Automated Fault Diagnosis
at Philips Medical Systems

A Model-Based Approach

Master’s Thesis

W.M. Lindhoud
Automated Fault Diagnosis at Philips Medical Systems

A Model-Based Approach

THESIS

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

W.M. Lindhoud

born in Gouda, the Netherlands

TU Delft
Software Engineering Research Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Philips Medical Systems
Veenpluis 4-6
Best, The Netherlands
www.medical.philips.com
Automated Fault Diagnosis at Philips Medical Systems

A Model-Based Approach

Author: W.M. Lindhoud
Student id: 1053884
Email: w.m.lindhoud@gmail.com

Abstract

While our machines get faster, better and cheaper every day the increase in complexity of these systems is huge. This is not different for the medical systems developed and serviced by Philips Medical Systems (PMS). Fault diagnosis is an essential key to keep these systems dependable. Currently, most fault diagnosis practices in industry are based on manual effort. An area that is not readily explored and exploited by industry, but could offer improvement, is automated fault diagnosis. Although, many useful mechanisms inside the PMS systems exist, so far, no research has been done on how to set up a diagnostic system. This work is a first exploration of the benefits that such a technique could have for the fault diagnosis of the Philips Cardio-Vascular X-Ray System. In particular, it defines the goals and qualities of a diagnostic approach in industry. Model-Based fault Diagnosis (MBD) is a reasoning technique for finding root causes of failures based upon a model. MBD suits the goals and desired attributes for a fault diagnosis approach at PMS best. A case study of a subsystem is used to examine what issues play a role when implementing a model-based approach to fault diagnosis. It is found that, at PMS, it is not possible to quantify the diagnostic accuracy of the current practice to fault diagnosis and MBD implementations. Consequently, it cannot be concluded that MBD improves fault diagnosis with respect to higher dependability of Philips Cardio-Vascular X-Ray Systems. However, by means of entropy it is shown that the accuracy of the model-based approach is expected to be higher than an approach in which experts define a mapping between symptoms and diagnoses. Further, it is shown that entropy is a valuable feedback mechanism to improve MBD implementations.

keywords: Model-Based Diagnosis, Fault Diagnosis, Service

Thesis Committee:

Chair: prof. dr. ir. A.J.C. van Gemund, Faculty EEMCS, TU Delft
University supervisor: dr. ir. H.G. Gross, Faculty EEMCS, TU Delft
Company supervisor: ing. B. Minderhoud, Philips Medical Systems
Committee Member: dr. M.M. de Weerdt, Faculty EEMCS, TU Delft
The work presented in this Master’s Thesis is a small step to complex medical systems that do not need any service, because these systems diagnose and repair themselves. Automated fault diagnosis is an interesting field to business, as well as an instructive topic for engineers. This thesis is written in completion of my Master’s program Computer Science at Delft University of Technology (TUD), in the group Software Engineering. The work is carried out at Philips Medical Systems (PMS), a leading producer and supplier of medical equipment. The business unit where I worked is called Cardio-Vascular, and is responsible for the development, sales, and service of Cardio-Vascular X-Ray systems. These systems allow doctors to successfully help people with cardiac and vascular diseases. Earlier, doing a graduation project at a company that does such good work, was simply just a daydream. Now, I find myself concluding it. I would like to thank Philips Medical Systems, and Ben Minderhoud in particular, for giving me that opportunity.

Although, a graduation assignment is mainly individual effort, many people have contributed to the success of this project. First of all, I would like to thank anybody who has been part in achieving this goal. Special thanks go to my company supervisor, Ben Minderhoud (PMS), and university supervisor, Hans-Gerhard Gross (TUD), for all constructive feedback. Not in the last place, the important role of Arjan van Gemund (TUD) in this project, as well as his supervision in an independent, but very instructive preceding research assignment, is gratefully acknowledged. I also would like to thank Alex Feldman, Jurryt Pietersma and Wei Zhang - my fellow members in the LYDIA project - for all support on the LYDIA tool set. Furthermore, I would like to thank all colleagues at Philips Medical Systems that spend time listening to me, giving me fruitful feedback, or cleared my mind with social talk. I would like to name Frank Spronck in particular, to thank him for our time in which we injected 'real' faults in a test system.

Finally, I would like to thank my family, all my friends and fellow students for being part of my life during the last years.

W.M. Lindhoud  
Best, the Netherlands  
June 27, 2006
## Contents

**Preface** iii  
**Contents** v  
**List of Figures** vii  
**List of Tables** ix  

1. **Introduction**  
   1.1 Fault Diagnosis  
   1.2 Automated Fault Diagnosis  
   1.3 Model-Based Fault Diagnosis  
   1.4 Contributions  
   1.5 Outline of the Thesis  

2. **State-of-the-Practice**  
   Fault diagnosis at PMS  
   2.1 Preliminaries  
   2.2 Overview Today's Practice  
   2.3 Manual Fault Diagnosis  
   2.4 Optimal Fault Diagnosis  

3. **Automated Fault Diagnosis**  
   3.1 Overview Techniques  
   3.2 Automated Induction using Black Box Information  
   3.3 Off-line Inference  
   3.4 Automated Induction using White Box Information  
   3.5 Model-Based Approach to Fault Diagnosis  
   3.6 Evaluation of Approaches to Fault Diagnosis  

4. **Model-Based Fault Diagnosis**  
   4.1 Fundamentals  
   4.2 Model-Based Diagnosis with LYDIA  
   4.3 Basic Assumptions  

---
CONTENTS

4.4 Modeling ................................................................. 39
4.5 Entropy Gain ......................................................... 43
4.6 MBD on the Power Supply ........................................... 45

5 Diagnosing the Beam Propeller Movement
of the Frontal Stand .................................................... 49
  5.1 Experimental Methodology ....................................... 49
  5.2 Scope of the Target System .................................... 50
  5.3 Off-line Inference by Experts ................................... 54
  5.4 MBD Implementation 1 .......................................... 57
  5.5 MBD Implementation 2 .......................................... 62
  5.6 MBD Implementation 3 .......................................... 64
  5.7 Results of the Case Study ...................................... 66

6 Conclusions and Recommendations ......................... 69
  6.1 Conclusions ....................................................... 69
  6.2 Recommendations ............................................... 70

Bibliography ............................................................ 73

A Glossary of terms .................................................... 77

B Case Study Details .................................................. 85
  B.1 Architecture ...................................................... 85
  B.2 System Data ...................................................... 86
  B.3 Behavior .......................................................... 86
  B.4 Full List of Examined Fault Scenarios ......................... 87
  B.5 Discretization of Implementation MBD-2 ..................... 89

C Lydia Models .......................................................... 91
  C.1 The Synthetic Models that are Used in this Thesis .......... 91
  C.2 Model of the Power Supply Example .......................... 94
  C.3 The Models Constructed for the Case Study .................. 100

D State-of-the-Future
  Fault Diagnosis at PMS ............................................. 117
List of Figures

1.1 The role of fault diagnosis in a dependable operation. 2
1.2 Model-Based Diagnosis 5

2.1 Philips Cardio-Vascular X-Ray System 10
2.2 Partial architecture of the power supply 11
2.3 The current procedure for doing fault diagnosis (and repair) 13

3.1 Categorization of fault diagnosis approaches 24
3.2 Power Supply - Black Box view 26
3.3 Partial architecture of the power supply (repetition of Figure 2.2) 27

4.1 3-inverter example 34
4.2 4-inverter example 44

5.1 Jobsheet referring to problems with the beam propeller movement 51
5.2 Beam Propeller Movement of the frontal stand 52
5.3 Partial architecture of the FRUs that implement the beam propeller movement. Signals enclosed by brackets are not observable, the others are 58
5.4 Modeling the control loops 60
5.5 Modeling the error signal 61

B.1 The parts of the frontal stand 85
B.2 Forces and torques on the frontal stand 87

D.1 Proposed procedure for doing fault diagnosis (and repair) at PMS 118
# List of Tables

3.1 Evaluation approaches. ................................................................. 25
3.2 Partial diagnosis look-up table of the Power Supply system for single faults (startup = true) .......................................................... 28
3.3 Input matrix for data analysis of the example fault scenario ................... 29
3.4 Evaluation approaches. ................................................................. 31

4.1 Mode catalog of the 4-inverter model, with $x=1$ and $y=0$. .................... 45
4.2 Partial LUT for a strong model of the Power Supply system for single faults (startup = true) .......................................................... 47

5.1 Example of a log entry (when error 11 occurs). .................................. 52
5.2 Components that are excluded from the target system. .......................... 53
5.3 Components that are included in the target system, with their health variables, and associated a priori failure probabilities. ...................... 54
5.4 Fault scenarios, and their adjudged diagnoses. ..................................... 55
5.5 Partial diagnosis look-up table of the Beam Propeller Movement for single faults .......................................................... 57
5.6 Discretization of the system data of fault scenario 1 till 11 to domain M. 63
5.7 Partial diagnosis Look-Up Table of the Beam Propeller Movement for single faults .......................................................... 64
5.8 Some additional measurements for fault scenarios in fault category C1 (current error) .......................................................... 65
5.9 Some additional measurements for fault scenarios in fault category C2 (speed error) .......................................................... 67
5.10 Some additional measurements for fault scenarios in fault category C3 (unjustified position error) .......................................................... 67
5.11 Some additional measurements for fault scenarios in fault category C4 (pos. val. error) .......................................................... 67
5.12 Entropy gain of 5 arbitrarily chosen additional sensor set-ups. .................. 67
5.13 5 additional fault scenarios, of which adjudged broken components are known. .......................................................... 68

B.1 The meaning of the system data that can be found in an error 11 entry. ........ 86
B.2 All fault scenarios, and their system data. ........................................ 88
Chapter 1

Introduction

Throughout the decades Philips Cardio-Vascular X-Ray System\(^1\) has gone through a lot of changes. These changes mainly pertain to more functionality and more sophisticated internal techniques, like newer techniques for X-ray generation and acquisition. Each of the components as well as the entire system has evolved over the years. There is no end to the continuous improvement, and going with that is the everlasting increasing complexity of these systems. All sorts of efforts have to be taken to cope with this complexity, so that doctors and patients can still depend on them. In other words, the systems have to be kept dependable. Dependability is a property that a successful system must have. It can be decomposed into the more lower level attributes availability, integrity, maintainability, reliability and safety [4]. The threats to a dependable operation come in the shape of faults, errors and failures and will hinder correct service delivery more and more often. The reasons for this is that managing the complexity has become a lot harder. Although the world of physical systems has never been perfect and systems are breaking as a rule, the effort put in servicing them is growing. One of these efforts is fault diagnosis. Fault diagnosis is the process of identifying the root cause(s) of a failure. A good fault diagnosis process, and/or a better design accommodating fault diagnosis, is essential for achieving high dependability. The sooner a broken component is identified, the sooner appropriate recovery actions can be taken, and users can depend on the system.

This thesis discusses an alternative approach to fault diagnosis, aimed at one of the modalities that Philips Medical Systems (PMS) develops and services, namely the Cardio-Vascular X-Ray System. Philips Medical Systems is a company that develops and services state-of-the-art medical systems. As most suppliers of embedded systems, it has to cope with the complexity crisis, caused by the shift from hardware to software and an increased amount of third party components. In order to preserve a desired level of safety and reliability, it is of prime concern that fault diagnosis is done efficiently and effectively upon the systems. The costs and effort that we devote to manually diagnosing the systems, and keeping them up and running in a safe way, are just no longer acceptable.

The next section points out the role of a fault diagnosis process in a dependable operation, presents the fault diagnosis practice at PMS of today, and shows the drawbacks of this approach. Sections 1.2 and 1.3 introduce alternative automated approaches to fault diagnosis. Section 1.4 gives the problem statement. Finally, this introducional chapter concludes with an outline of this thesis.

\(^1\)The meaning of all terms and acronyms that are printed italic are described in the Glossary of Terms, Appendix A.
1.1 Fault Diagnosis

It is practically impossible to develop a complex system that is free from faults. These faults could lead to failures; the undesirable situation that the system does not perform its intended function. Fault tolerance techniques aim at the preservation of a dependable operation in the presence of faults. Fault diagnosis is one of these techniques. Error detection and fault recovery include other techniques to achieve fault tolerance and should not be confused with fault diagnosis techniques. Error detection identifies the presence of a fault. Fault diagnosis identifies the cause of an error. Fault diagnosis is important, because when the the cause of a failure is known, it is possible to transform a faulty system state into a healthy system state (fault recovery). For example, by replacing broken components with healthy components. Preferably, fault diagnosis prevents healthy components or even the entire system from being replaced. This means error detection, fault diagnosis, and fault recovery are interrelated. Figure 1.1 shows the the role of each in achieving a dependable operation. The dashed circle in the figure delimits the main topic of this thesis: fault diagnosis. Systems consists of software, hardware and mechanical components. The focus of this thesis is on hardware. Because the hardware discussed in this thesis is part of Philips Cardio-Vascular X-Ray System, the next subsection presents today’s approach to fault diagnosis at PMS.

1.1.1 Fault Diagnosis at PMS

Philips Cardio-Vascular X-Ray System is a safety critical system. For this reason, it is a prime concern to avoid catastrophic consequences for users and the environment. That is why certain error detection mechanisms - the ones detecting errors that could result in failures harming doctors
Introduction

1.1 Fault Diagnosis

and patients - have been implemented sufficiently. However, knowing an error usually does not map directly to the malfunctioning component(s). Usually, the symptoms that identify the wrongdoer are more complex. Currently, most approaches in industry, also at PMS, that aim at the identification of broken parts are based on manual effort. The people confronted with this task, the service engineers, are supported by various artifacts. These artifacts include explaining of common fault diagnosis procedures, tests of individual or groups of components, logging, and mappings of symptoms on suspicious components. In many situations the information that a service engineer can use is not sufficient for producing an accurate diagnosis. In these cases the service engineer has no other option than to call a help desk with more expertise. If the first-line help desk also lacks sufficient information a second- and third-line help desk could be consulted. If the failure is very rare, even developers have to add their knowledge to recover the system.

Remote Monitoring

Recently, a new technique, remote monitoring, has been introduced at PMS. Among other possible uses, it could be used to improve the current fault diagnosis approach. The, previously mentioned, propagation of a diagnosis problem through the organization requires a lot of time and effort. This is most problematic when new systems are introduced. Today’s companies are forced to shorten time-to-market and optimize customer satisfaction. They cannot afford a time consuming diagnostic process. For this reason, in the beginning of 2002, a project has been started within Philips Cardio-Vascular Development [6, 29, 2]. The aim of this project was to make sure that the introduction of Philips’ most recent Cardio-Vascular X-Ray Systems went as smoothly as possible. This has led to the first systems that can be monitored remotely. A web interface presents data to experts in a readable format. The results showed a significant decrease in the propagation of problems. With the monitoring, the feedback loop between the showing of a system failure and the activation of the appropriate actions to recover the fault, has been shortened. This is because more information can be combined, and experts were able to interpret it all remotely, without traveling to the hospital where the system is located.

