_MODEL

and Sensed Node: PORT_LEM at:

Tue Jul 31 14:13:16 2001

Plus milliseconds: 393

Number of Hits at Mission Time: 252.865

```
Component Level Diagnostic Iteration: 2
PORT_LIFT_THRUSTER: 4
PORT_LEM_SENSOR: 3
STBD_LIFT_THRUSTER: 2
PORT_LAMP_CONTROL: 2
Diagnostic Iteration 2 Started At:
Tue Jul 31 14:13:16 2001
Plus milliseconds: 403

Correlator Started at
Tue Jul 31 14:13:16 2001
Plus milliseconds: 403
Invoked at Mission Time: 252.865
Delta_Correlations:
Bad_Node: PORT_LEM
Delta_Matches:
GYRO_PITCH

Similar_Alarm_Correlations:

Recent_Comp_Correlations:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER

Correlator Finished at
Tue Jul 31 14:13:16 2001
Plus milliseconds: 413

Model Based Diagnosis Engines Started at
Tue Jul 31 14:13:16 2001
Plus milliseconds: 413

Engines invoked at: 252.865
Roll Engine Proposes:

Port Power Engine Proposes:
PORT_LIFT_THRUSTER
PORT_LAMP_CONTROL
```

```
Stbd Power Engine Proposes:
Model Based Diagnosis Engines Finished at
Tue Jul 31 14:13:16 2001
Plus milliseconds: 814

Number of Hits at Mission Time: 252.865
Component Level Diagnostic Iteration: 2
PORT_LIFT_THRUSTER: 6
STBD_LIFT_THRUSTER: 3
PORT_LAMP_CONTROL: 3
PORT_LEM_SENSOR: 3
PITCH_GYRO_SENSOR: 1

Diagnostic Iteration 2 Finished At:
Tue Jul 31 14:13:16 2001
Plus milliseconds: 824

Exiting diagnostics at:
Tue Jul 31 14:13:16 2001
Plus milliseconds: 824

Residual detected between
Modelled Node: PORT_NAV_POWER_MODEL
 and Sensed Node: PORT_LEM at:
Tue Jul 31 14:13:16 2001
Plus milliseconds: 894

Number of Hits at Mission Time: 253.13
Component Level Diagnostic Iteration: 3
PORT_LIFT_THRUSTER: 6
PORT_LEM_SENSOR: 4
STBD_LIFT_THRUSTER: 3
PORT_LAMP_CONTROL: 3
PITCH_GYRO_SENSOR: 1
Diagnostic Iteration 3 Started At:
Tue Jul 31 14:13:16 2001
```

```
Plus milliseconds: 894

Correlator Started at
Tue Jul 31 14:13:16 2001
Plus milliseconds: 904
Invoked at Mission Time: 253.13
Delta_Correlations:
Bad_Node: PORT_LEM
Delta_Matches:
GYRO_PITCH

Similar_Alarm_Correlations:

Recent_Comp_Correlations:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER

Correlator Finished at
Tue Jul 31 14:13:16 2001
Plus milliseconds: 904

Model Based Diagnosis Engines Started at
Tue Jul 31 14:13:16 2001
Plus milliseconds: 904

Engines invoked at: 253.13
Roll Engine Proposes:

Port Power Engine Proposes:
PORT_LIFT_THRUSTER
PORT_LAMP_CONTROL

Stbd Power Engine Proposes:
Model Based Diagnosis Engines Finished at
Tue Jul 31 14:13:17 2001
Plus milliseconds: 304

Number of Hits at Mission Time: 253.13
```

Component Level Diagnostic Iteration: 3

PORT_LIFT_THRUSTER: 8

STBD_LIFT_THRUSTER: 4

PORT_LAMP_CONTROL: 4

PORT_LEM_SENSOR: 4

PITCH_GYRO_SENSOR: 2

Diagnostic Iteration 3 Finished At:

Tue Jul 31 14:13:17 2001

Plus milliseconds: 315

Exiting diagnostics at:

Tue Jul 31 14:13:17 2001

Plus milliseconds: 315

Residual detected between

Modelled Node: PORT_NAV_POWER_MODEL

and Sensed Node: PORT_LEM at:

Tue Jul 31 14:13:17 2001

Plus milliseconds: 375

Number of Hits at Mission Time: 253.33

Component Level Diagnostic Iteration: 4

PORT_LIFT_THRUSTER: 8

PORT_LEM_SENSOR: 5

STBD_LIFT_THRUSTER: 4

PORT_LAMP_CONTROL: 4

PITCH_GYRO_SENSOR: 2

Diagnostic Iteration 4 Started At:

Tue Jul 31 14:13:17 2001

Plus milliseconds: 385

Correlator Started at

Tue Jul 31 14:13:17 2001

Plus milliseconds: 385

Invoked at Mission Time: 253.33

Delta_Correlations:

```
Bad_Node: PORT_LEM
Delta_Matches:
GYRO_PITCH

Similar_Alarm_Correlations:

Recent_Comp_Correlations:

Correlator Finished at
Tue Jul 31 14:13:17 2001
Plus milliseconds: 395

Model Based Diagnosis Engines Started at
Tue Jul 31 14:13:17 2001
Plus milliseconds: 395

Engines invoked at: 253.33
Roll Engine Proposes:

Port Power Engine Proposes:
PORT_LIFT_THRUSTER
PORT_LAMP_CONTROL

Stbd Power Engine Proposes:
Model Based Diagnosis Engines Finished at
Tue Jul 31 14:13:17 2001
Plus milliseconds: 805

Number of Hits at Mission Time: 253.33
Component Level Diagnostic Iteration: 4
PORT_LIFT_THRUSTER: 9
PORT_LAMP_CONTROL: 5
PORT_LEM_SENSOR: 5
STBD_LIFT_THRUSTER: 4
PITCH_GYRO_SENSOR: 3


Diagnostic Iteration 4 Finished At:
```

```
Tue Jul 31 14:13:17 2001
Plus milliseconds: 815

Exiting diagnostics at:
Tue Jul 31 14:13:17 2001
Plus milliseconds: 815

Residual detected between
Modelled Node: PORT_NAV_POWER_MODEL
 and Sensed Node: PORT_LEM at:
Tue Jul 31 14:13:17 2001
Plus milliseconds: 875

Residual detected between
Modelled Node: ROLL_MODEL
 and Sensed Node: AOSI_ROLL at:
Tue Jul 31 14:13:17 2001
Plus milliseconds: 875

Number of Hits at Mission Time: 253.61
Component Level Diagnostic Iteration: 5
PORT_LIFT_THRUSTER: 9
PORT_LEM_SENSOR: 6
PORT_LAMP_CONTROL: 5
STBD_LIFT_THRUSTER: 4
PITCH_GYRO_SENSOR: 3
AOSI_EZ3_COMPASS_SENSOR: 1
Diagnostic Iteration 5 Started At:
Tue Jul 31 14:13:17 2001
Plus milliseconds: 875

Correlator Started at
Tue Jul 31 14:13:17 2001
Plus milliseconds: 885
Invoked at Mission Time: 253.61
Delta_Correlations:
Bad_Node: PORT_LEM
Delta_Matches:
```

```
GYRO_PITCH
Bad_Node: AOSI_ROLL
Delta_Matches:
PORT_LAMP
STBD_LAMP
PORT_DIODE_TEMP

Similar_Alarm_Correlations:

Recent_Comp_Correlations:

Correlator Finished at
Tue Jul 31 14:13:17 2001
Plus milliseconds: 895

Model Based Diagnosis Engines Started at
Tue Jul 31 14:13:17 2001
Plus milliseconds: 895

Engines invoked at: 253.61
Roll Engine Proposes:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER

Port Power Engine Proposes:
PORT_LIFT_THRUSTER
PORT_LAMP_CONTROL

Stbd Power Engine Proposes:
Model Based Diagnosis Engines Finished at
Tue Jul 31 14:13:18 2001
Plus milliseconds: 406

Number of Hits at Mission Time: 253.61
Component Level Diagnostic Iteration: 5
PORT_LIFT_THRUSTER: 11
PORT_LAMP_CONTROL: 7
PORT_LEM_SENSOR: 6
```

```
STBD_LIFT_THRUSTER: 5
PITCH_GYRO_SENSOR: 4
STBD_LAMP_CONTROL: 1
PORT_POWERSTACK_TEMP_SENSOR: 1
AOSI_EZ3_COMPASS_SENSOR: 1


Diagnostic Iteration 5 Finished At:
Tue Jul 31 14:13:18 2001
Plus milliseconds: 416

Exiting diagnostics at:
Tue Jul 31 14:13:18 2001
Plus milliseconds: 416

Residual detected between
Modelled Node: PORT_NAV_POWER_MODEL
 and Sensed Node: PORT_LEM at:
Tue Jul 31 14:13:18 2001
Plus milliseconds: 476

Residual detected between
Modelled Node: ROLL_MODEL
 and Sensed Node: AOSI_ROLL at:
Tue Jul 31 14:13:18 2001
Plus milliseconds: 476

Number of Hits at Mission Time: 253.88
Component Level Diagnostic Iteration: 6
PORT_LIFT_THRUSTER: 11
PORT_LAMP_CONTROL: 7
PORT_LEM_SENSOR: 7
STBD_LIFT_THRUSTER: 5
PITCH_GYRO_SENSOR: 4
AOSI_EZ3_COMPASS_SENSOR: 2
STBD_LAMP_CONTROL: 1
PORT_POWERSTACK_TEMP_SENSOR: 1
Diagnostic Iteration 6 Started At:
```

```
Tue Jul 31 14:13:18 2001
Plus milliseconds: 486

Correlator Started at
Tue Jul 31 14:13:18 2001
Plus milliseconds: 486
Invoked at Mission Time: 253.88
Delta_Correlations:
Bad_Node: PORT_LEM
Delta_Matches:
GYRO_PITCH
Bad_Node: AOSI_ROLL
Delta_Matches:
PORT_LAMP
STBD_LAMP
GYRO_PITCH
PORT_DIODE_TEMP

Similar_Alarm_Correlations:

Recent_Comp_Correlations:

Correlator Finished at
Tue Jul 31 14:13:18 2001
Plus milliseconds: 496

Model Based Diagnosis Engines Started at
Tue Jul 31 14:13:18 2001
Plus milliseconds: 506

Engines invoked at: 253.88
Roll Engine Proposes:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER

Port Power Engine Proposes:
PORT_LIFT_THRUSTER
PORT_LAMP_CONTROL
```

Stbd Power Engine Proposes:

Model Based Diagnosis Engines Finished at

Tue Jul 31 14:13:19 2001

Plus milliseconds: 27

Number of Hits at Mission Time: 253.88

Component Level Diagnostic Iteration: 6

PORT_LIFT_THRUSTER: 13

PORT_LAMP_CONTROL: 9

PORT_LEM_SENSOR: 7

STBD_LIFT_THRUSTER: 6

PITCH_GYRO_SENSOR: 6

STBD_LAMP_CONTROL: 2

PORT_POWERSTACK_TEMP_SENSOR: 2

AOSI_EZ3_COMPASS_SENSOR: 2

Diagnostic Iteration 6 Finished At:

Tue Jul 31 14:13:19 2001

Plus milliseconds: 37

Exiting diagnostics at:

Tue Jul 31 14:13:19 2001

Plus milliseconds: 37

Residual detected between

Modelled Node: PORT_NAV_POWER_MODEL

and Sensed Node: PORT_LEM at:

Tue Jul 31 14:13:19 2001

Plus milliseconds: 97

Residual detected between

Modelled Node: ROLL_MODEL

and Sensed Node: AOSI_ROLL at:

Tue Jul 31 14:13:19 2001

Plus milliseconds: 97

```
Number of Hits at Mission Time: 254.085
Component Level Diagnostic Iteration: 7
PORT_LIFT_THRUSTER: 13
PORT_LAMP_CONTROL: 9
PORT_LEM_SENSOR: 8
STBD_LIFT_THRUSTER: 6
PITCH_GYRO_SENSOR: 6
AOSI_EZ3_COMPASS_SENSOR: 3
STBD_LAMP_CONTROL: 2
PORT_POWERSTACK_TEMP_SENSOR: 2
Diagnostic Iteration 7 Started At:
Tue Jul 31 14:13:19 2001
Plus milliseconds: 97

Correlator Started at
Tue Jul 31 14:13:19 2001
Plus milliseconds: 107
Invoked at Mission Time: 254.085
Delta_Correlations:
Bad_Node: PORT_LEM
Delta Matches:
Bad_Node: AOSI_ROLL
Delta_Matches:
PORT_LAMP
STBD_LAMP
GYRO_PITCH
PORT_DIODE_TEMP

Similar_Alarm_Correlations:

Recent_Comp_Correlations:

Correlator Finished at
Tue Jul 31 14:13:19 2001
Plus milliseconds: 117

Model Based Diagnosis Engines Started at
Tue Jul 31 14:13:19 2001
```

Plus milliseconds: 117

Engines invoked at: 254.085

Roll Engine Proposes:

PORT_LIFT_THRUSTER

STBD_LIFT_THRUSTER

Port Power Engine Proposes:

PORT_LIFT_THRUSTER

PORT_LAMP_CONTROL

Stbd Power Engine Proposes:

Model Based Diagnosis Engines Finished at

Tue Jul 31 14:13:19 2001

Plus milliseconds: 648

Number of Hits at Mission Time: 254.085

Component Level Diagnostic Iteration: 7

PORT_LIFT_THRUSTER: 15

PORT_LAMP_CONTROL: 11

PORT_LEM_SENSOR: 8

STBD_LIFT_THRUSTER: 7

PITCH_GYRO_SENSOR: 7

STBD_LAMP_CONTROL: 3

PORT_POWERSTACK_TEMP_SENSOR: 3

AOSI_EZ3_COMPASS_SENSOR: 3

Diagnostic Iteration 7 Finished At:

Tue Jul 31 14:13:19 2001

Plus milliseconds: 658

Exiting diagnostics at:

Tue Jul 31 14:13:19 2001

Plus milliseconds: 668

Residual detected between

Modelled Node: PORT_NAV_POWER_MODEL

```
 and Sensed Node: PORT_LEM at:
Tue Jul 31 14:13:19 2001
Plus milliseconds: 718

Residual detected between
Modelled Node: ROLL_MODEL
 and Sensed Node: AOSI_ROLL at:
Tue Jul 31 14:13:19 2001
Plus milliseconds: 718

Number of Hits at Mission Time: 254.29
Component Level Diagnostic Iteration: 8
PORT_LIFT_THRUSTER: 15
PORT_LAMP_CONTROL: 11
PORT_LEM_SENSOR: 9
STBD_LIFT_THRUSTER: 7
PITCH_GYRO_SENSOR: 7
AOSI_EZ3_COMPASS_SENSOR: 4
STBD_LAMP_CONTROL: 3
PORT_POWERSTACK_TEMP_SENSOR: 3
Diagnostic Iteration 8 Started At:
Tue Jul 31 14:13:19 2001
Plus milliseconds: 728

Correlator Started at
Tue Jul 31 14:13:19 2001
Plus milliseconds: 728
Invoked at Mission Time: 254.29
Delta_Correlations:
Bad_Node: PORT_LEM
Delta_Matches:
Bad_Node: AOSI_ROLL
Delta_Matches:
PORT_LAMP
STBD_LAMP
GYRO_PITCH
PORT_DIODE_TEMP
```

```
Similar_Alarm_Correlations:

Recent_Comp_Correlations:

Correlator Finished at
Tue Jul 31 14:13:19 2001
Plus milliseconds: 738

Model Based Diagnosis Engines Started at
Tue Jul 31 14:13:19 2001
Plus milliseconds: 748

Engines invoked at: 254.29
Roll Engine Proposes:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER

Port Power Engine Proposes:
PORT_LIFT_THRUSTER
PORT_LAMP_CONTROL

Stbd Power Engine Proposes:
Model Based Diagnosis Engines Finished at
Tue Jul 31 14:13:20 2001
Plus milliseconds: 249

Number of Hits at Mission Time: 254.29
Component Level Diagnostic Iteration: 8
PORT_LIFT_THRUSTER: 17
PORT_LAMP_CONTROL: 13
PORT_LEM_SENSOR: 9
STBD_LIFT_THRUSTER: 8
PITCH_GYRO_SENSOR: 8
STBD_LAMP_CONTROL: 4
PORT_POWERSTACK_TEMP_SENSOR: 4
AOSI_EZ3_COMPASS_SENSOR: 4
```

Diagnostic Iteration 8 Finished At:

Tue Jul 31 14:13:20 2001

Plus milliseconds: 289

Exiting diagnostics at:

Tue Jul 31 14:13:20 2001

Plus milliseconds: 289

Residual detected between

Modelled Node: PORT_NAV_POWER_MODEL

and Sensed Node: PORT_LEM at:

Tue Jul 31 14:13:20 2001

Plus milliseconds: 349

Residual detected between

Modelled Node: ROLL_MODEL

and Sensed Node: AOSI_ROLL at:

Tue Jul 31 14:13:20 2001

Plus milliseconds: 349

Number of Hits at Mission Time: 254.565

Component Level Diagnostic Iteration: 9

PORT_LIFT_THRUSTER: 17

PORT_LAMP_CONTROL: 13

PORT_LEM_SENSOR: 10

STBD_LIFT_THRUSTER: 8

PITCH_GYRO_SENSOR: 8

AOSI_EZ3_COMPASS_SENSOR: 5

STBD_LAMP_CONTROL: 4

PORT_POWERSTACK_TEMP_SENSOR: 4

Diagnostic Iteration 9 Started At:

Tue Jul 31 14:13:20 2001

Plus milliseconds: 359

Correlator Started at

Tue Jul 31 14:13:20 2001

Plus milliseconds: 359

Invoked at Mission Time: 254.565

```
Delta_Correlations:
Bad_Node: PORT_LEM
Delta_Matches:
Bad_Node: AOSI_ROLL
Delta_Matches:
GYRO_PITCH

Similar_Alarm_Correlations:

Recent_Comp_Correlations:

Correlator Finished at
Tue Jul 31 14:13:20 2001
Plus milliseconds: 369

Model Based Diagnosis Engines Started at
Tue Jul 31 14:13:20 2001
Plus milliseconds: 369

Engines invoked at: 254.565
Roll Engine Proposes:
PORT_LIFT_THRUSTER
STBD_LIFT_THRUSTER

Port Power Engine Proposes:
PORT_LIFT_THRUSTER
PORT_LAMP_CONTROL

Stbd Power Engine Proposes:
Model Based Diagnosis Engines Finished at
Tue Jul 31 14:13:20 2001
Plus milliseconds: 910

Number of Hits at Mission Time: 254.565
Component Level Diagnostic Iteration: 9
PORT_LIFT_THRUSTER: 19
PORT_LAMP_CONTROL: 14
PORT_LEM_SENSOR: 10
```

```
STBD_LIFT_THRUSTER: 9
PITCH_GYRO_SENSOR: 9
AOSI_EZ3_COMPASS_SENSOR: 5
STBD_LAMP_CONTROL: 4
PORT_POWERSTACK_TEMP_SENSOR: 4

Diagnostic Iteration 9 Finished At:
Tue Jul 31 14:13:20 2001
Plus milliseconds: 920

Domain Independent Diagnostics (post iterations) Started at:
Tue Jul 31 14:13:20 2001
Plus milliseconds: 920

Nodal Power Supply Diagnosis:
PORT_BATTERY  Fault Rating: 0.
STBD_BATTERY  Fault Rating: 0.
PORT_NAV_POWER  Fault Rating: 1. (nominated)
STBD_NAV_POWER  Fault Rating: 1. (nominated)
TWELVE_PCONTROL  Fault Rating: 0.5
TWENTYFOUR_PCONTROL  Fault Rating: 0.
COMMS_POWER  Fault Rating: 0.
FIVE_PCONTROL  Fault Rating: 0.1
FIFTEEN_PCONTROL  Fault Rating: 0.5
MINUSFIFTEEN_PCONTROL  Fault Rating: 0.5
Domain Independent Diagnostics (post iterations) Finished at:
Tue Jul 31 14:13:20 2001
Plus milliseconds: 930

Number of Hits at Mission Time: 254.565
Component Level Diagnostic Iteration: 10
PORT_LIFT_THRUSTER: 19
PORT_LAMP_CONTROL: 14
PORT_LEM_SENSOR: 10
STBD_LIFT_THRUSTER: 9
PITCH_GYRO_SENSOR: 9
AOSI_EZ3_COMPASS_SENSOR: 5
```

STBD_LAMP_CONTROL: 4

PORT_POWERSTACK_TEMP_SENSOR: 4

PORT_NAV_POWER: 1

STBD_NAV_POWER: 1

Component-Specific Diagnostic Level Entered at

Tue Jul 31 14:13:20 2001

Plus milliseconds: 930

Adding Related Sub-Components to Suspicion Index:

Final Suspicion Index at Mission Time: 254.565

Sub-Component Diagnostics Level

PORT_LIFT_THRUSTER: 19

PORT_LIFT_THRUST_CONTROL: 0

PORT_LIFT_MOTOR: 0

PORT_LIFT_THRUSTER_BRUSH: 0

PORT_LIFT_GBOX: 0

PORT_LIFT_PROPELLER: 0

PORT_LAMP_CONTROL: 14

PORT_LEM_SENSOR: 10

STBD_LIFT_THRUSTER: 9

STBD_LIFT_THRUST_CONTROL: 0

STBD_LIFT_MOTOR: 0

STBD_LIFT_THRUSTER_BRUSH: 0

STBD_LIFT_GBOX: 0

STBD_LIFT_PROPELLER: 0

PITCH_GYRO_SENSOR: 9

AOSI_EZ3_COMPASS_SENSOR: 5

STBD_LAMP_CONTROL: 4

PORT_POWERSTACK_TEMP_SENSOR: 4

PORT_NAV_POWER: 1

STBD_NAV_POWER: 1

Dynamic Rule Testing of Component: PORT_LIFT_THRUSTER Started At:

Tue Jul 31 14:13:20 2001

```
Plus milliseconds: 940

Scan finished (no match found) at:
Tue Jul 31 14:13:21 2001
Plus milliseconds: 771

Dynamic Rule Testing of Component: PORT_LIFT_THRUST_CONTROL Started At:
Tue Jul 31 14:13:21 2001
Plus milliseconds: 891

Exact Match Found at:
Tue Jul 31 14:13:22 2001
Plus milliseconds: 482

Dynamic Rule Attached to Component: PORT_LIFT_THRUST_CONTROL

Component-Specific Information:
Exact Match found with Component: PORT_LIFT_THRUST_CONTROL at:
Tue Jul 31 14:13:22 2001
Plus milliseconds: 482

Exiting diagnostics at:
Tue Jul 31 14:13:22 2001
Plus milliseconds: 492
```

References

[1] J. S. Albus, R. Quintero, and R. Lumia. An overview of NASREM: The NASA/NBS standard reference model for telerobot control system architecture. Technical Report NISTIR 5412, National Institute of Standards and Technology, USA, April 1994.