Drawbacks of Today’s Practice

The application of remote monitoring is very new, and the fault diagnosis practice of today still has its drawbacks. It occurs frequently that service engineers facing a system failure need help from people that developed the system. The skills and knowledge required for diagnosing the systems are increasing, and few people are able to perform the hard task. Still, a fault diagnosis practice should accurately find root causes of failures in a way that optimizes dependable operation at minimum costs and risks. The motivation for this project assumes a more optimal approach compared with the current approach. There are three key issues that could be, and should be, improved.

- Firstly, the repair time is days or even weeks.
- Secondly, too much human involvement is error-prone, and depends on the skills of currently employed experts.
- Thirdly, in case part of the system requires redesign, many tools, code and mechanisms that support fault diagnosis require re-implementation.

2The project is currently known as ServiceWAX.
1.2 Automated Fault Diagnosis

Remote monitoring improves the diagnostic performance by providing a great amount of useful data. However, experts still have to interpret this data manually. Also knowledgeable experts need to devote considerable time and effort to extract information and draw conclusions about faulty components. Automating this process is called **automated fault diagnosis**, and is discussed in the next section.

### 1.2 Automated Fault Diagnosis

Automated approaches provide many potential benefits compared to manual approaches. Automated fault diagnosis approaches automate the interpretation of raw data, in order to produce diagnoses. Such approaches are likely to be faster, less error prone, and less dependent on human intervention. Therefore, automating the fault diagnosis process can be one of the solutions towards improving dependability. Figure 1.1 shows how a diagnostic system could add to dependability. By processing data that is (remotely) monitored, the diagnostic system produces diagnoses and supports the service engineer in the diagnosing task. Then, the service engineer could focus on repair (the supervisory controller in the figure), and does not have to bother about the interpretation of log data. If the supervisory controller is also automated, a diagnosis that has been made within milliseconds could add to safety and reliability. The exact benefits depend on the specific approach and implementation of the diagnostic system. The remote monitoring technique produces data that contains information about faulty components. This is a practical environment to examine various automated approaches to fault diagnosis.

The possible automated approaches can be characterized by black box and white box approaches. A **black box** approach uses externally observable behavior, but does not state anything about the structure of the system or behavior of internal components of the system. An example of a black box approach is a data mining technique that searches for correlations between log data and faulty components. It is unable to explain why a certain correlation exists. A **white box** approach uses internally non-observable behavior as well as externally observable behavior. The behavior of the whole system is defined by the structure and the behavior of internal components. An example of a white box approach to fault diagnosis are experts that manually define a mapping between symptoms and diagnoses. The mapping is implemented by application-specific code or, more generally, by using expert systems.

Another characterization of approaches is whether the approach uses cause-to-effect or effect-to-cause reasoning. The mapping of symptoms on diagnoses, as implemented by expert systems, is effect-to-cause reasoning. The next section introduces an approach that defines cause-to-effect relations.

### 1.3 Model-Based Fault Diagnosis

**Model-Based fault Diagnosis (MBD)** is a technique for doing fault diagnosis based on a model of a system. The **model** specifies all relevant information for doing fault diagnosis. A separate tool, a so-called **diagnostic engine**, operates on this model to pinpoint root causes of failures. It is first suggested by Reiter [25] and continued by de Kleer, Mackworth and Reiter [10].

The idea behind the technique is shown in Figure 1.2. The cloud represents the real system, and its runtime operation. The model formally defines nominal, and possibly faulty behavior, of the system. The diagnostic engine uses this formal model for computing predictions of system behavior. During system operation, live data is gathered and processed in order to obtain actual
observed system behavior. At certain moments in time, the diagnostic engine compares prediction and reality to enable the search for the root cause of a failure. This way, two different knowledge domains are separated, and can be developed independently: domain knowledge of the system (the model) and knowledge about search algorithms (the diagnostic engine).

The diagnostic engine should be as general as possible. In other words, given a formal model of an arbitrary system, the diagnostic system should be able to produce diagnoses. There are several diagnostic systems available. The work in this thesis uses the LYDIA approach to MBD [1, 27]. LYDIA (Language for sYstem DIagnosis) is a new language for specifying the model. There is ongoing research in an accompanying tool set that includes a diagnostic engine. Currently, the Lydia approach is successfully applied in industry at ASML, a company developing lithography systems for the semi-conductor industry [22, 23, 5].

The diagnostic engine of LYDIA has already been constructed, and is used in the work done for this thesis. In a model-based approach, the only effort is the construction of the model. However, modeling is not a trivial activity, and there are various lessons that can be learned about modeling, modeling in LYDIA in particular, and using the LYDIA diagnostic engine.

## 1.4 Contributions

The previous sections lead to the hypothesis that automated fault diagnosis, and in particular the model-based approach, can improve fault diagnosis at PMS. Improvement of fault diagnosis could enable higher dependability of Philips Cardio-Vascular X-Ray Systems, lower costs to develop diagnostic tools, lower cost to diagnose a failure, or be less dependent on employees that are skilled to diagnose the systems. In other words, the alternative approach should have a higher diagnostic performance.
1.5 Outline of the Thesis

This work explores the possible benefits of automated fault diagnosis of the most recent Cardio-Vascular X-Ray Systems at Philips Medical Systems. In particular, the problem that is encountered in the work for this thesis is to improve fault diagnosis of Philips Cardio-Vascular X-Ray Systems with respect to higher dependability. An improved fault diagnosis approach is able to increase dependability of the system, because timely and accurate identification of root causes of failures enables system recovery. Fault diagnosis is only successful when produced diagnoses agree with reality, so the result could be used to recover the system. Accuracy is the extent to which diagnoses agree with reality, and is considered to be the most important criterion for diagnostic performance.

This thesis presents a proof-of-concept of the model-based approach to fault diagnosis, aimed at Philips Cardio-Vascular X-Ray Systems. In order to achieve this, the aim of the project is to

1. uncover the drawbacks of today’s practice to fault diagnosis.
2. show which automated techniques to fault diagnosis could possibly increase diagnostic performance.
3. motivate the choice for the model-based approach by criteria for diagnostic performance.
4. apply the model-based approach to an example system. This is a subsystem of the Philips Cardio-Vascular X-Ray System. This case study elicits modeling issues that typically occur when MBD is applied in the industrial domain.
5. study the applicability of LYDIA to model specific dynamics, as well as the use of the accompanying tool set in a real-life scenario.

The modeling issues that are discussed in the case study include a discussion of an approach to modeling, the use of entropy to estimate the diagnostic performance of a model, and the use of entropy to determine next best measurements. Beyond the scope of this thesis is to deal with multiple and dependent faults.

In this thesis it is shown that the current practice at PMS is not optimal with respect to criteria for diagnostic performance, and that model-based diagnosis is likely to improve the fault diagnosis process at PMS. However, it is not possible to conclude that MBD improves fault diagnosis with respect to higher dependability of Philips Cardio-Vascular X-Ray Systems. The experimental methodology that should be used to establish this conclusion requires that the diagnostic accuracy can be determined for both the current practice as the proposed MBD approach. Despite the fact that it is possible at PMS to know what observations caused a failure, it is impossible to determine the adjudged diagnosis that explains these observations. Consequently, it is not possible to conclude that MBD improves fault diagnosis with respect to higher dependability of Philips Cardio-Vascular X-Ray Systems (see Chapter 6).

It is shown that entropy is a valuable metric to show the improvement of different MBD implementations. This way, it is established that an MBD approach is able to achieve the same accuracy and entropy as fault diagnosis applied by experts. Furthermore, entropy is also able to suggest various alternative MBD implementations that lead to higher accuracy.

1.5 Outline of the Thesis

The outline of the thesis is as follows. In the next chapter, the current approach at PMS to fault diagnosis is reviewed and analyzed by means of an example. The final section of Chapter 2 presents
Introduction

1.5 Outline of the Thesis

criteria to estimate diagnostic performance of approaches to fault diagnoses. This enables an evalua-
tion of the current approach, and a precise definition of the problem. This problem is solved in the
remainder of this thesis. Chapter 3 presents possible solutions to the problem. It presents automated
approaches to fault diagnosis, and uses the criteria of Chapter 2, as a motivation for developing
a model-based approach. Then, Chapter 4 introduces the theory of model-based fault diagnosis.
Chapter 5 elaborates the technique on a subsystem of the Philips Cardio-Vascular X-Ray System,
the so-called beam propeller movement of the frontal stand. This chapter presents MBD as a so-
lution that improves fault diagnosis with respect to higher dependability. Chapter 6 concludes this
thesis with a discussion on the solution, conclusions and recommendations.
Chapter 2

State-of-the-Practice
Fault diagnosis at PMS

This chapter describes the current approach to fault diagnosis, that Philips nowadays applies on their Cardio-Vascular X-Ray Systems. This description is given to show why a new approach could improve the fault diagnosis process, and what items of the current fault diagnosis process could be improved. The outline of this chapter is as follows. The first section introduces preliminaries. The second section gives an overview of today’s means and procedures within PMS for doing fault diagnosis. The third section provides a more concrete understanding of the current practice by applying it on a real-life example. This real-life example is a subsystem of the Philips Cardio-Vascular X-Ray System. The final section of this chapter shows why the current approach is suboptimal. It does by introducing items that make a fault diagnosis technique good, and use them as criteria for estimating the diagnostic performance of the current approach.

2.1 Preliminaries

This section introduces preliminaries. Section 2.1.1 introduces the system that is subject to diagnosis; the Philips Cardio-Vascular X-Ray System. Section 2.1.2 introduces the real-life example that is used in Section 2.3 to give a concrete example of today’s approach to fault diagnosis at PMS. This example is also used in Chapter 3 and Chapter 4 to give concrete examples of alternative approaches to fault diagnosis. Section 2.1.3 introduces terminology and concepts that are required to discuss issues in the remainder of this thesis.

2.1.1 Philips Cardio-Vascular X-Ray System

The Cardio-Vascular (C/V) X-Ray System is one of the modalities developed and serviced by Philips Medical Systems. The system is used to enable diagnosis and treatment of patients with cardiac and vascular diseases. Figure 2.1 shows a picture of such a system. In short it works as follows: the patient lies on the table. One or (in the picture) two stands are positioned around the patient in order to capture images of the body. To enable the capturing, two devices have been assembled on the far ends of the stands. The first, a collimator, is used to limit and aim the radiation beam. The second, the so-called flat detector, is used to capture the X-rays. Then, the signals are processed to digital output that can be shown on the monitors. The doctors use the shown information to diagnose or operate a patient.
2.1 Preliminaries

2.1.2 Introduction of the Power Supply Example

This section introduces an example that is used throughout this thesis for clarifying fault diagnosis related concepts and ideas. The Cardio-Vascular X-Ray System is a complex system that has dozens of components. Consequently, the power supply of all electrical components has quite some complexity. Figure 2.2 partially shows the architecture of the system in respect to the power supply. The idea is to keep it simple, and therefore it is not complete in its depicted components. On the left, the most important power supply of the Cardio-Vascular X-Ray System, the so-called Power Distribution Unit, is shown. Its function is to switch on the various subsystems by supplying voltage (230V). The components on the right (Flat Detector, TBCB, CRCB, Collimator, Chiller) are all components that need voltage. Their behavior is beyond the scope of this thesis. The function of the component in between, the chameleon, is to provide part of the components with low voltages (24V).

From now on, this example is referred to as the power supply example. The power supply example is used in Chapter 3 for introducing various automated approaches to fault diagnosis. Chapter 4 uses it to show a realistic example of model-based diagnosis applied to the Philips Cardio-Vascular X-Ray System. The following preliminary section also uses the power supply example in order to introduce some fault diagnosis related concepts.

2.1.3 Faults, Errors and Failures

The previous section introduced a subsystem of the Philips Cardio-Vascular X-Ray System. Notice that the term subsystem refers to a subset of interrelated components within the X-ray system, and not to the subsystem decomposition as defined in the system design document [16] of the X-Ray System. This section uses power supply example to introduce notions related to fault diagnosis. Fault diagnosis is a mean to deal with the threats to a dependable operation: faults, errors and failures. It is important that the reader fully understands these concepts, because these are part of
the environment of any fault diagnosis technique. Consider the power supply example. The function of the power supply subsystem is: if the power distribution unit, all cables, all fuses and components themselves are "healthy", the Flat Detector, TBCB, CRCB, Collimator and Chiller are on if the system is switched on. This healthy case is referred to as the intended, or nominal behavior. If something in the system is broken the system is "unhealthy", and shows other behavior than the nominal behavior. This behavior is unexpected and preceded by a transition of correct functioning to malfunctioning.

Such a transition is called a failure. A failure is defined as an event that occurs when the delivered service deviates from correct service [4]. Due to a failure the operator is unable to perform certain functions of the system. For example, the operator is unable to generate images of a patient’s vasculars. The reason for this is that after a failure the external system state of the system deviates from the correct system state. Here, it is assumed that the deviations are part of the state of the power supply example, as shown in Figure 2.2. Suppose the deviation from the correct system state is that TBCB, CRCB and Collimator are off, while they should be on. This is called the error. An error is defined as the deviation between external system state and correct system state [4]. In fault diagnosis, it is common to use the word symptom instead of error. The terms symptom and error are interchangeable. The cause of error is called a fault. An example of a fault that could cause an error in the power supply example is the malfunctioning of CableB. A fault is defined as the adjudged or hypothesized cause of an error [4]. In respect to the result of a fault diagnosis process, a fault is called a diagnosis. Like the terms error and symptom, the terms diagnosis and fault are interchangeable. There are many classes and types of faults (see [4]). A fault could refer to the malfunctioning of a component, but it could just as well refer to particular causes why a component malfunctions. However, in a company that uses many third party components in its systems, it is likely that these low-level details are unknown. Therefore, fault diagnosis at PMS aims to identify
faulty components. The situation in which one component is adjudged or hypothesized to be broken is called a *single fault*. If more components are adjudged or hypothesized to be broken, there are *multiple faults*. These concepts are important, because taking into account multiple faults greatly affects the number of possible causes for a particular error; it is exponential.

A *fault scenario* is a specific occurrence of a fault. In this thesis, *fault scenario* is defined as a tuple that consists of a fault and an error. An example of a fault scenario is the malfunctioning of CableB and its manifestation in the error; TBCB, CRCB and Collimator are off, while they should be on. Further decomposing the "CableB subsystem" could uncover faults on a lower level of detail. For example, CableB is not connected to the Power Distribution Unit. Or Cable does not conduct current. However, the fault diagnosis techniques described in this thesis aim to identify broken components that could be replaced, the so-called *FRUs*. So, the statement "CableB is broken" suffices. A multiple fault diagnosis of the error is that the FRUs LV_PS2 and FuseD are broken. An important issue to realize is that not all faults cause an error, and not all errors cause a failure. Faults can be active (cause an error) or dormant (not causing an error). Also, internal errors could be dealt with by other subsystems. Fault diagnosis could aim at the identification of all kinds of faults. However, the work in this thesis aims at the identification of active faults that result in a failure.

In the example fault scenario described above, the diagnostic process should identify CableB as a suspicious component. Section 2.3 shows how the current diagnostic process at PMS identifies CableB. The next chapter shows how automated approaches diagnose CableB as a broken component. The next chapter also extends the fault diagnosis related concepts and terminology that is introduced in this section. The remainder of this chapter uses the terms that are explained above for describing and evaluating the current approach to fault diagnosis at PMS.

### 2.2 Overview Today’s Practice

The need for fault diagnosis starts when the operator experiences a failure of the system. This section has been divided into two parts. First, the procedure that starts when a failure occurs is discussed. Then, the second part introduces the available means that service engineers and other *troubleshooters* have.

#### 2.2.1 Current Procedure to Fault Diagnosis

The fault diagnosis process of today has a procedure that is quite complex. Consider Figure 2.3. This figure clarifies all actors that play a role, and all actions that are performed when a failure occurs. Normally, the operator uses the system without a problem. When difficulties are encountered then he/she calls the local help desk for help. If the problems in using the system can not be solved by giving instructions for use, a service engineer is sent to the hospital in order to diagnose and possibly repair the system. In customary fault scenarios this is sufficient. However, the service engineer has not been educated to solve all problems. In many cases, he is puzzled and has to call the local help desk for help. This way, it is possible that a problem has to be propagated to the more expertized regional and global help desks. Eventually, people that developed the system are called to help the service engineer or solve the problem themselves. Figure 2.3 shows which activities must be done for diagnosing a system, when an operator faces a failure. *Service Innovation* is a department at PMS Cardio-Vascular that is responsible to provide service engineers with information that is necessary for diagnosing Philips Cardio-Vascular X-Ray Systems. Developers of the system know much of this information. Therefore, employees at the Service Innovation department interview developers in order to write down knowledge that can be used for diagnosing. The legend of Figure
2.3 refers to these efforts of Service Innovation as preparatory activities. The goal of the Service Innovation department is trying to prevent as many problems as possible from propagating to the C/V development department. If a diagnosis is adjudged, it is used to recover the system. At PMS this means that the malfunctioning components are replaced with healthy components. This requires that substitutable FRUs are provided as soon as possible.

2.2.2 Current Means to Fault Diagnosis

The list below is an overview of all means that a troubleshooter has in order to diagnose and/or repair a malfunctioning system. The order gives an indication of the importance of the specific means for the service engineer. Especially the last two items are rarely used in practice.

**Interviewing the operator about the problem** Best case, the interview already pinpoints the search for the malfunctioning FRU to a certain subsystem or collection of FRUs. It is also possible that he finds out that the failure might be caused by an incident or abnormal use of the system.

**Visual checking the system** The service engineer has access to the system, so he is able to look for unexpectedly flashing LEDs (that indicate the status of some FRUs), loose cables or any mechanical damage.

**Checking the log** A log is a digital file that consists of data that a system produces during real-time operation. This data contains information about past system behavior (e.g.,: the occurrence of events, error messages, warnings, values of parameters). They all could pinpoint to a

---

1In this section we use the term *troubleshooter*, instead of *service engineer*, to emphasize the fact that these means are available to all troubleshooters.
2.3 Manual Fault Diagnosis

Particular subsystem. However, the service engineer has to understand this information, in order to be able to use it for diagnosing a system.

Performing Power-On-Self-Tests (POSTs) For some FRUs a so-called POST exists. A POST is a test of one FRU that is automatically performed at startup. It can either fail or succeed. A failure indicates that the FRU is broken or not correctly connected.

Performing Build-In-Self-Tests (BISTs) For some FRUs a so-called BIST exists. A BIST is a test of one or more FRUs that can be performed in Field Service mode. Just like a POST it can either fail or succeed. A failure indicates that something is wrong with the associated FRUs and their interconnections.

Searching for Symptom-Cause-and-Solution sheets For well-known problems the Service Innovation department has made a bundle of quick lookup sheets, each containing symptoms, and their corresponding cause and solution.

Searching the Error-on-Solution database If a service engineer finds a certain error message in the log file, he can search a database for the corresponding cause and solution. Again, only well-known problems are included.

Using Technical Drawings To enable the checking of cable- and FRU-connections, Service Innovation has made a collection of technical drawings that define the structure of the system at various levels of detail.

Using Fault Isolation Procedures (FIPs) These are tree-like graphs that define a sequence of steps and tests in order to repair a malfunctioning part of the system. It is an artifact made by Service Innovation.

Reviewing job sheets All essential action that troubleshooters perform must be documented in a so-called job sheet. If a problem can not be eliminated, it is useful to examine this history of diagnosis and repair. Unfortunately, these job sheets are ambiguous in what components are diagnosed to be broken, and which repair actions are taken.

Checking trace files Tracing contains, just like logging, information about the past system behavior. The difference is that tracing contains much more detail. Only (special) experts of the system can take advantage of it in their fault diagnosis (when this happens the problem will most likely be solved by changing the design of the system).

Symptom-Cause-and-Solution sheets, Error-on-solution database, Technical Drawings and FIPs are all artifacts made by Service Innovation, and illustrated in Figure 2.3 by the books in the actor’s hands.

2.3 Manual Fault Diagnosis

This section describes a concrete example of the current approach to fault diagnosis, when a failure occurs in the power supply example. The failure is the - earlier discussed - example fault scenario in which CableB is broken. As Section 2.2 described, a service engineer has to visit the hospital.

\footnote{Field service mode is a special use of the system, especially made for diagnosing it. Using the system in this mode enables the viewing of parameters and the performing of tests.}
where a system is located, and start the search for the malfunctioning FRU. Some service engineers are familiar with checking the log, and start their search there. If the service engineer recognizes the appropriate messages, the TBCB, CRCB and the Collimator are the starting point of the search. From experience the service engineer knows the fuses of these components are the first suspects. So, these are the first components that are checked. Then the TBCB, CRCB and Collimator themselves are examined. This can be done, because of various status LEDs, that give an indication of their health. If none of these seems to be the cause, all other components are sequentially checked (looking at the LEDs, cabling, etc.) for inconsistencies. Eventually, the service engineer will find out that CableB is not connected well or is broken. The latter can be detected by measurement.

2.3.1 Drawbacks

The described process above shows that finding a simple broken cable takes much time. In the real case, there are even more cables, fuses and components. All of them must be checked, and this is time consuming. Interviews with experts and service engineers show that it takes approximately one hour to diagnose the system. If one of the more complex components (the components depicted on the right of Figure 2.2) are broken, and LEDs do not indicate malfunctioning, the identification of these as the wrongdoers takes much more time. In these situations, developers of the power supply example need to help, because they know more about the system.

2.4 Optimal Fault Diagnosis

This section defines and clarifies the problem that this thesis addresses precisely. The former described the current approach to fault diagnosis at PMS. A better diagnostic approach can only be suggested if it is known which items make a fault diagnosis process good. If these items are known, it is possible to evaluate the current approach, and evaluate alternative approaches. Section 2.4.1 introduces the items that an ideal fault diagnosis would have. Section 2.4.2 presents the evaluation using the items of an ideal fault diagnosis process as criteria. The final subsection shows what items could possibly be introduced, or improved, in a new approach to fault diagnosis in order to achieve higher dependability.

2.4.1 Ideal Fault Diagnosis

The ideal approach to fault diagnosis utilizes as much information as possible in the search for root causes of failures, in a way that optimizes dependable operation at minimum costs and risks. This applies to PMS as well as to all companies constructing embedded systems. In order to address this ideal approach as much as possible, items are identified that make a fault diagnosis approach 'perfect'. These items can be used to evaluate diagnostic approaches. Dash and Venkatasubramanian list some characteristics that a diagnostic system should ideally posses [9]. With this list as a starting point, and by interviews with PMS employees, it is derived that the following item are present in an ideal approach.

1. **Accuracy.** Accuracy is the extent to which a diagnosis produced by the diagnostic process agrees with reality. If a diagnosis is not accurate (inaccurate) there can be two situations:
2.4 Optimal Fault Diagnosis

- **false alarm:** If a component is diagnosed to be faulty while it is healthy, costs increase. At least, if you assume that the diagnosis is being used to bring the system in a better state \(^3\).
- **missed diagnosis:** If a component is diagnosed healthy while it is not, a failure is likely to reoccur and degrade the dependability.

2. **Speed of diagnosis.** The sooner an error and its fault is identified, the less impact a failure has. A fast diagnostic process increases the availability of the system. In other words, the **customer downtime** will be minimized.

3. **Low Uncertainty.** Also known as **diagnostic resolution** or **isolability.** This is the extent to which a diagnostic process is able to minimize the set of suspicious components. It can be measured by using entropy. The more a root cause of failure is isolated, the less repair time is needed. Consequently, costs decrease and the availability of the system increases.

4. **Context independency.** If the successful working of the diagnostic process highly depends on forces outside the sphere of influence of the company, obviously risks increase (e.g., dependence on employees, network of others, etc.).

5. **Low development costs.** Development costs are all the costs that have to made prior to the start of the diagnostic process. These are the costs for developing all the diagnostic tools (e.g., supporting artifacts, training sessions for troubleshooters, other prepare actions that precede the operational phase).

6. **Low runtime costs.** Runtime time costs are all the costs that the company makes to keep the diagnostic process up and running.

7. **Explanation facility.** The justification of a diagnosis helps in evaluating diagnostic decisions. It increases the trust in a dependable operation of the system. Also, the quality of the system can be evaluated and improved.

8. **Adaptability.** Design changes occur frequently. There is always need for more/other functionality and better internal working. Therefore, the ability of a diagnostic process to cope with design changes is a prime concern. A failing in this ability results in a decrease of all other attributes that are listed in this list.

The following items would add to perfectness, but are outside the scope of this thesis:

- **Robustness.** This refers to the extent to which a diagnostic process can handle unexpected situations.

- **Novelty Identifiability.** The ability to detect and diagnose faults that did not occur before.

- **Ability to deal with multiple faults.**

- **Reasonable storage and computational requirement.**

Although these 4 items are important attributes in an ideal approach to fault diagnosis, the work that this thesis presents does not discuss these items. However, the technique presented in Chapter 4 could deal with these items, but in the case study of 5 the diagnostic performance in respect to robustness, novel identifiability, multiple faults, and storage and computation requirement is not examined.

\(^3\) At PMS this means that a component that is diagnosed as unhealthy will be replaced.
2.4.2 Evaluation Current Approach

Throughout the years the diagnostic performance steadily became suboptimal. When the first Cardio-Vascular systems were placed in hospitals, the systems could easily be diagnosed and repaired manually. There was only one power supply, a couple of cables, and the more complex components could be manually measured quite easily. Almost all functionality was implemented by hardware. Therefore, analog measuring was the most efficient way for diagnosing these systems. The skills and knowledge of service engineers perfectly fitted the diagnostic tasks. However, ever since, various trends have changed the situation:

1. Increased complexity, caused by evolved techniques and additional functionality.
2. A shift from a hardware-centric to software-centric embedded system.
3. An increased number of third party components.

These trends have decreased the presence of the items that an ideal approach should have. These items are used as criteria to evaluate the current approach, as follows:

- **Accuracy.** The current process and available tools do not allow the determination of how accurate the diagnosis is. For this, it should be known if a replacement of the diagnosed FRU recovered the failure, and this information is not available. The only way to estimate the accuracy, is to examine job sheets for reoccurrence of problems, or to interview troubleshooters for their experiences. Both sources make it plausible that today’s diagnostic process is not very accurate, unless failures are known and very well understood. Unfortunately, it is impossible to quantify this attribute with the current techniques.

- **Speed of diagnosis.** The speed of diagnosis can be measured by recording the time between the moment that a failure occurs and the moment that a troubleshooter isolates the root cause of that failure. In the current situation it can be recorded per failure, by interviewing operators and examining job sheets. The period is several days in case of a mainstream problem. Otherwise the problem has to be escalated through one or more of the help desks (recall Figure 2.3) and could take weeks, if not months.

- **Low Uncertainty.** The uncertainty of today’s diagnoses is, like the accuracy, hard to determine. Again, only job sheets and interviews can give some insights. These indicate that the certainty of service engineer is disputable in many cases. However, it is not possible to quantify this because the job sheets do not provide a list of possible diagnoses.

- **Context Independency.** Section 2.3.1 described the actions that employees perform for solving a failure of the power supply. It shows that, nowadays, many diagnostic knowledge depends on current employees. Therefore, reliance on many people is seen as a drawback of the current approach. It indicates strong environmental influences on the diagnostic process; if people change jobs, the diagnostic capabilities within the organization degrades. Consequently, the current practice is not very independent of its environment.

- **Development Costs.** The development costs of a diagnostic process that aims at maintaining complex systems, such as the Philips Cardio-Vascular X-Ray System, are expected to be very high, and so they are. So, any qualification is always relative to other approaches. The exact development costs are not examined, but it is known that the development costs typically outweigh the runtime costs.
2.4 Optimal Fault Diagnosis

- **Runtime Costs.** The runtime costs of the current diagnostic approach are high compared to the automated approaches suggested in the next chapter. Many people are involved and that means high labor costs. Also, the fixed costs of having the help desks up and running.

- **Explanation Facility.** Nowadays, the ability to explain the causes of a failure only consists of the job sheets. However, usually these do not contain enough detail to understand a specific failure.

- **Adaptability.** In case of a design change, all the artifacts that support the service engineer, and are related to the particular subsystem, require re-implementation. This includes fault isolation procedures, technical drawings, Symptom-Cause-and-Solution sheets and the Error-on-solution database. This means that the Adaptability is not well addressed in the current approach.

2.4.3 The Goals of a New Approach

The above shows that not all attributes are present in today’s practice. Therefore, it can be concluded that the current approach is not able to meet the definition of a perfect fault diagnosis process. The most striking flaws are speed of diagnosis, inaccuracy, uncertainty and inflexibility. The main cause of the inability to achieve a desired presence of these items is that information is not at the right time, at the right place, in a suitable form for diagnosing. Another flaw that is caused by a lack of information is the reason that accuracy of the current practice cannot be determined.

Employees that develop the system, as well as service engineers, all have valuable knowledge that is important for the diagnostic process. The artifacts described in Section 2.2 aim to record this knowledge. However, these are not able to cope with the increased complexity. It is more and more important to use data from the log. The interpretation of this data, that is specific for each particular version of the system, requires information that is hard to record. Service engineer are supposed to interpret this data, but it cannot be expected that employees have enough information in order to use their own, time consuming and error-prone, human interpretation, for each produced diagnosis. For these reasons, this lack of information, makes that the current approach is not fast, accurate and flexible.

The new proposed approach should make better use of information in the organization. An automated approach is a mean to achieve this. This automated approach should be able to improve accuracy, speed of diagnosis, decrease uncertainty, and offer more flexibility in case of a design change. An approach that allows for validation of the presence of these items is in favor above other. This way, it is possible to improve fault diagnosis with respect to higher dependability. The next chapter shows the possible automated approaches. The items of Section 2.4.1 are used as criteria for the rationale to choose the most optimal approach.
Chapter 3

Automated Fault Diagnosis

In the previous two chapters, we have seen that the fault diagnosis approach, currently used, has attributes that can be improved. The complexity of today’s systems hinders an effective manual search for the root cause of failure. The shift to software not only adds to the complexity of the system itself, but also expands the observability of that same system\(^1\). So, there is a lot more data that must be interpreted. Consequently, the knowledge of the system that most troubleshooters have is not enough for interpreting those large amounts of data. Computing is not that restricted to problem size as humans are, and therefore automated solutions are likely to deal with the increased complexity. In this chapter a rational for choosing a particular automated approach is established.