[2] A. A. Aldea-Corrales. *The Use of Scheduling and Hierarchical Modelling Techniques for Time-Limited Diagnosis.* PhD thesis, Heriot-Watt University, Edinburgh, UK, July 1994.

[3] A. Alessandri, M. Caccia, and G. Veruggio. A model-based approach to fault diagnosis in unmanned underwater vehicles. In *OCEANS '98 Conference Proceedings*, volume 2, 1998.

[4] D. Bartz and R. Sallade. Creating a diagnostic engineering toolset. In *AUTOTESTCON '94. IEEE Systems Readiness Technology Conference. 'Cost Effective Support Into the Next Century', Conference Proceedings.*, pages 587–594, 1994.

[5] T. Bearse and A. Carrier. Bridging the testability analysis/test program gap -an approach that builds on the navy's integrated diagnostics support system. In *AUTOTESTCON '93. IEEE Systems Readiness Technology Conference. Proceedings , Sept. 1993*, pages 459–465, 1993.

[6] T. Bearse, H. Dill, and M. Lynch. Application of model based diagnostics to telecommunication satellites. In *AUTOTESTCON '98. IEEE Systems Readiness Technology Conference., 1998 IEEE*, pages 392–397, 1998.

[7] R. Belhassine-Cherif and A. Ghedamsi. Diagnostic tests for communicating nondeterministic finite state machines. In *Proceedings of the Fifth IEEE Symposium on Computers and Communications, 2000.*, pages 424 –429. IEEE, 2000.

[8] D. E. Bernard et al. Design of the remote agent experiment for spacecraft autonomy. In *IEEE Aerospace Conference*, volume 2. IEEE, 1998.

[9] J. Bohr. Diagnostic metrics-a critical element of the diagnostic architecture. In *AUTOTESTCON '99. IEEE Systems Readiness Technology Conference, 1999. IEEE*, pages 215–221, 1999.

[10] V. Boppana and M. Fujita. Modeling the unknown! towards model-independent fault and error diagnosis. In *Proceedings of the International Test Conference.*

[11] A. Bos, A. V. Gemund, and C. Witteveen. Model-based diagnosis support for satellite-based instruments. In *IEEE AUTOTESTCON Proceedings.*

[12] K. Bossley, M. Brown, and C. Harris. Neurofuzzy indentification of an autonomous underwater vehicle. *International Journal of Systems Science*, 30(9):901–913, 1999.

[13] P. Brassel and R. Burkhart. Army aviation readies for the twenty-first century. In *AUTOTESTCON '92. IEEE Systems Readiness Technology Conference, Conference Record, pages = 373-378, year = 1992,.*

[14] M. Brazet. The application of integrated diagnostics techniques to heavy duty vehicle testing. In *AUTOTESTCON '92. IEEE Systems Readiness Technology Conference, Conference Record*, pages 31–35, 1992.

[15] M. Brazet. Aegis orts - the first and future ultimate integrated diagnostics system. *IEEE Aerospace and Electronics Systems Magazine*, 9:40–45, 1994.

[16] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 1986.

[17] A. Buchner, S. Anand, D. Bell, and J. Hughes. A framework for discovering knowledge from distributed and heterogeneous databases. In *IEEE Colloqium on Knowledge Discovery and Data Mining*, pages 8/1–8/4, 1996.

[18] K. Butler and J. Momoh. A neural net based approach for fault diagnosis in distribution networks. In *Power Engineering Society Winter Meeting, 2000*, volume 2. IEEE, 1999.

[19] M. Caccia, G. Indiveri, and G. Veruggio. Modeling and identification of open-frame variable configuration unmanned underwater vehicles. *IEEE Journal of Oceanic Engineering*, 25:227–240, April 2000.

[20] J. Cattarius and D. Inman. Experimental verification of intelligent fault detec-

tion in rotor blades. *International Journal of Systems Science*, 31(11):1375–1379, 2000.

[21] A. Cavallini, M. Conti, A. Contin, G. Montanari, and G. Pasini. An integrated diagnostic tool based an pd measurements. In *Electrical Insulation Conference and Electrical Manufacturing and Coil Winding Conference, 2001. Proceedings*, pages 219–224, 2001.

[22] K. Cavanaugh. An integrated diagnostics virtual test bench for life cycle support. In *Aerospace Conference, 2001, IEEE Proceedings.*, volume 7, pages 3235–3246, 2001.

[23] M. Chantler, G. Coghill, Q. Shen, and R. Leitch. Selecting tools and techniques for model-based diagnosis. *Artificial Intelligence in Engineering 12*, pages 81–98, May 1998.

[24] P. Chen, Y. Sasaki, S. Nakayama, and T. Toyota. Plant inspection and diagnosis robot for the detection of a faulty machine part by ga control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems*, 1998.

[25] J. Cigler. An integrated diagnostic support system approach to fault isolation in the operational environment. In *Proceedings of the IEEE 1989 National Aerospace and Electronics Conference, 1989. NAECON 1989.*, volume 3, pages 1321–1328, 1989.

[26] M. Courtoise and J. Pouilly. Integrated diagnostic for new french fighter. In *AUTOTESTCON '92. IEEE Systems Readiness Technology Conference, Conference Record*, pages 425–433, 1992.

[27] A. A. da Silva, A. Insfran, P. da Silveira, and G. Lambert-Torres. Neural networks for fault location in substations. *IEEE transactions on power delivery*, 11(1):234–239, January 1996.

[28] F. Dai and M. Sugisaka. Fuzzy fault query approach in the fault diagnosis of a power plant system. In *Proceedings of the IEEE International Conference on Intelligent Robots & Systems, 1999*, pages 752–756, 1999.