At the end of this chapter, a technique is chosen that is used in the new approach to fault diagnosis. In order to motivate this decision, this chapter gives an overview of approaches to fault diagnosis, and briefly discusses some example implementations. The first section introduces a logical overview of the categories that a fault diagnosis approach can be in. The sections that follow present example implementations, and use the power supply example to show how each implementation produces a diagnosis. This way, the reader is given understanding in the drawbacks and advantages of each approach. This understanding is useful for judging the evaluation of these approaches, that is given in the final section of this chapter. This evaluation is done according to the criteria of Section 2.4.1, and is used to motivate the choice for a specific implementation.

3.1 Overview Techniques

In this section the reader is given an overview of the important concepts that play a role in fault diagnosis techniques, and uses them to categorize approaches to the diagnosis problem. The current approach to fault diagnosis at PMS, as discussed in the previous chapter, can be placed in the categorization that is presented in this section. Also, the examples of automated implementations that are presented in subsequent sections can be characterized by this categorization.

Fault diagnosis process is a means to improve the dependability of a system by making it fault tolerant. That is, the system should remain operational in the presence of faults. There are quite a few automated fault diagnosis techniques. The only scientific attempt trying to provide an overview is done by Dash and Venkatasubramanian [9]. Implicitly or explicitly, in all techniques the topics logic, complexity theory and system theory play a role. Logic because diagnosing is a reasoning task. Complexity theory because reasoning is time/space complex. System theory because diagnosing is

\(^1\)Note that an automated approach does not exclude the use of the observations that can still - and only - be obtained manually (e.g., a LED that blinks)
about explaining the (abnormal) behavior of a system. Logic, complexity theory, and system theory provide some concepts that are important to explain. This explanation of fault diagnosis related concepts allows the reader to understand the categorization of fault diagnosis approaches, that is given at the end of this section.

### 3.1 Overview Techniques

#### Automated Fault Diagnosis

3.1.1 System, Model and Target System

First of all, the concepts *system* and *model* require some explanation. After all, fault diagnosis is about explaining the behavior of *systems*. A *system* is an entity that interacts with other entities, i.e., other systems, including hardware, software, humans, and the physical world with its natural phenomena [4]. Examples of systems are the Philips Cardio-Vascular X-Ray System, the power supply example, and the electrical grid of the hospital. Systems are complex entities that possess much detail. Therefore, a *model* is used to abstract from these details, and allow to reason about the *system*. There are numerous possible models for one particular system. This is because a *model* is an abstract representation of a system from a particular viewpoint, chosen by the modeler. In the interpretation of model that used in this thesis, a model can be both written down on some medium, or could remain only in the knowledge of an individual. Usually, a person or entity that diagnoses a system focuses on part of the system, in order not to be distracted by irrelevant parts of the system that do not affect the diagnosis. The part of the system that is focused upon is called the *target system*, and is defined as the set of components that is subject to a particular fault diagnosis process. The model only specifies information about this target system, and other entities are considered as the environment of the system.

3.1.2 Observability of the Target System

In fault diagnosis, it is necessary to examine the system state of a target system. *Observability* of a system is the extent to which internal states of a system can be inferred by knowledge of its external outputs. On one extreme no diagnoses can be derived from observing the system. In this case a fault diagnosis approach could only reason about the hypothetical state of the system. In an ideal situation, on the other hand, observing the system yields no uncertainty about the correct diagnosis. A property of the system that can be observed is called an *observable*. These variables can somehow be measured by an external entity. Other variables that play a role in the model of a system are impossible to measure. Usually, making additional observations of observables, and use them for deriving diagnoses, decreases the number of possible explanations. Therefore, better observability of the target system increases diagnostic performance. The way of making an observation can be done by some automated entity and/or by humans. This distinction is important, because some variables are only observable by applying manual effort, while others can also be observed by an automated entity. Automated fault diagnosis approaches most practically use observations that are made automatically, although it is possible that manually made observations are inserted to an automated process.

3.1.3 Black Box versus White Box Models

This viewpoint of the modeler determines what properties of a system are described by the model. The concepts *structure* and *behavior* are important in fault diagnosis. These are also the two properties that have a central role in system specification languages, such as VHDL [3] and LYDIA [27]. The *behavior* of the system is what the system does to implement its function, and is described
by a sequence of states [4]. The structure is what enables the system to generate the behavior [4]. One of the ideas in system theory is that the structure is as important for implementing the function of a system as the individual behavior of components. The reason for this is that the interaction between individual components results in the intended behavior of the whole. These relations between structure and behavior are important to understand, because when creating an approach to fault diagnosis, various viewpoints on these concepts can be used. Two very important viewpoint types are black box and white box. The definitions of these concepts that are used in this thesis are:

**Black Box:** A model that describes the externally observable behavior, but does not state anything about the structure of the system, or behavior of internal components of the system.

**White Box:** A model that describes the internally non-observable behavior as well as externally observable behavior. The behavior of the whole system is defined by the structure and the behavior of internal components.

Many views on a system include more information than just system input and output, but do not provide a complete behavioral, or structural description of the system. These views are sometimes referred to as grey boxes, but terms like these are not well established. The problem with these terms is that it is not clear what is considered to be a ‘complete’ white box, because a model is never complete. For this reason, in this thesis a model is called white box whenever it uses some structural or internal behavioral information. Otherwise, it is a black box model.

### 3.1.4 Abductive Model versus Consistency-based Models

The distinction between black box and white box models determines what information is used in the model. Another distinction determines how the information is stated. A distinction between a consistency-based and an abductive approach to fault diagnosis [18], is as follows:

**Abductive model:** A model that defines a diagnosis as a set of abnormality assumptions that covers (or, in terms of logic, implies) the observations. [18, 24]

**Consistency-based model:** A model that defines a diagnosis as a set of assumptions about a system component’s abnormal behavior such that observations of one component’s misbehavior are consistent with the assumption that all the other components are acting correctly [11, 25].

The important difference between the two is that the former, an abductive model, defines effect-to-cause relations between observables, while that latter defines cause-to-effect relations between observables (also called first principles). This requires different kinds of reasoning schemes.

### 3.1.5 Effect-to-Cause versus Cause-to-Effect Reasoning

In effect-to-cause reasoning, the effect "there is no light" can be explained by the cause "the light bulb is broken", knowing that that when a light bulb is broken there is no light. In cause-to-effect reasoning, the fact "the light bulb is broken" is the only cause that is consistent with the effect "there is no light". Let s be a symptom and f be a fault (e.g., the fault f denotes the cause "the light bulb is broken", and the symptom s denotes the effect "there is no light"). In logic, the basic inference rule of abductive reasoning can be characterized by the following reasoning pattern [8]:

$$
\begin{align*}
  s \\
  f \rightarrow s \\
  \vdash f
\end{align*}
$$

(3.1)
This rule is unsound with respect to deduction, because of the fact that \( f \) implies \( s \) does not mean that the \( s \) implies \( f \). In other words, although the consequent \( s \) of the sentence is affirmed, the antecedent \( f \) can still be false and true. The abductive approach to fault diagnosis requires information about faulty behavior and an inference method that generates explanations for observed symptoms. In this inference method, it is possible that the premises "symptom \( s \) is observed" and "a fault \( f \) implies symptom \( s \)" can be explained by the conclusion that fault \( f \) has occurred. The inference method for an abductive model is abduction, while a consistency-based model can be solved by deduction. In the discussion of MBD (Chap. 4) the abductive sentences of the form \( f \to s \) can be added to the consistency-based model, and allow the deductive inference mechanism to derive diagnoses with less uncertainty (called a strong model). The relations between abductive and deductive reasoning is described in [8]. A consistency-based model that describes the relations between a fault and a symptom is:

\[
\neg f \rightarrow \neg s \\
\neg s \\
\vdash f
\]

(3.2)

This rule is sound from a deductive point of view. Recall the definition of a consistency-based model that is given earlier. Suppose the fault denotes that a component is broken (e.g., "the light bulb is broken"). The first formula of 3.2 defines the set of assumptions about a component's abnormal behavior, by saying that if the system is healthy, there is no symptom of a failure. The symptom \( s \) is the observation of the component's misbehavior. These two facts are only consistent if there is a fault \( f \).

### 3.1.6 Online versus Off-line inference

The inference of a model and observations to diagnoses can be done real-time with real-life observations. The inference is applied each time the system is diagnosed. This is called online inference. It is also possible to do much of the inference beforehand, in the development phase of the diagnostic process. In this case a so-called Look-Up Table (LUT) is created. A LUT is a table that defines a mapping between symptoms and consistent diagnoses. During the operational phase, real-life observations are searched in this LUT to come up with diagnoses. This is called off-line inference. A LUT can be implemented in various ways. Manual approaches include implementations of the LUT like the Symptom-Cause-and-Solution sheets, Error-on-solution database, and - although very differently represented - FIPs that are used at PMS, as described in Chapter 2. Effect-to-cause reasoning always use an implementation of the LUT to store the symptom-on-diagnosis mapping. A LUT and its look-up mechanism can be automated by application-specific code or, more generally, by using expert systems or diagnostic reasoning.

### 3.1.7 Categorization of Fault Diagnosis

The previous subsections introduced various characteristics that fault diagnosis approaches can have. In this subsection these characteristics are used to categorize fault diagnosis approaches. Figure 3.1 shows a categorization of fault diagnosis on the direction of reasoning (left: effect-to-cause, right: cause-to-effect) and the kind of information that is used (top: white box, bottom: black box). All approaches in this spectrum can be applied by a manual approach or automated implementation. In each quadrant, there are numerous possible approaches that differ in implementation and/or reasoning scheme:
1. The first quadrant depicted in Figure 3.1 shows approaches that use a consistency-based model for deriving diagnoses. A manual method includes a human that knows the consistency-based model by head, and applies deductive reasoning for each failure that occurred, to come up with a diagnosis. This is one of the ways service engineers at PMS, and expert at PMS, diagnose systems. In an automated approach the consistency-based model is formalized, and the deductive inference of diagnoses is automated by a solver. This technique is called Model-Based Diagnosis, and is introduced in the following chapter. If the consistency-based model only specifies structural information, it is not possible to deductively derive anything. In this case, it is possible to use induction when pass/failed-outcomes of output observables are available. This method is explained in Section 3.2.

2. The second quadrant depicted in Figure 3.1 shows approaches that use an abductive model. A manual method includes a human that has abductive knowledge about the system, combines this with real-life observations, and applies abductive reasoning to come up with diagnoses (this is also done by service engineers and experts at PMS). In an automated method the inference is done off-line in order to construct a LUT. One might say that service engineers, or an automated entity, that is given a LUT only use black box information. This is true, but these approaches are still categorized as white box because the information for constructing this LUT is white box information. Furthermore, it is important to notice is that the LUT can also be constructed using a consistency-based model, as delimited by the arrow between verb—deduction—and LUT.

3. The third quadrant depicted in Figure 3.1 shows approaches that only use real-life observations of input and output values, and a history of diagnoses (black box information combined with diagnostic experience). For example, by diagnostic experience a service engineer knows that a high value of some counter indicates that a certain power supply might be broken. This results in observation-diagnosis tuples, that can be used as examples to the inductive reasoning process. This way symptom-on-diagnosis mappings are derived, and a LUT is constructed. Service engineers at PMS that have much experience with a system use inductive reasoning. This way, they implicitly constructed a LUT by head. An automated implementation of inductive reasoning using black box information is a data mining technique. This method is described in Section 3.2. Like in the second quadrant, the LUT specifies effect-to-cause reasoning, and using it to diagnose the system can be implemented in a manual way, or automated way.

4. The fourth quadrant is not applicable (N/A). It is not possible to reason from cause-to-effect using only black box information.

Table 3.1 shows the approaches that are considered in this thesis. PMS-1, PMS-2, PMS-3 and PMS-4 are four approaches that are used at Philips Medical Systems. PMS-1 refers to the activity within PMS in which Symptom-Cause-and-Solution sheets, Error-on-Solution database and FIPs are constructed. These are the LUTs constructed and used at PMS. The project in which Philips Cardio-Vascular X-Ray Systems are remotely monitored (ServiceWAX) provides another LUT, and is also referred to by PMS-1. The difference with the other LUTs is that the expert uses real-life observations in its inference, instead of using observations made in test situations or observations that he/she would expect. PMS-2 refers to the way experts, and very skilled service engineers, diagnose systems when faced with a failure. PMS-3 refers to the way of reasoning in which service engineers learn symptoms-on-diagnosis mappings from experience. PMS-4 refers to the way service
specialists diagnose the systems. These are people from the regional and global help desks (see Section 2.2.1) that diagnose systems when a service engineer lacks sufficient information.

These four manual approaches are all part of the current approach to fault diagnosis at PMS. The relation with the current procedure to fault diagnosis, that is described by Section 2.2.1, is as follows. The first person that tempts to diagnose the system is a service engineer. This service engineer uses earlier experience with similar symptoms (PMS-3), uses the LUTs of approach PMS-1, or uses his/her own white box knowledge to reason about the system (PMS-2). If the service engineer is unable to successfully diagnose the system, specialists of the helpdesks try to use their abductive knowledge to derive the diagnosis (PMS-4). If these specialist are also unable to successfully diagnose the system, developers of the system could use their consistency-based knowledge to deduce the correct diagnosis (PMS-2). Finally, it is possible that these developers also use induction on a consistency-based model. This is not common practice, and therefore it is not shown in Table 3.1. Section 3.4 describes this concept in its description of the automated approach ‘Automated induction (WB)’.

Table 3.1 names four automated approaches. These approaches are considered in this chapter:

**Automated induction (BB):** A black box approach that uses historical data. Historical data are observations and diagnoses of (many) previous days that the system was operational. This historical data contains possible correlations between observables and diagnoses made by service engineers. A data mining technique automates the induction of observation-diagnosis tuples to rules that are inserted into the LUC.

**Automated induction (WB):** A white box approach that only uses the structure of a system and passed/failed-outcomes of observables. A data mining technique uses the pass/failed outcome
of output observables and structure to pinpoint to one or more suspected components. A well-known example of this approach is Pinpoint [7].

**Off-line inference:** A white box approach in which experts use their system knowledge to deduce and/or abduce a mapping between symptoms and diagnoses.

**MBD:** A white box approach in which an automated engine deduces diagnoses from a consistency-based model and real-life observations. The model defines cause-to-effect relations based on first principles.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Quadrant</th>
<th>Reasoning</th>
<th>Off-line/Online</th>
<th>Diagnosis</th>
<th>By Whom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMS-1</td>
<td>1 and 2</td>
<td>Manual deduction and abduction</td>
<td>Offline</td>
<td>Manual</td>
<td>Service Innovation, production and development.</td>
</tr>
<tr>
<td>PMS-2</td>
<td>1 and 2</td>
<td>Manual deduction and abduction</td>
<td>Online</td>
<td>Manual</td>
<td>Service Engineers and experts.</td>
</tr>
<tr>
<td>PMS-3</td>
<td>3</td>
<td>Manual induction</td>
<td>Off-line</td>
<td>Manual</td>
<td>Service engineers</td>
</tr>
<tr>
<td>PMS-4</td>
<td>2</td>
<td>Abduction</td>
<td>Off-line</td>
<td>Manual</td>
<td>Service specialists</td>
</tr>
<tr>
<td>Automated Induction (BB)</td>
<td>3</td>
<td>Automated induction</td>
<td>(Online)</td>
<td>Automated</td>
<td>Some automated entity (see Section 3.2)</td>
</tr>
<tr>
<td>Automated Induction (WB)</td>
<td>1</td>
<td>Automated induction</td>
<td>(Online)</td>
<td>Automated</td>
<td>Some automated entity (see Section 3.4)</td>
</tr>
<tr>
<td>Off-line Inference</td>
<td>1 and 2</td>
<td>Manual deduction and abduction</td>
<td>Offline</td>
<td>Automated</td>
<td>Expert system (see Section 3.3)</td>
</tr>
<tr>
<td>MBD</td>
<td>1 (and 2 and 3)</td>
<td>Automated deduction</td>
<td>Online</td>
<td>Automated</td>
<td>MBD engine (see Section 3.5 and Chapter 4)</td>
</tr>
</tbody>
</table>

Table 3.1: Evaluation approaches.

Table 3.1 classifies MBD in quadrant 1 of Figure 3.1. However, in MBD it is also possible to emulate fault diagnosis approaches in quadrant 2 and 3. This is useful, because sometimes it is not possible to formalize a consistency-based model of the system. Another note to Table 3.1 is that the approaches that use inductive reasoning need a history of diagnoses before diagnoses can be inferred realtime. Therefore, the ‘online’ predicate of approaches ‘Automated Induction (BB)’ and ‘Automated Induction (WB)’ is written between parentheses; it only holds after a certain time period. In the following sections, the approaches of Table 3.1 are shortly discussed by using the power supply example (as introduced in Section 2.1.2). This is done to provide the reader a more concrete understanding of these approaches. The evaluation of these approaches that is presented in the final section of this chapter is based upon own estimation. Therefore, sufficient understanding is needed to judge this evaluation. Figure 3.3 repeats the figure of the power supply example that was previously shown in Chapter 2.

---

[2] The model used in MBD can also define symptom-diagnosis mappings that are abductively deduced during the modeling activity. In other words, MBD is able to explicitly implement a LUC (see Chap. 4).
3.2 Automated Induction using Black Box Information

Automated induction using black box information can be automated by a data mining technique. This approach requires three things. Firstly, a technique for observing the system. Secondly, a way to determine the values of observables at the moment a failure occurred. Thirdly, determination of what component was actually broken for a particular failure. If these three pieces of information are available it is possible to search correlations between values of observables and adjudged broken components. Figure 3.2 shows a black box view on the power supply example system. The only information that is known are the input and output observables. Recall the example fault scenario in which we considered the broken CableB. Suppose that historical data extracted from the job sheets shows that tens of times CableB has been replaced. This is a realistic scenario considering Philips Medical Systems has hundreds of Cardio-Vascular systems all over the world. Suppose all repair actions of the service engineers actually recovered these failures of the power supply. Then, the log of each system that encountered such a failure, contains the following observation:

<table>
<thead>
<tr>
<th>status_Chiller</th>
<th>status_FD</th>
<th>status_TBCB</th>
<th>status_CRCB</th>
<th>status_Collimator</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

A well written data mining program would easily find this (linear) correlation between this observation and a replacement of CableB. Then, a supporting diagnostic engine could use found correlations to build a LUT (analogous to Table 3.2) and use it in future cases to produce diagnoses. Another occurrence of the listed observation could be looked up in the table when the historical data is sufficient. This way, the work of the service engineer is reused.

3.3 Off-line Inference

This section presents the approach to fault diagnosis in which experts apply the inference of the LUT beforehand. Again, the example of the power supply is used to improve concrete understanding of this technique. The idea is to take advantage of experts their knowledge of the system. Given

---

3 PMS has such a technique, namely ServiceWAX [2].
4 At PMS, the Philips Cardio-Vascular System often logs errors that can be used to identify the time of a failure.
5 Not possible at PMS, although a service engineer documents his/her actions in a job sheet (see Chapter 6).
observations on the system, most practically embedded in the log, an expert is able to draw more conclusions about possible malfunctioning components than the average service engineer. Let us now consider the power supply, as depicted in Figure 3.3. Suppose the on/off-stati of the five components on the right of the figure could be known by examining the log. Also, the on/off-status of the entire system is known by means of the log. Together, this input and the five outputs constitute the observations upon the system. These are all in the boolean domain, where true denotes that a component is on and false denotes that a component is off. Once the observables are known, the next step is to ’look for the symptoms in the observables’. For example, even a nonexpert understands that if the operator switches on the system, all components should eventually be on too.

So, every valuation of the boolean observables, that is does not meet this expectation, is a symptom of misbehavior. Let V be the set of all valuations, a symptom s is then:

$$\forall s \in V \mid \text{not} (status_{FD} = status_{TBCB} =
\text{status}_{CRCB} = \text{status}_{Collimator} = \text{status}_{Chiller} = \text{StartUp})$$ (3.3)

However, the only conclusion that can be drawn from this symptom is that ’something’ is wrong with the power supply. It is not possible to pinpoint to a subset of the system’s components. On the

---

6Currently, the stati of the Flat Detector, Collimator and Chiller are logged. Enabling the logging of the other two, the TBCB and CRCB, would require minor adjustments of the system.
3.4 Automated Induction using White Box Information

This section discusses a technique that automatically induces white box information. This technique requires a structural model of the system and a fault detection mechanism. The power supply example is very suitable to explain how such an approach would work. The outputs can be considered as error detection mechanisms. This supplies us with \textit{passed/failed} results of particular paths through the system. In our example, if CableB is broken the path to the Flat Detector will pass, but the path to the Collimator is likely to fail. For hardware, these paths (called traces) can be extracted from the system structure. Table 3.3 shows the traces that a broken CableB yields. This

<table>
<thead>
<tr>
<th>Y</th>
<th>Observations</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CHIL</td>
<td>FD</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.2: Partial diagnosis look-up table of the Power Supply system for single faults (startup = true)
Table 3.3: Input matrix for data analysis of the example fault scenario

3.5 Model-Based Approach to Fault Diagnosis

Model-Based fault Diagnosis is a white box technique that defines behavior of the system rather in terms of cause-to-effect than to effect-to-cause. Although all approaches described above can be implemented by means of a MBD technique. The next chapter gives a proper introduction to the subject, this section only offers a first idea.

An illustrative way to look at the solution that consistency-based approach provides, is to view it as the removal of assumptions to resolve inconsistencies between predicted and observed behavior. Using the assumption that all components function correctly, the behavioral model enables the calculation of the effects, given the cause (startup = true). In the example, if the system is switched on all components should be on too. During system behavior this prediction is compared to the effects that are actually observed. If CableB is broken the prediction and observations do not coincide. The prediction is that all variables are true, while it is observed that the TBCB, CRCB and Collimator are false. Therefore, the assumption that all components are healthy is wrong. Any assumption that a component (or group of components) is functioning correctly could be false. A diagnostic engine is responsible to search for the false assumptions. If the assumption that CableB is functioning correctly is dropped, the prediction falls together with the observations. Therefore, CableB is a single fault diagnosis. Obviously, it is much more likely that just this cable is broken than all components. Determining what assumptions do not hold can be seen as a search problem and is time/space complex. Continuing the search would state that dropping the assumption that the PDU is healthy yields another single fault diagnosis. The search process would also find many groups of components that can not all be healthy at the same time (for example the set of all components). These are the multiple fault diagnoses. Usually, a MBD engine produces a list of possible diagnoses. A calculation of probabilities is used to order the list. This way, a service engineer can use the output of the MBD approach to prioritize his/her diagnostic activities.

3.6 Evaluation of Approaches to Fault Diagnosis

In this section the approaches that are described in this chapter and the current approach (as described in Chap. 2) are evaluated using the items of section 2.4.1 as criteria. Table 3.4 shows...
this evaluation. The technique that is referred to as ‘current approach’ is a combination of the approaches PMS-1, PMS-2, PMS-3 and PMS-4 that are characterized by Table 3.1. The values of all scores range from very bad (−−) to very good (++) . The assessment of all scores is based on own opinion, experience and interviews with experts. The remainder of this section motivates the assessment of Table 3.4.

The estimation I made is that the current approach achieves better accuracy than an approach that uses automated induction on black box information, but worse than the three other automated approaches. This is because the other these three automated approaches use information of experts, while the current approach uses information of service engineers, and ‘Automated Induction (BB)’ only uses black box information. The speed of diagnosis is higher for an automated approach than for manual approaches. MBD is faster than the other automated approaches. This is in the domain of milliseconds, thus only essential when produced diagnoses are used for automated system recovery. Like accuracy, the diagnostic resolution is best when experts use their knowledge about the system. Either by applying the inference off-line (called ‘Off-line inference’ in Table 3.4), or by online inference of an automated entity that uses a model made by experts (MBD). Intuitively, inference of white box information scores better on diagnostic resolution than inference of black box information. The current approach is estimated having better diagnostic resolution as inference on black box information.

The context independency (denoted by ‘independence’ in Table 3.4) of techniques is worse for both the current approach and the inference of black box information. As said before, the current approach depends on the skills and experience of currently employed people. ‘Automated induction (BB)’ requires much data. This data must be produced by systems that have the same characteristics as systems that are diagnosed by the using the inductively constructed LUT. Furthermore, a manually inferred LUT of ‘Off-line inference’ scores less on context independency than ‘Automated induction (WB)’ and MBD. This is because these consistency-based models contain more knowledge. This is useful in case employees change jobs.

There has been no research of the development costs of the various diagnostic techniques. For PMS, the technique ‘Automated induction (BB)’ is cheapest, because the remote monitoring tools already provide most of the needed functionality. The runtime costs of the approaches has not been investigated as well. The only estimation made is that automated approaches require less runtime costs, because the labor costs are likely to be less.

The approaches that use a consistency-based model score highest on adaptability. This is because changes in the design do not require time consuming and error-prone inferencing of a new LUT (‘Off-line inference’ and ‘Current approach’). The inductive process of ‘Automated induction (BB)’ and ‘Current approach’ require that systems, that implemented the new design, already produced sufficient data. The explanation facility is best for the ‘Off-line inference’, ‘Automated Induction (WB)’ and MBD approaches. In these approaches, the diagnoses can be explained in terms of the consistency-based model, while the current approach and ‘Automated induction (BB)’ do not have such a model. MBD scores higher than ‘Off-line inference’ and ‘Automated induction (BB)’, because diagnoses made by ‘Off-line inference’ can only be explained by the experts that implemented it, and ‘Automated induction (WB)’ uses a consistency-based model that only specifies structure.

As a consequence of the evaluation given by Table 3.4, MBD is chosen as a proposed approach to fault diagnosis. It is likely to improve the fault diagnosis process at PMS, because it has equal score or higher score than the current approach. Also, it scores best of all automated approaches. The following chapter introduces model-based fault diagnosis.
### 3.6 Evaluation of Approaches to Fault Diagnosis

<table>
<thead>
<tr>
<th>Approach</th>
<th>Criteria</th>
<th>Accuracy</th>
<th>Speed</th>
<th>Diagnostic Resolution</th>
<th>Indepen-dency</th>
<th>Development Costs</th>
<th>Runtime Costs</th>
<th>Adaptability</th>
<th>Explanation Facility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current approach</td>
<td></td>
<td>0</td>
<td>−−</td>
<td>0</td>
<td>−−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Automated Induction (BB)</td>
<td></td>
<td>−−</td>
<td>+</td>
<td>−−</td>
<td>−−</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Off-line Inference</td>
<td></td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>−</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Automated Induction (WB)</td>
<td></td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MBD</td>
<td></td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 3.4: Evaluation approaches.
Chapter 4

Model-Based Fault Diagnosis

The previous chapter concluded with a rationale for developing a model-based approach to fault diagnosis. Model-Based Diagnosis (MBD) is the model-based approach to fault diagnosis. This chapter presents its basic notions.

MBD is a reasoning technique to isolate root causes of failures, that uses a clearly separated model of a system. The model is compositional, and specifies all information that is relevant for diagnosing a certain system. A diagnostic engine, that is application-independent, operates on this model in order to perform its task; diagnosing the system. This idea, namely separating a compositional model and an application-independent engine that reasons in terms of this model comes from the Model-Based Computing paradigm. The task is not restricted to fault diagnosis: there are model-based approaches to maintenance, recovery planning, testability analysis at design-time and health prognosis.

The task of model-based diagnosis is to identify causes of malfunctioning. The system does not necessarily have to be an artifact. MBD could also be used for diagnosing other kinds of systems. Its applicability is diverse:

• **Engineering and Physical Systems.** These systems are man-made, and therefore, it is relatively easy to implement model-based approaches to these systems. Despite the hardware-to-software shift a significant part of Philips Cardio-Vascular X-Ray System still consists of these type of subsystems. The work presented in this thesis is in this area.

• **Software and Program Logics.** An increased part of today’s embedded systems, and more generally, of today’s society, consists of software subsystems. MBD seems a fitting candidate to reduce the complexity that one has to deal with, when checking the correct functioning of software.

• **Biological and Eco-Systems.** Living organisms are much more complex than man-made artifacts. Therefore, diagnosing them is much more problematic. Many expert systems that map symptoms on diagnoses exist. It seems reasonable that a model-based approach that also specifies nominal behavior increases diagnostic performance.

• **Complex Systems.** During the last century scientist have developed models to understand incredibly complex phenomena such as economy, climate, internet, etc. The accuracy of these models has improved throughout the years. In a model-based approach, these models are formalized in some language. The models and observations could be fed into a diagnostic engine in order to automatically draw conclusions about their states, and use them to our benefit.
4.1 Fundamentals

MBD is the process of finding differences between behavior predicted by a model, and behavior observed during runtime operation. The model is assumed to be correct, so the differences should be explained by faulty components.

Consider Figure 4.1. It shows a simple digital circuit, consisting of 3 inverters: A, B and C. This classical diagnosis example is commonly used to introduce the basic notions of MBD. This text does so likewise. Let \( w = 1 \), then \( y \) and \( z \) should be 1 as well. If observations during runtime indicate that \( y = 0 \) and \( z = 1 \), there is a discrepancy between observed and predicted behavior. Any discrepancy is called a symptom, and in this case the symptom is that \( y = 0 \) while \( y = 1 \) is predicted. This symptom could be explained by the malfunctioning of inverter B. However, it could also be that both inverter A and inverter C are broken. No subset of \{B\} or \{A, C\} is able to explain the symptom, and therefore \{B\}, \{A, C\} is called the minimal fault set. Another possible candidate is that all inverters are broken. Actually, all supersets of \{B\} or \{A,C\} are candidates, although with lower probability.