[29] S. Dao and B. Perry. An overview of data mining in heterogeneous schema integration. In *WESCON96*, pages 478–483, 1996.

[30] J. de Kleer and B. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 4:97–130, 1987.

[31] J. Dean. Integrated diagnostics: confusion and solutions. In *AUTOTESTCON '96, Test Technology and Commercialization. Conference Record*, pages 436–440, 1996.

[32] S. Deb, K. Pattipati, and R. Shrestha. Qsi's integrated diagnostics toolset. In *AUTOTESTCON, 97. 1997 IEEE Autotestcon Proceedings*, pages 408–421, 1997.

[33] B. Deuker, M. Perrier, and B. Amy. Fault diagnosis of subsea robots using neuro-symbolic hybrid systems. In *IEEE International Conference on Robotics & Automation*, 1998.

[34] D. Doel. Accuracy quantification for integrated diagnostics. In *Proceedings of the IEEE 1991 National Aerospace and Electronics Conference, 1991. NAECON*, volume 3, pages 1225–1234, 1991.

[35] R. Dunia and S. Qin. Subspace approach to multidimensional fault identification and reconstruction. *AICHe Journal 44(8)*, pages 1813–1831, 1998.

[36] R. Dunia and S. Qin. A unified geometric approach to process and sensor fault identification and reconstruction: the unidimensional fault case. *Computer and Chemical Engineering 22*, pages 927–943, 1998.

[37] J. Finlay and A. Dix. *An Introduction to Artificial Intelligence.* UCI Press, London, UK, 1996.

[38] J. Franco. Using integrated diagnostics on automatic test equipment. In *AUTOTESTCON '91. IEEE Systems Readiness Technology Conference. Improving Systems Effectiveness in the Changing Environment of the '90s, Conference Record.*, pages 337–343, 1991.

[39] S. Freschi. Cost and benefit considerations for implementing an open systems approach to integrated diagnostics. In *AUTOTESTCON '99; The IEEE Systems Readiness Technology Conference*, pages 391–404. IEEE, August 30th to September 2nd 1999.

[40] S. Freschi et al. Open systems integrated diagnostics demonstration (OSAIDD) study. Technical report, Office of the Secretary of Defense, USA.

January 1999.

[41] K. Gaffney and P. Morones. Advanced-aircraft integrated-diagnostics system-concept evaluation. In *Reliability and Maintainability Symposium, 1994. Proceedings., Annual*, pages 20–25, 1994.

[42] M. Garbiras and K. Goebel. Fusing diagnostic information without a priori performance knowledge. In *Proceedings of the Third International Conference on Information Fusion, 2000*, volume 6, pages 9 –16. IEEE, 2000.

[43] S. Ghoshal, R. Shrestha, A. Ghoshal, V. Malepati, S. Deb, K. Patripati, and D. Kleinman. An integrated process for system maintenance, fault diagnosis and support. In *IEEE Aerospace Conference, 1999. Proceedings.*, volume 3, pages 129–138, 1999.

[44] K. Goebel, M. Krok, and H. Sutherland. Diagnostic information fusion: requirements flowdown and interface issues. In *Aerospace Conference Proceedings, 2000*, volume 6, pages 155 –162. IEEE, 2000.

[45] K. Hamilton, D. Lane, N. Taylor, and K. Brown. Enhancing auv fault-diagnosis capabilities with the RECOVERY system. *IEEE Journal of Oceanographic Engineering*, 2001.

[46] K. Hamilton, D. Lane, N. Taylor, and K. Brown. Fault diagnosis on autonomous robotic vehicles with RECOVERY: An integrated heterogeneous-knowledge approach. In *IEEE International Conference on Robotics & Automation*, volume 4, pages 3232 –3237. IEEE, May 2001.

[47] T. Hansen. Diagnosing multiple faults using knowledge about malfunction behaviour. In *First International Conference on Industrial Engineering Applications of AI and ES*, volume 1, pages 29–36, 1988.

[48] W. Hardman, A. Hess, and J. Scheaffer. Sh-60 helicopter integrated diagnostic system (hids) program-diagnostic and prognostic development experience. In *IEEE Aerospace Conference, 1999. Proceedings.*, volume 2, pages 473–491, 1999.

[49] W. Hardman, A. Hess, and J. Scheaffer. A helicopter powertrain diagnostics and prognostics demonstration. In *Aerospace Conference Proceedings, 2000 IEEE*, volume 6, pages 355–365, 2000.

[50] B. Havlicsek. Integrating diagnostic knowledge. *IEEE Aerospace and Electronics Systems Magazine*, 4:54–59, 1989.

[51] W. Hornfeld and E. Frezel. Intelligent auv on-board health monitoring software (INDOS). In *IEEE International Conference on Robotics & Automation*, 1998.

[52] Y.-C. Huan, H.-T. Yang, and C.-L. Huang. Developing a new transformer fault diagnosis system through evolutionary fuzzy logic. *IEEE Transactions on Power Delivery*, 12:761–767, 1997.

[53] B. Huang. Detection of abrupt changes of total least squares models and application in fault detection. *IEEE Transactions on Control Systems Technology*, 9:357–367, 2001.

[54] S.-Y. Huang and K.-T. Cheng. Errortracer: design error diagnosis based on fault simulation techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18:1341–1352.

[55] M. Hunt, B. von Konsky, S. Venkatesh, and P. Petros. Bayesian networks and decision trees in the diagnosis of female urinary incontinence. In *Proceedings of the 22nd annual EMBS international conference*, pages 551–554. IEEE, July 2000.

[56] C. Hunter. Integrated diagnostics for embedded memory built-in self test on powerpc/sup tm/ devices. In *Computer Design: VLSI in Computers and Processors, 1997. ICCD '97. Proceedings.*, pages 549–554, 1997.