The question is, how could MBD automate the reasoning of above? The model-based approach requires two artifacts: a model, and a diagnostic engine that operates on that model. These two ingredients of MBD are described now.

Model

The model of the MBD approach describes the behavior and structure of the components. Let \( h \) indicate the health of a component. If \( h = 1 \), the component is "healthy" and obeys certain behavior rules. For a combinational system, such as the 3 inverters example, the behavioral rules can be formalized to propositional logic:

\[
h_A \Rightarrow (x \leftrightarrow \neg w)
\]
This formalization from the concept of 3 inverters to these behavioral rules should be done by humans, and is called the modeling activity. This activity, that could most conveniently be performed by developers of a system, is believed to be the most difficult part of MBD.

**Diagnostic Engine**

The second artifact of MBD is a *diagnostic engine*. This diagnostic engine implements an inference mechanism that is able to produce diagnoses based on any formally described model, and observations made during runtime operation. This can be done by solving the system of equations, as the model provides, by using rules from propositional logic. Figure 4.1 shows the variables that can be observed; \( w, y \) and \( z \). Substituting these *observables* in the system of equations gives:

\[
\begin{align*}
  h_A &\Rightarrow \neg x \\
  h_B &\Rightarrow x \\
  h_C &\Rightarrow \neg x
\end{align*}
\]  

(4.2)

Then, applying the rule \((p \Rightarrow q) \Leftrightarrow (\neg p \lor q)\) yields:

\[
\neg h_A \lor \neg x \land \neg h_B \lor x \land \neg h_C \lor \neg x = 1
\]  

(4.3)

This can be rewritten to DNF-form:

\[
\neg h_A \neg h_B \neg h_C \lor \neg h_A \neg h_B \neg x \lor \neg h_A \neg h_C x \lor \neg h_B \neg h_C \neg x \lor \neg h_B \neg x = 1
\]  

(4.4)

Finally, reducing it to the following prime implicants yields:

\[
\neg h_A \neg h_C x \lor \neg h_B \neg x = 1
\]  

(4.5)

This result is the minimal fault set; either component A and C are broken (if \( x = 1 \)) or component B is broken (if \( x = 0 \)).

Another way for computing diagnoses is to use conflicts in order to produce the diagnosis. A conflict is a set of components that cannot be healthy all together. For example, given the symptom "\( y = 0 \) while \( y = 1 \) is predicted", the set \{A, B, C\} is a conflict. In other words, the assumption that component A, B and C are all healthy should be removed. Section 3.5 used this view on solving the diagnosis problem, for giving the reader a first idea on MBD. A minimal conflict is a set of components that is no longer a conflict if you remove one of its members. These are the interesting sets, because they correspond to diagnoses with high probability. Applying the resolution rule \((p \lor q) \land (r \lor \neg q) \Rightarrow (p \lor r)\) to Equation 4.3 yields:

\[
\neg h_A \lor \neg h_B \lor \neg h_C = 1
\]  

(4.6)

Then, using *De Morgan’s Laws*:

\[
\begin{align*}
  \neg (\neg h_A \lor \neg h_B) \land (\neg h_B \lor \neg h_C) &= 0 \\
  \neg (\neg h_A \lor \neg h_B) \lor \neg (\neg h_B \lor \neg h_C) &= 0 \\
  \neg h_A \lor \neg h_B \lor \neg h_B \lor \neg h_C &= 0
\end{align*}
\]  

(4.7)
Thus, \{A, B\} and \{B, C\} are conflict sets. Then, finding the minimal conflicts can be achieved by applying an algorithm for the Hitting Set problem. This also results in the sets \{B\} and \{A, C\}. This method is used in [11], and is based upon the extraction of conflicts and candidates from a lattice in order to produce the diagnoses. Results using this method yield exactly the same results, as when writing the propositional model to DNF-form.

In summary, MBD of combinational systems means solving a model for \(h\) using propositional logic. The next section describes the MBD implementation that is used in this thesis. It uses the first method, writing the model to DNF-form, for solving the model for \(h\).

### 4.2 Model-Based Diagnosis with **LYDIA**

This section introduces the system modeling language **LYDIA (Language for sYstem DIAgnosis)**, and the corresponding **LYDIA** toolkit. Other diagnostic systems are GDE by de Kleer and Williams [11], Sherlock by de Kleer and Williams [12], and Livingston by Williams and Nayak [30]. See [28] for a comprehensive introduction to **LYDIA**. The **LYDIA** toolkit consists of tools enabling the diagnosis of systems; the diagnostic engine. The language **LYDIA** is able to describe structure and behavior of a wide range of phenomena. It is compositional, which means that it can be used to build up a system from previously defined constituents. These constituents can be hardware components, processes and even software subsystems. The following is - like in the previous section - split up into the two artifacts MBD requires: the modeling language **LYDIA** and the diagnostic engine that operates on a **LYDIA** model.

#### **LYDIA** Language

**Lydia** is a declarative language. Each **LYDIA** statement is a proposition. All statements are true, and apply concurrently. Recall the example of the 3 inverters, as shown in Figure 4.1. Its propositional system of equations of 4.2 can easily be translated to **LYDIA**. The behavioral rule for one inverter (in proposition logic: \(h \Rightarrow (i \iff \neg o)\)) is defined in **LYDIA** as follows:

```lydia
system inverter(bool i, h, o) {
    //declaration health variable
    attribute health(h) = true;
    attribute probability(h) = h ? 0.99 : 0.01;

    //behavioral rule
    h => (i = !o);  // If healthy, output equals inverse of the input
}
```

Then, 3 of such inverters can be connected by the following structural description:

```lydia
#include inverter.sys  // Include the definition of one inverter

system inverters3 {
    bool w,                 //input
    bool hA, hB, hC,       //healths
    bool y, z             //outputs
} {
    // Declaration internal variable
```
bool x;

// Declaration observables
attribute observable (w, y, z) = true;

// Declaration inverters
system inverter invA, invB, invC;

// Connect the 3 inverters
invA (w, hA, x);
invB (x, hB, y);
invC (x, hC, z);

diagnosis

The three lines that start with invA, invB and invC each instantiate an inverter, and use the names of variables to specify how the signals are interconnected. It is possible to use the system inverter3 in turn to describe higher level systems. The parameters w, y, z allow the interconnection to other components. These three variables are also declared as observables. Variable x is declared as an internal variable. The specification of an inverter includes the declaration of the health variable, with associated probability. These probabilities that components are broken (or, that a health variable is false) are used to order the list of diagnosis that the diagnostic engine outputs. In most cases, this list has more than one item. This means that the output of the diagnostic engine still has uncertainty. Refer to Section 4.5 for an introduction on uncertainty. The next section describes the diagnostic tool, that is used to operate on LYDIA models, such as the 3-inverter model of this section.

LYDIA Diagnostic Engine

There are a number of tools that operate on the LYDIA models. Design and implementation, as well as a comprehensive description, of the LYDIA tool set are beyond the scope of the thesis (see Lydia documentation: [1, 28, 27, 13]). The tools, that are discussed for the work of this thesis are:

lydia: Combined LYDIA compiler and diagnostic engine. Diagnoses a system. Inputs are the observations on the system, and a LYDIA model. Output is a list of possible diagnoses, ordered by probability.

cdas: The separate diagnostic engine, that is also part of the above mentioned lydia tool. Input is a compiled format of the original LYDIA model. Currently, the variables and time of a LYDIA model, to be diagnosed by cdas, should be in the boolean domain. There is a separate engine for models that use multi-valued (more than two possible values) variables, called mvcdas. It is not possible to diagnose models that are in the integer or floating point domain.

lsim: Interprets and simulates continuous-time LYDIA models. The time and variables are allowed to be in the floating point domain. To some extent, this tool could also be used for diagnosing systems.

4.3 Basic Assumptions

In the work for this thesis it is assumed that faults occur independently. However, in many real-life systems it is common that one faults evokes another, thus faults are not independent. The
4.3 Basic Assumptions

Model-Based Fault Diagnosis

probability calculation of the LYDIA tool uses this assumption. Despite this, it is possible to reason about dependent faults. The inference mechanism of the MBD technique specifies a dependency between faults, the inference mechanism takes this into account.

Other topics that are not yet fully solved by MBD technology are state and time.

The previous part of this chapter briefly introduced the basics of MBD and the LYDIA approach to MBD. The remainder of this chapter introduces some more advanced topics. It is restricted to the parts that are necessary to achieve the goal of the project described in this thesis: presenting a proof-of-concept of a model-based approach to fault diagnosis, aimed at the Philips Cardio-Vascular X-Ray System. This proof-of-concept, that will be presented in the next chapter, requires that MBD achieves a higher diagnostic performance than the current approach. A higher diagnostic performance can be achieved by:

1. **Improving the diagnostic engine.** Better and more sophisticated algorithms for solving a propositional model could increase the speed of diagnosis. This way, also larger-sized diagnostic problems become feasible. The work of [15] and [14] is in this area.

2. **Improving the model.** If the model defines more relevant information, the diagnostic resolution and accuracy of diagnoses increase.

3. **Increasing the observability.** A system that is in its use phase has a fixed observability. Part of the observations can be observed automatically, because it is available in some digital format. Other observations can only be made manually. Extra sensors could increase the observability of the system, and improve accuracy of diagnoses and diagnostic resolution.

Considering the three points mentioned above, what are the important topics of model-based diagnosis, in respect to the work presented in this thesis?

- **ad 1.** Currently, the time to produce a diagnosis is taking days, while a successful model-based approach is able to produce diagnosis within milliseconds. Consequently, the proof-of-concept does not require improvement of the diagnostic engine. A very slow algorithm already improves speed of diagnosis.

- **ad 2.** Improving a model, on the other hand, is within the scope of this thesis. The next chapter discusses a case study of the model-based approach, and better models in this work allow for a proof of higher diagnostic performance. Section 4.4 introduces types of models, and ways to take on the modeling problem. In order to optimize the solution to the problem, Section 4.5 introduces a metric that can be used to estimate the quality of a model.

- **ad 3.** Increasing observability is likely to increase diagnostic performance. Although, the Philips Cardio-Vascular System is fixed in respect to this work, a promising advantage is the following: the metric introduced in Section 4.5 allows to study which additional measurements have the highest effect on the accuracy. This way, it is possible to improve the diagnostic performance of systems in the design phase.

So, improving the diagnostic engine is not important for this work, increasing observability is nice, but modeling is very important. Therefore, the next section introduces the modeling activity.
4.4 Modeling

The previous section concluded by pointing out the importance of a good model. Engineers and scientist use models to understand the behavior or construction of physical systems. The differences between these models and reality drive the work that is being done. Scientist try to refine their models, in order to remove differences, and obtain better understanding. Engineers try to search for anomalies in their artifacts, that explain the differences between models and observed behavior. Constructing the model of a system is not a trivial activity. It is hard to determine what information is relevant for a particular use of the model. Superfluous information easily degrades the conclusion that could be drawn from the model. In fault diagnosis, the model should specify all information that can be used to draw conclusions about the health of components, not more. As explained in the previous sections of this chapter, in MBD, the model is formalized, the physical system is observed, and differences between the two are input to the diagnostic engine for producing a list of possible diagnoses. Irrelevant information could increase this list, while relevant information could shorten it. The remainder of this section presents the basics of modeling.

4.4.1 Types of models

There are various types of models. The type depends on the kind of information that is specified, and the way it is stated. There are four types of models:

**Structural Model** Description of the system that only defines the set of components and its interconnections. No behavioral information is specified. Without facts whether or not an observation is allowed it is impossible to derive any diagnosis. In the 3-inverter example, the behavioral rule for one inverter is not specified, but the fact that the input is connected to the output of the inverter. The structural definition becomes:

```plaintext
system inverter(bool i, h, o) {
    attribute health(h) = true;
    attribute probability(h) = h ? 0.99 : 0.01;

    // If healthy, a correct input results in a correct output
    h => (i = o);
}

system inverters3 {
    bool correct_w, //input
    bool hA, hB, hC, //healths
    bool correct_y, correct_z //outputs
} {
    // Declaration internal variables
    bool correct_x;

    // Declaration observables
    attribute observable (correct_y, correct_z) = true;

    // Declaration inverters
    system inverter invA, invB, invC;
```
//input is assumed to be correct
correct_w = true;

// Connect the 3 inverters
invA ( correct_w, hA, correct_x);
invB ( correct_x, hB, correct_y);
invC ( correct_x, hC, correct_z);

The user of the diagnostic system should be able to determine whether a specific output is
correct or not. These are the observables.

**Weak Fault Model** This model only defines the nominal behavior of the system. The description
of the 3-inverters in Section 4.2 is a *weak model*; it does not define how an inverter behaves if
something has been broken (if \( h = 0 \)).

**Strong Fault Model** Description of the system that defines all modes of operation. A *mode of
operation* is a state of a component in which it obeys an unique behavioral rule. The nominal
behavior of a weak model could specify more than one nominal modes of operation. A strong
fault model also defines all known false modes of operation. The corresponding *LYDIA* code
follows the form \( !h \Rightarrow s \). Examples are stuck-at-zero, stuck-at-one, etc. This way, it is
possible in MBD to define an abductive model [18], as discussed in Chapter 3. A strong
model of the 3-inverters is:

```c
system inverter(bool i, h, o) {
    h \Rightarrow (i = !o); // If healthy, output equals inverse of the input
    !h \Rightarrow (o = 0); // If unhealthy, the output is stuck-at-zero
}
```

**Model that is not from First Principles** It might be that it is practically impossible, for certain
parts of the system, to define the correct behavioral rules. In these cases it is possible to
include mappings of symptoms on broken components (of the form \( s \rightarrow f' \)). MBD diagnosis
does not forbid to use these explicitly specified statements, that are abductively or deductively
derived by humans. A (partial) model that is not based on first principles, and that is equal to
the weak model of the 3-inverter example is:

```c
// mapping of symptom (y=0, z=1) on diagnosis
((y=0) and (z=1)) \Rightarrow (  
    !hB  
    or (!hA and !hC)  
)
```

In theory, the expressive power of rules that are not based on first principles is the same. After
all, the rules are equivalent to a consistency-based model ('\( s \rightarrow f' \) corresponds to \( \neg f \rightarrow \neg s' \)).
The only difference is the way how relevant information is specified. As explained in Chapter 3, this effects the time to create the model, is error-prone, and inflexible in case of a design change. But is some cases there is no other possibility.

### 4.4.2 Granularity

The first step of making a model is deciding the granularity, in other words ‘level of detail’, of the model. Granularity has two dimensions; depth and breadth.