[57] G. Indiveri. *Modelling and indentification of underwater robotic systems.* PhD thesis, University of Genova, Genova, Italy, December 1998.

[58] E. Jackson. Real-time model-based fault detection and diagnosis for automated systems. In *Proceedings of the IEEE Industry Applications Society Dynamic Modeling Control Applications for Industry Workshop.*

[59] M. James and L. Dubon. Autonomous tools and techniques for reducing operational and maintenance costs in space planes. In *Aerospace Conference Proceedings, 1999*, volume 2, pages 145 –152. IEEE, 1999.

[60] M. James and L. Dubon. An autonomous diagnostic and prognostic monitoring system for nasa's deep space network. In *Aerospace Conference Proceedings, 2000*, volume 2, pages 403 –414. IEEE, 2000.

[61] K. Kolcio, M. Hanson, and L. Fesq. Validation of autonomous fault diagnostic software. In *1998 IEEE Aerospace Conference*, volume 4, 1998.

[62] J. Kurien and D. Clancy. Autonomous control of complex dynamical systems in support of a manned mission to mars. In *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials*, volume 2, 1999.

[63] M. Larsson. Diagnosis and analysis of diagnosis properties using discrete event dynamic systems. In *Proceedings of the 37th IEEE Conference on Decision and Control, 1998*, volume 4, 1998.

[64] D. Lavo, B. Chess, T. Larrabee, and I. Hartanto. Probabilistic mixed-model fault diagnosis. In *Proceedings of the International Test Conference, 1998.*

[65] R. Lebron and R. Rossi. Automated integrated diagnostic analysis for aircraft mechanical systems. In *AUTOTESTCON '92. IEEE Systems Readiness Technology Conference, Conference Record*, pages 265–270, 1992.

[66] L. Lewis. A case-based reasoning approach to the management of faults in communications networks. In *Proceedings of the Ninth Conference on Artificial Intelligence for Applications*, pages 114 –120. IEEE, 1993.

[67] S. Magrabi and P. Gibbens. Decentralised fault detection and diagnosis in navigation systems for unmanned aerial vehicles. In *Proceedings of the 2000 IEEE Position Location and Navigation Symposium*, pages 363–370. IEEE, 2000.

[68] J. Marcus, J. Mahanna, and R. Gruessner. Integrated diagnostic concepts for advanced technology rotorcraft. In *AUTOTESTCON '88. IEEE International Automatic Testing Conference, Futuretest. Symposium Proceedings*, pages 171–177, 1988.

[69] A. Mathur, S. Deb, and K. Pattipati. Modeling and real-time diagnostics in teams-rt. In *American Control Conference, 1998. Proceedings of*, volume 3, pages 1610–1614, 1998.

[70] A. Mathur, S. Ghoshal, D. Haste, C. Domagala, R. Shrestha, V. Malepati, and K. Pattipati. An integrated support system for rotorcraft health management and maintenance. In *Aerospace Conference Proceedings, 2000 IEEE*, volume 6,

pages 1–8, 2000.

[71] P. McCown. Abet-an architecture for integrated diagnostics? In *AUTOTESTCON '92. IEEE Systems Readiness Technology Conference, Conference Record*, pages 259–263, 1992.

[72] M. Mimnagh, W. Hardman, and J. Schaeffer. Helicopter drive system diagnostics through multivariate statistical process control. In *Aerospace Conference Proceedings, IEEE, 2000*, volume 6, pages 381–415, 2000.

[73] A. Misra, G. Provan, G. Karsai, G. Bloor, and E. Scarl. A generic and symbolic model-based diagnostic reasoner with highly scalable properties. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics, 1998*, volume 4, 1998.

[74] J. Molnar. Summary and applicability of analog fault detection/isolation techniques. In *IEEE Autotestcon Proceedings*, pages 383 –389, 1997.

[75] P. J. Mosterman and G. Biswas. Model based diagnosis of dynamic systems. In *Seventh Journees du L.I.P.N.*, pages 143–154, University of Paris-Nord, Villetaneuse, France, September 1997.

[76] C. Ng and K. Chow. An expert system for diagnosis of electronic equipment using structural model and trouble-shooting heuristics. In *Proceedings of the Fourth IEEE Region 10 International Conference, TENCON'89.*

[77] M. Nolan. Integrated diagnostics: needs, technologies and current initiatives. In *Aerospace and Electronics Conference, 1989. NAECON 1989., Proceedings of the IEEE 1989 National*, volume 3, pages 1252–1256, 1989.

[78] S. Ofsthun. An approach to intelligent integrated diagnostic design tools. In *AUTOTESTCON '91. IEEE Systems Readiness Technology Conference. Improving Systems Effectiveness in the Changing Environment of the '90s, Conference Record.*, pages 319–328, 1991.

[79] R. Patton, J. Chen, and H. Benkhadda. A study on neuro-fuzzy systems for fault diagnosis. *International Journal of Systems Science*, 31(11):1441–1448, 2000.

[80] R. Patton, J. Chen, and C. Lopez-Toribio. Fuzzy observers for non-linear dynamic systems fault diagnosis. In *IEEE International Conference on Decision*

& Control, 1998.

[81] G. Penido, J. Nogeuira, and C. Machado. An automatic fault diagnosis and correction system for telecommunications management. In *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, 1999. Distributed Management for the Networked Millennium*, pages 777–791. IEEE, 1999.

[82] D. Perrault and M. Nahon. Fault tolerant control of an autonomous underwater vehicle. In *OCEANS'98 Conference Proceedings*, volume 2. IEEE, 1998.

[83] L. Portinale. Behavioral petri nets: a model for diagnostic knowledge representation and reasoning. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 27:184–195, April 1997.

[84] C. Price. *Computer-Based Diagnostic Systems.* Springer, London, UK, 1999.