If you consider depth in granularity, it is possible to tune the level of abstraction. For each system, there is an infinite number of levels of abstraction. Consider the 3-inverter example again. On the highest level of granularity the 3-inverter system has one entity; the 3-inverter system. In this case, the LYDIA model has one health variable, and the only outcome of the diagnostic engine is whether or not the system is broken. This means that, in case of malfunctioning, a supervisory controller would have to replace the entire system with a healthy one. In this case, the costs to recover the system are higher than when a model that specifies the system at a lower level of granularity is used. On one lower level, the model distinguishes 3 entities, namely the 3 inverters, as shown in Figure 4.1. The results of these diagnoses state for each inverter if it has to be replaced with a healthy one, or that it could be preserved. Going another step lower could be that the model also specifies the transistor and resistor that (could) implement the inverter. But usually these components cannot be replaced, and diagnosing this level of detail is of no added value.

If you consider breadth in granularity, it is possible to include or exclude certain components from being modeled. Given a certain level of granularity, what components should, and which should not be specified by the model? For example, the 3-inverter system might include a casing that protects the system from the outside. Should this component be included? Maybe the casing gets easily damaged, and is the cause for the breaking down of inverters in the first place. If the casing never breaks, adding it, would only unnecessarily increase complexity. The particular considerations greatly depend on the specific system, the environment, and the motives of the modeler.

As said before in this thesis, modeling of a system is not a trivial activity. Although modeling is a divergent activity without general procedures, this thesis considers three possible approaches to modeling:

1. Modeling the entire system. The modeler chooses its granularity, based on domain-dependent and general considerations (e.g. Could the supervisory controller use the correct or incorrect functioning of this component for recovering the system? Does this component ever brake down? Is it possible to, directly or indirectly, observe the state?)

2. Modeling to cover faults. The modeler composes a list of all known faults (things that could brake down). Only those components that could reveal a fault are included in the model.

3. Modeling based on log data. Most existing systems log values of certain parameters to allow for monitoring the state of the system. Experts, mostly developers of the system, know what to conclude about the system state given a set of parameters. The modeler chooses the granularity of the system in order to reproduce the same diagnosis for a certain set of parameters.

Of course, a modeler could use more approaches in parallel, in order to achieve a high quality model.

### 4.4.3 Discretization of Observations

The output of sensors, or other sources of data in a system, usually contains much resolution. Most sensor output is continuous. It is hard to use variables with high levels of resolution in a model, and
still be able to produce accurate diagnoses. Discretization is the process of mapping variables in a many-valued domain to variables in a few-valued domain.

Some terminology regarding to this topic is useful. Note that this terminology is defined for this thesis, and might not correspond to terminology in other literature. Suppose there is a system that has 4 components, and logs 5 signals to a log file. Three of those signals are in the floating point domain ($f_1, f_2$ and $f_3$), and two are in the boolean domain ($b_1, b_2$). Instances of the tuple $(f_1, f_2, f_3, b_1, b_2)$ are called system data, and this term is generally defined as:

**System Data:** All unprocessed digital data that the target system produces.

Each instance of the system data, for example $(500, 400, 50, true, true)$, is part of a fault scenario, and the definition of this term is:

**Fault scenario:** A particular failure of the system. It is characterized by the system data that is logged at the moment a failure occurs, in addition with the actual broken component.

Often, the system data is hard to interpret, and its values should be discretized to a domain with few members. The term discretization is defined as follows:

**Discretization:** A mapping of variables within the system data, that are in a many-valued domain, on observables, that are in a few-valued domain.

Thus, discretization maps the variables in the system data to a domain with a finite number of members. It could be that the source domain of system data variables is unknown. Preferable, the few-valued domain has as few as possible members, say 2, 3, or 4. Each observable is a derivative of the system data. For example, suppose the system data $(f_1, f_2, f_3, b_1, b_2)$ is mapped on the two observables $(o_1, o_2)$ by a discretization $D$. $D$ defines two derivatives of the system data, say $o_1 = f_1 - f_2 - f_3 > 10$ and $o_2 = b_1 \lor b_2$. The result of the discretization for a fault scenario with values $(500, 400, 50, true, true)$ is then $(true, true)$.

Because of the many-to-few mapping it is very likely that various fault scenarios are not different in the conclusions that could be drawn. These are said to correspond to the same fault category. A definition is:

**Fault category:** A set of fault scenarios that have the same values for (a subset of) the observables.

Suppose the system data of two fault scenarios is resp. $(500, 400, 50, true, true)$ and $(50, 20, 10, false, true)$. Discretization $D$ maps the system data of both fault scenarios to the observables $(o_1, o_2) = (true, true)$. Consequently, the MBD engine infers the same diagnoses. Therefore, these fault scenarios belong to the same fault category.

The system data of the 3-inverter example, as well as the example of Section 4.6, only consists of variables that are already in the boolean domain. The next chapter presents a complex example that shows how a discretization simplifies modeling. It uses the concepts system data, discretization, fault scenario and fault category.

### 4.4.4 Compositional Subsystems

Another important issue is to make subsystem entities compositional. This is one of the characteristics of model-based computing. The 3-inverter example presented above is specified in a compositional model. A non-compositional model would be:
By the way, this is an intermediate step within the LYDIA compiler. It does not change the semantics of the model. Compositional models improve maintainability, allow for reuse, and improve readability. Suppose the inverters are implemented by resistive-drain (that is, using one transistor and one resistor). For some reason, the supervisory controller is interested in the health of these transistors and resistors. The only code that requires to be updated is the specification of a single inverter.

### 4.5 Entropy Gain

This section discusses a metric that can be used to estimate the diagnostic performance of a specific MBD implementation, namely entropy. Entropy can be used as a feedback mechanism to improve the quality of a model, or decide which measurements are best. Entropy is a heuristic, introduced by Shannon [26], to quantify information. De Kleer and Williams suggested to use entropy as a heuristic for quantifying the uncertainty of diagnosis [11]. The idea is as follows. The outcome of the diagnostic engine has some uncertainty. If no information about a system is available, anything can be broken, and the uncertainty is at its maximum. A stronger model, or more observations upon the physical system, can shorten the list of possible diagnoses; the uncertainty decreases. The extent to which a specific MBD implementation decreases uncertainty is called entropy gain (although entropy loss seems to make more sense, this thesis follows the convention to use the term entropy gain). Entropy is measurable in bits. This enables the comparison of various MBD implementations. In other words, entropy in MBD could be used to optimize models, and as a mean to decide the best measurements on the target system.

Below, entropy is used to determine the best measurement points of a classical diagnosis example, namely a digital circuit of 4 inverters.

#### 4.5.1 Best Next Measurements

As explained before, a list of diagnoses is consistent with a certain set of observations. Additional measurement could decrease the number of possibilities, thus improve diagnostic resolution. This is called sequential fault diagnosis [23]. In industry, doing an additional measurement increases costs. Entropy outcomes can be used to decide which measurements are best to be done first. Consider Figure 4.2. It shows a simple digital circuit of 4 inverters, A, B, C and D, in a pipeline structure. This section uses entropy to decide the best points to measure in this circuit. The input and output
4.5 Entropy Gain Model-Based Fault Diagnosis

Figure 4.2: 4-inverter example

are already known; the input $x$ is true and the output $y$ is false. The considered points are referred to by $a$, $b$ and $c$.

The prerequisite for the entropy calculation is the assignment of an *a priori probability* to each health variable. This a priori probability is the probability that $h=0$, without having made any observations. The 4-inverter system has 4 health variables, with a priori probabilities:

$$p(\neg h_A) = 0.01$$
$$p(\neg h_B) = 0.01$$
$$p(\neg h_C) = 0.01$$
$$p(\neg h_D) = 0.01$$

The interpretation of, for example $p(\neg h_A) = 0.01$, is not that randomly picking $n$ components yields $0.01 \times n$ broken components. The meaning depends on the model of the system. If no observations have been made, all single faults and multiple faults are possible. There are 4 health variables. This means that if $x$ and $y$ are not observed, the number of possible diagnoses is $2^4 = 16$. This information, the existence of 16 possible diagnoses, can be stored in 4 bits. The corresponding outcome of the LYDIA diagnostic engine is as follows:

$(0.960596) \ h_A = \text{true}, \ h_B = \text{true}, \ h_C = \text{true}, \ h_D = \text{true}$
$(0.00970299) \ h_A = \text{false}, \ h_B = \text{true}, \ h_C = \text{true}, \ h_D = \text{true}$
$(0.00970299) \ h_A = \text{true}, \ h_B = \text{false}, \ h_C = \text{true}, \ h_D = \text{true}$
$(0.00970299) \ h_A = \text{true}, \ h_B = \text{false}, \ h_C = \text{false}, \ h_D = \text{true}$
$(0.00970299) \ h_A = \text{true}, \ h_B = \text{true}, \ h_C = \text{false}, \ h_D = \text{true}$

The firstly listed diagnosis, that states that all inverters are healthy, has highest probability. Specific diagnoses are referred to as *health vectors*. The correct functioning of all inverters is denoted by the health vector $\bar{h} = \{1, 1, 1, 1\}$, and its probability is calculated by using the formula:

$$p(\{h_A, h_B, h_C, h_D\}) = p(h_A) \times p(h_B) \times p(h_C) \times p(h_D)$$

This formula assumes that the variables are independent: the correct or incorrect functioning of one component has no effect on the health of any other component. The interpretation of, $p(\bar{h} = \{1, 1, 1, 1\}) = 0.970596$, is that diagnosing the 4-inverter system at an arbitrary moment, without any observations being made until that moment, yields a probability of 0.97% that the entire system is healthy. $p(\bar{h} = \{0, 1, 1, 1\}) = 0.00970299$ means that there is a probability of 0.01% that $h_A=0$. Also, there is probability of 0.01% that $h_B=0$, etcetera.

The *a priori* probabilities of the health vectors allow for the calculation of entropy. The entropy $(H)$ is defined as:

$$H = - \sum p_i \log p_i,$$
where \( p_i \) is the probability that health vector \( i \) is the actual candidate given a specific set of observations. Formula 4.10 is a cost function to estimate the expected cost of identifying the actual candidate. It is constructed as follows. The cost of locating a particular candidate is proportional to \( \log p_i^{-1} \) (so, a binary search through \( p_i^{-1} \) objects). The expected costs of identifying one candidate is the multiplication of the costs to locate it (\( \log p_i^{-1} \)) and the probability that the candidate is the actual candidate (\( p_i \)). The entropy \( H \), defined by Formula 4.10, adds the costs of all candidates (\( \sum p_i \log p_i^{-1} = -\sum p_i \log p_i \)).

A mode catalog is a table that specifies all possible observations for each health vector, and can be derived from the model. The used 4-inverter model, as listed in Appendix C.1.4, is a strong model, and the fault mode of 1 inverter is defined by: \( \neg h_i \Rightarrow (o = i) \). The mode catalog of this 4-inverter model contains 8 entries, when none of the observables \( a \), \( b \), and \( c \) are observed. This information, the existence of 8 health vectors, can be stored in 3 bits, in contrast to the 4 bits needed for storing the information that there are 16 health vectors. This does not imply that the entropy gain is \( 4 - 3 = 1 \) bit, because entropy also depends on the probability of each health candidate (the entropy gain of measuring the input \( x \) and output \( y \) is 0.064). Table 4.1 shows the 4-inverter model, when \( x=1 \) and \( y=0 \). The entropy of this mode catalog (no additional observation are made), is 0.260. This means that the uncertainty of the outcome of the MBD engine is 0.260.

Consider that \( a \) is being measured. There are two possibilities: \( a=0 \) or \( a=1 \). In either case another set of entries is consistent with the observations. These possible new mode catalogs define the cost to locate the actual candidate. The expected entropy of observing \( a \) is 0.046. The entropy gain is defined as the entropy a priori model minus the expected entropy of observing \( a \): 0.260 - 0.046 = 0.213. The entropy gain of observing \( b \) and \( c \) is respectively 0.221 and 0.213. Thus, the entropy gain of observing \( b \) is highest, and is the best choice for to measure.

### 4.6 MBD on the Power Supply

This section presents a real-life application of MBD on the power supply example. The model is shown in Appendix C.2.1. The model specifies six systems, namely \texttt{Cable, Fuse, Low\_voltage\_power\_supply, PDU, Unit, and Power\_Supply}. The specification of the behavior is very abstract. The model describes the power supply example as a digital system. The used variables are all in the boolean domain. A value of \( 1 \) denotes that there is current, and a value of \( 0 \) denotes that there is no current. For example, the model of a \texttt{Cable} specifies that when a cable is healthy it conducts current, using the following definition:
system Cable {
  bool p, //input
  bool q //output
}

//declaration health variable
bool h;
attribute health(h) = true;
attribute probability(h) = h ? 0.92 : 0.08;

//definition behavior
h => ( q = p );
}

The boolean variable \( p \) is the input of the cable, and cable \( q \) is the output of the cable. Each instantiation of a cable declares a health variable that defines a probability of 8 percent of being false. This value, and the health probability values of the other system, are chosen to order the list of diagnoses that the LYDIA diagnostic engine produces. The system power_supply defines the structure of the power supply example, and by doing so, it defines the behavior of the whole power supply example system. There are 6 variables defined as observables; the inputs and outputs to the system. In the fault scenario described earlier, the values of these observables are:

<table>
<thead>
<tr>
<th>Status</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>status_Chiller</td>
<td>true</td>
</tr>
<tr>
<td>status_FD</td>
<td>true</td>
</tr>
<tr>
<td>status_TBCB</td>
<td>false</td>
</tr>
<tr>
<td>status_CRCB</td>
<td>false</td>
</tr>
<tr>
<td>status_Collimator</td>
<td>false</td>
</tr>
</tbody>
</table>

The result of inserting these values in the diagnostic engine LYDIA produces a long list of possible diagnoses. The malfunctioning of CableB has highest probability. The second item on the produced list is that the PDU is at false. The results of a MBD implementation can be characterized by a LUT. The LUT is constructed by inserting all possible observations to the diagnostic engine LYDIA. Table 3.2 shows the LUT that presents the results of the model listed in Appendix C.2.1.

4.6.1 Strong Model of the Power Supply

The model listed in Appendix C.2.1 is a weak model of the power supply example. A strong model of the power supply example is listed in Appendix C.2.2. This model specifies that a component conducts no current if it is broken. This affects the results that the diagnostic engine LYDIA produces. Table 4.2 shows the new LUT for single faults. The values of the observables in entry 7, 8 and 15 no longer imply that the PDU could be broken. The strong model of the PDU specifies that when PDU malfunctions all outputs are stuck-at-zero, so none of the components could get voltage. So, the fault scenario described above yields that CableB is at false, without any uncertainty. For fault scenarios that have observables that map on the 15th entry of Table 4.2 this means that the Chameleon is broken without any uncertainty. The disadvantage of strong models is that the model is less robust. Suppose the PDU also has fault modes that imply that only one of its outputs is stuck-at-zero. If undefined fault modes exist, although not expected, the diagnostic engine no longer produces correct results for certain fault scenarios.