[85] C. Price and N. Taylor. Multiple fault diagnosis from FMEA. In *Proceedings of AAAI-97*, Providence, Rhode Island, USA, 1997. IAAI-97.

[86] R. Price. Back to the future for integrated diagnostics (electronic military systems). In *AUTOTESTCON '88. IEEE International Automatic Testing Conference, Futuretest. Symposium Proceedings*, pages 199–201, 1988.

[87] K. W. Pryztula and D. Thompson. Construction of bayesian networks for diagnostics. In *IEEE Aerospace Conference Proceedings, 2000*, volume 5, pages 193 –200, 2000.

[88] G. Rae and S. Dunn. On-line damage detection for autonomous underwater vehicles. In *Proceedings of the 1994 Symposium on Autonomous Underwater Vehicle Technology.*

[89] K. Ramkumar, M. Druckenmuller, Y. Xi, P. Philips, H. Presig, W. Ho, and K. Lim. A fault-detection and diagnosis scheme by dynamic computation of finite-state automaton tables. In *Proceedings of The 25th Annual Conference of the IEEE Industrial Electronics Society, 1999*, volume 2, pages 698 –703. IEEE, 1999.

[90] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach.* Prentice Hall International Editions, London, UK, 1995.

[91] P. Schaefer, R. Colgren, R. Abbott, H. Park, A. Fijany, F. Fisher, M. James,

S. Chien, R. Mackey, M. Zak, T. Johnson, and S. Bush. Technologies for reliable autonomous control (TRAC) of UAVs. In *Digital Avionics Systems Conferences, 2000*, volume 1, pages 1–7. IEEE, 2000.

[92] D. Schreckenghost, P. Bonasso, D. Kortenkamp, and D. Ryan. Three tier architecture for controlling space life support systems. In *Proceedings of the IEEE International Joint Symposia on Intelligence and Systems, 1998*, 1998.

[93] Q. Shen and R. Leitch. Diagnosing continuous systems with qualitative dynamic models. *Artificial Intelligence in Engineering 9*, pages 107–125, 1995.

[94] R. Spina and S. Upadhyaya. Linear circuit fault diagnosis using neuromorphic analyzers. *IEEE transactions on circuits and systems-II: analogue and digital signal processing*, 44(3):188–196, March 1999.

[95] L. P. Su, M. Nolan, G. deMare, and D. Carey. Prognostics framework [for weapon systems health monitoring]. In *AUTOTESTCON Proceedings, IEEE Systems Readiness Technology Conference, 1999*, pages 661 –672. IEEE, 1999.

[96] L. P. Su, M. Nolan, G. deMare, and B. Norman. Prognostics framework-update ii. In *AUTOTESTCON Proceedings, 2000*, pages 497 –504. IEEE, 2000.

[97] M. Takai and T. Ura. Development of a system to diagnose autonomous underwater vehicles. *International Journal of Systems Science*, pages 981–988, September 1999.

[98] L. Tarassenko, A. Nairac, N. Townsend, I. Buxton, and P. Cowley. Novelty detection for the identification of abnormalities. *International Journal of Systems Science*, 31(11):1427–1439, 2000.

[99] A. B. Trunov and M. M. Polycarpou. Robust nonlinear fault diagnosis: application to robust systems. In *Proceedings of the 1999 IEEE international conference on control applications*, kohala coast, island of hawaii, USA, August 1999. IEEE.

[100] E. Tunstel, A. Howard, and H. Seraji. Fuzzy rule-based reasoning for rover safety and survivability. In *Proceedings of the IEEE International Conference on Robotics & Automation, 2001*, pages 1413–1420, Seoul, Korea, 2001.

[101] A. T. Vemuri, M. M. Polycarpou, and S. A. Diakourtis. Neural network based fault detection in robotic manipulators. *IEEE Transactions on Robotics and*

Automation, 14(2):342–348, April 1998.

[102] M. Vieira and C. Theys. Bayesian analysis for the fault detection of three-phase induction machine. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, 1998*, volume 4, pages 2237 –2240. IEEE, 1998.

[103] Z. Vukic, H. Ozbolt, and D. Pavlekovic. Improving fault handling in marine vehicle course-keeping systems. *IEEE Robotics and Automation Magazine*, pages 39–52, June 1999.

[104] R. Washington. On-board real-time state and fault identification for rovers. In *IEEE International Conference on Robotics & Automation*, 2000.

[105] Y. Xi, K.-W. Lim, W.-K. Ho, and H. Preisig. Diagnosability of faults using finite-state automaton model. In *Proceedings of TENCON 2000*, volume 2, pages 367 –371. IEEE, 2000.

[106] T. Yan, A. Nagamura, T. Arai, and N. Kuwahara. Concept design of remote fault diagnosis system for autonomous mobile robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2000.

[107] K. C. Yang, J. Yuh, and S. Choi. Experimental study of fault-tolerant system design for underwater robots. In *IEEE International Conference on Robotics & Automation*, pages 1051–1056, 1998.

[108] K. C. Yang, J. Yuh, and S. Choi. Fault-tolerant system design of an autonomous underwater vehicle - ODIN: an experimental study. *International Journal of Systems Science*, 30(9):1011–1019, 1999.

[109] G. G. Yen. Health monitoring of vibration signatures. In *Proceedings of the 23rd International Conference on Industrial Electronics, Control and Instrumentation*, volume 3. IEEE, 1997.

[110] G. G. Yen and K.-C. Lin. Conditional health monitoring using vibration signatures. In *Proceedings of the 38th Conference on Decision and Control*, Phoenix, Arizona, USA, December 1998.

[111] Y. Zhang and X. Li. Detection and diagnosis of sensor and actuator failures using imm estimator. *IEEE Transactions on Aerospace and Electronic Systems*,

34:1293–1313, 1998.

THESIS CONTAINS CD/DVD