However, the amount of multiple fault explanations for certain observable values greatly decreases. And this way, the diagnostic engine skips states of the system that are impossible. This
Model-Based Fault Diagnosis

4.6 MBD on the Power Supply

<table>
<thead>
<tr>
<th>Y</th>
<th>CHIL</th>
<th>FD</th>
<th>TBCB</th>
<th>CRCB</th>
<th>COL</th>
<th>Observations</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>None: entire system healthy</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>FuseD, LV_PS1, Collimator</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>FuseC, CRCB</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>None: only multiple faults</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>FuseB, TBCB</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>None: only multiple faults</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>LV_PS2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CableB</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>FuseA, CableA, LV_PS1, Flat Detector</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>None: only multiple faults</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>None: only multiple faults</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>None: only multiple faults</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>None: only multiple faults</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>None: only multiple faults</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>None: only multiple faults</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Chameleon</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PDU</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Partial LUT for a strong model of the Power Supply system for single faults (startup = true)

improves speed of diagnosis and diagnostic resolution. The entropy is lower for a strong model, so constructing a strong model results in entropy gain.

The benefits of applying the model-based approach at the power supply example is a speed of diagnoses that decreases from an hour to a few milliseconds.
Chapter 5

Diagnosing the Beam Propeller Movement of the Frontal Stand

The previous chapter presented the basic theory of model-based fault diagnosis. It already included an explanation of MBD on a real subsystem of the Philips Cardio-Vascular X-Ray System. This trivial example already shows benefits, but the main part of the Philips Cardio-Vascular X-Ray System has more complexity. Therefore, this chapter presents a case study of model-based diagnosis on a much more complex subsystem.

In this case study, a MBD implementation for diagnosing failures of the beam propeller movement is developed. The LYDIA tool set is used as a diagnostic engine, and the LYDIA language is used for specifying the model. The first section of this chapter describes the experimental methodology and the approach to the case study. Section 5.2 describes the scope of the target system. The third section presents the fault diagnosis that experts are able to do using log data. Then, the three remaining sections deal with the model-based approach to fault diagnosis. Each of the sections presents a MBD implementation, and estimates its diagnostic performance in terms of entropy. Section 5.4 presents a MBD implementation (MBD-1) that achieves equal entropy as experts are able to achieve. Section 5.5 presents a MBD implementation (MBD-2), that uses an improved model on the same system data, and calculates its entropy gain. The final section of this chapter presents which potential MBD implementation (MBD-3) could achieve the highest entropy gain, by considering various extra measurement points.

5.1 Experimental Methodology

Recall that the problem encountered in the work presented in this thesis is to improve fault diagnosis of Philips Cardio-Vascular X-Ray Systems with respect to higher dependability. This case study presents a proof-of-concept of the model-based approach to fault diagnosis, aimed at Philips Cardio-Vascular X-Ray Systems. The goal of this case study is to study the applicability of MBD on an example subsystem of the Philips Cardio-Vascular X-Ray System, namely the beam propeller movement of the frontal stand. The best way to do this is by using a metric for accuracy. Accuracy is generally considered as the most important criterion for diagnostic performance.

MBD produces a list of diagnoses, in order of probability. If a supervisory user would like to use the result of MBD for its recovery actions, only one of the solutions can be accepted. The first
5.2 Scope of the Target System

Diagnosing the Beam Propeller Movement of the Frontal Stand

item of the list is the diagnosis candidate that has highest probability, and is the most obvious choice of a supervisory user. A diagnosis is accurate if the first item of the list agrees with reality. The metric for accuracy is defined as follows. Given a set of fault scenarios, the diagnostic accuracy of a MBD implementation is defined as the ratio of the number of accurate diagnosed failures and the total number of diagnosed failures. This metric for accuracy is formally defined as follows: Let \( S \) be a list of \( n \) fault scenarios. Let \( j \) refer to the \( j^{th} \) item of list \( S \). Then, \( D_j \) is defined as the list of diagnoses produced by the diagnostic engine for fault scenario \( j \). The accepted outcome of the diagnostic engine, for fault scenario \( j \), is defined as the first item of the list \( D_j \), and denoted by \( d_{MBD}^j \). Finally, let \( d_{real}^j \) be the adjudged broken component. The accuracy (\( A \)) of a MBD implementation is then calculated by the formula:

\[
A = \frac{\sum (d_{MBD}^j = d_{real}^j)}{n}
\]  

An alternative metric for diagnostic accuracy, that is used in a case study at ASML [5], is not chosen to apply on this case study. In this approach there are two models of the system. One is synthesized to a simulation engine of the system, and the other to a diagnostic engine for diagnosing the simulated system. The advantage of this metric is that physical use of the real system is not necessary. The disadvantage is that building the model aimed at simulation requires much behavioral information. The simulation model that should be developed for this case study would be too complex, and therefore this accuracy metric could not be used.

5.1.1 Approach

As explained in the previous chapter, developing a MBD implementation requires the construction of a model. The approach to modeling that is taken uses log data as a starting point (see Section 4.4). Modeling is not a trivial activity, and for this reason a feedback mechanism for improving the solution is useful. Entropy is this feedback mechanism. The approach that is used for this case study is:

1. Define the scope of the target system, based on the system data of a set of fault scenarios.
2. Experts interpret system data, and assign a list of possible diagnoses to each fault scenario.
3. The modeler specifies a consistency-based LYDIA model, that if inserted to the LYDIA diagnostic engine yields the same results as (2), for each fault scenario.
4. Calculate the entropy of the model.
5. Repeat steps 2, 3, and 4 until no more entropy gain could be achieved.

Step 1 is described by Section 5.2. Section 5.3 presents step 2. Section 5.4 presents step 3, and 4. Section 5.5 shows another iteration of step 2, 3, and 4, resulting in an improved version of the model. Section 5.6 presents the other way to entropy gain; extending the observability of the system.

5.2 Scope of the Target System

This section defines the scope of the system that is subject to diagnosis. This is the first step of the approach. This section starts by giving some context of the case study; the function of the beam propeller movement. Then, the log entry that contains information about malfunctioning behavior is used to define the scope of the target system. This results in a collection of FRUs that are subject to the fault diagnosis process that is described in the remainder of this chapter.
Diagnosing the Beam Propeller Movement of the Frontal Stand

5.2 Scope of the Target System

---

* 04/29/2005 12:41:25 CLARIFY (CLARIFY)* Clarify Case:
1593793* Material #: 72246 Serial #: 26204* Unknown Contact:
********** FE, FSE** Customer Information:* NEW YORK METHODIST
HOSPITAL ATT: ACCOUNTS PAYABLE* 506 SIXTH ST* BROOKLYN, NY
11215 US* Phone: 2350** *** PHONE LOG 04/29/2005 03:40:52 PM
ReneeL* Q/O.... RESTORATION OF JUNCTION MOVEMENT*** *** CASE
CLOSE 04/29/2005 03:40:56 PM ReneeL* Q/O....RESTORATION OF
JUNCTION MOVEMENT** *** SUBCASE 1593793-1 CLOSED 04/29/2005
03:41:18 PM ReneeL*CA(00009021),05/16/2005: Assisted
********** with removal and reinstallation junction motor and
gear box. And calibration and alignment. Plus total replacement of
Poly G power supply. *; *;*;*CA(2350),05/18/2005: performed
total restoration of junction rotation installed new motor,
gear box, brake and belt mechanism replaced control ckt incl LUC
and Extension, twin MVR high power and power supply calibration
complete tested ok*;

---

Figure 5.1: Jobsheet referring to problems with the beam propeller movement.

5.2.1 The Beam Propeller Movement function

When the doctor is diagnosing a patient he likes to make images of different parts of the patient's heart and vascular veins and under various angles. In order to do so he is able to move the table and frontal stand in various angels of freedom. There is a subsystem defined, called Geometry, that is responsible for such mechanical movements within the Philips Cardio-Vascular X-Ray System. It consists of mechanics as well as the related hardware and software components. Of course, these mechanical movements are subject to failures.

The text of Figure 5.1 shows a job sheet that is written by a service engineer faced with such a failure. Without experience, it is quite hard to read, but if you take a close look on this data you can see it is referring to the beam propeller movement\(^1\). It seems that the movement has caused quite some trouble: all FRUs that seem to have something to do with the movement have been replaced. However, it is very unlikely that all these FRUs were malfunctioning. Why did the service engineer replace all parts, anyway? Probably he did not know what was wrong and could not afford the risk of leaving the operator with any more failures. The remainder of this chapter clarifies how MBD can help.

Of all movements the beam propeller movement is chosen because its FRUs are accessible quite easily\(^2\). The beam propeller movement function enables the operator to rotate the frontal stand. By means of some kind of peripheral device - usually a joystick - the user requests a speed in a certain direction. The system does whatever it needs to do in order to make the frontal stand rotate at the requested speed, see Figure 5.2. Most of the time this is sufficient, however there are occasions in which the operator faces difficulties. In that case, a failure has occurred related to the beam propeller movement. Because this part of the medical system is safety critical, developers have thought of an error detection mechanism in order to prevent the moving parts from doing any harm. If the

\(^1\)“Junction movement” is the old term within PMS for “Beam propeller movement”.

\(^2\)Remark: An automated approach has even more benefits on a system when the components are hard to reach, because the diagnostic process does not need to access the parts physically. However, this pilot study prefers a more comfortable set-up.
Diagnosing the Beam Propeller Movement of the Frontal Stand

5.2 Scope of the Target System

Figure 5.2: Beam Propeller Movement of the frontal stand

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefectNr</td>
<td>63000011</td>
</tr>
<tr>
<td>Description</td>
<td>Application error: Error on Junction Movement</td>
</tr>
<tr>
<td>Type</td>
<td>Error</td>
</tr>
<tr>
<td>Unit</td>
<td>Geometry</td>
</tr>
<tr>
<td>Count</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>-</td>
</tr>
<tr>
<td>Time</td>
<td>17:38:16</td>
</tr>
</tbody>
</table>

Table 5.1: Example of a log entry (when error 11 occurs).

The error detection mechanism detects an error, appropriate recovery action are invoked, and an error is logged. The following section describes the error that is logged by the error detection mechanism.

5.2.2 The Error of the Beam Propeller Movement

The error that is logged by the error detection mechanism contains information about the state of the system, at the moment the failure occurred. The target system only includes those components that this state refers to.

Table 5.1 shows an example of the error message. Such a message is logged each time one of the Geometry’s mechanical movements malfunctions. Among the Geometry developers it is known as error 11. A log entry typically contains diagnostic information about a part of the system. In this case, an occurrence of error 11 pinpoints to a failure of one of the mechanical movements within the Geometry subsystem. The field Description of Table 5.1 indicates what movement’s error detection mechanism has detected the error: the beam propeller movement of the frontal stand. In addition to that, it would help if the error message contains information about the system’s state at the time the error occurred. In this case were are quite lucky: there is a Exception Description...
Diagnosing the Beam Propeller Movement
of the Frontal Stand

5.2 Scope of the Target System

part in the row Info that enumerates all kinds of parameters. These parameters all say something about the state of the system.

The scope of the target system depends on considerations, that are specific for PMS, this particular subsystem, and motives of the modeler. Not all components that could cause a failure of the beam propeller movement are included in the target system. The architecture of the system that contains components that are considered is described in Appendix B.1. The considerations to decide if components are included or excluded in the target system are as follows. For all known components the following questions are asked:

1. Could a PMS service engineer replace this component with a healthy one? In other words, is the component a FRU?
2. Does the malfunctioning of this component causes an error 11 message to be logged?
3. Could an expert think of any (hypothetically) produced error 11 log data, of which any conclusions about the component’s state could be derived?
4. Does including the component lead to superfluous complexity of the model?
5. Is the component part of a control loop?

If, for a certain component, all answers on these questions are positive, the component is included in the scope of the target system. The last consideration is special to this specific subsystem. The beam propeller movement is implemented by means of three nested control loops, respectively a position loop, speed loop, and current loop. These three control loops play an important role, and therefore this question is of importance (see Section 5.3).

Table 5.2 shows the components that are excluded (see Appendix B.1 for the place of a component within the architecture). The discussed fault diagnosis only addresses the components that are

<table>
<thead>
<tr>
<th>Component</th>
<th>Failing Consideration</th>
</tr>
</thead>
<tbody>
<tr>
<td>All components that propagate or influence (the cloud in Figure 5.2) the requests speed signal until it enters the backpanel control unit.</td>
<td>2</td>
</tr>
<tr>
<td>backpanel control unit</td>
<td>3</td>
</tr>
<tr>
<td>LUC</td>
<td>3</td>
</tr>
<tr>
<td>Power Supply</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 5.2: Components that are excluded from the target system.

part of the subsystem Geometry. All components of other subsystems that might lead to a failure of the beam propeller movement do not cause an error 11 message to be logged. The backpanel control unit and the LUC are not included. The reason for this is that in the current technical set-up it is impossible to differentiate if faults are activated by the backpanel control unit, LUC or its corresponding LUC_Extension. Because the LUC_Extension has, by far, the highest a priori failure probability, it has been chosen to include this component in the target system. The power supply is not included, because adding it leads to superfluous complexity of the model. Note that the fault diagnosis process proposed below could be extended to enable the diagnosis of these, currently, omitted FRUs as well.

Table 5.3 names the set of components that are within the scope of the diagnostic process. Each of them has been given an a priori component failure. These values only determine the order of
the list of diagnoses that the MBD implementations discussed in this chapter produce. The order of failure probabilities is determined by interviewing experts. The table also names the health variables that will be used in the LYDIA models presented in the following sections.

<table>
<thead>
<tr>
<th>FRU</th>
<th>Description</th>
<th>Health Variable</th>
<th>A-priori Failure Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUC Extension</td>
<td>The controller of the mechanical movement. It consists of three nested control loops (position, speed and current).</td>
<td>h_{EXT}</td>
<td>0.03</td>
</tr>
<tr>
<td>MVR</td>
<td>Controls the current and voltage towards the motor. MVR stands for Motor Voltage Regulator.</td>
<td>h_{MVR}</td>
<td>0.02</td>
</tr>
<tr>
<td>Motor/Brake</td>
<td>The motor and brake are combined in one FRU, namely the Motor/Brake unit (MBU). If the brake is released, the motor has to generate a torque in order to accelerate or slow down the stand’s rotating.</td>
<td>h_{MBU}</td>
<td>0.01</td>
</tr>
<tr>
<td>Stand</td>
<td>The mechanical body that is subject to the rotation.</td>
<td>h_{Stand}</td>
<td>0.05</td>
</tr>
<tr>
<td>Potmeter/Encoder</td>
<td>Both the potentiometer as the encoder are used as sensors to measure the position of the stand. For reasons of accuracy and error detection the values are combined.</td>
<td>h_{PEU}</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table 5.3: Components that are included in the target system, with their health variables, and associated a priori failure probabilities.

5.3  Off-line Inference by Experts

This section describes step 2 of the case study approach. This step requires that experts derive a LUT by human inference of system data. The motivation of this step is as follows. Failing to acquire relevant knowledge for constructing a consistency-based model is the most important show stopper for the success of the model-based approach. Abductive and deductive reasoning by experts provides an extra source of modeling information. The information that expert use for constructing the symptom-diagnosis mapping is mostly gained by experience, and hard to make explicit by interviews. This is why this modeling step is very useful for acquiring that 'hidden’ information, and use it for constructing a consistency-based model.

5.3.1  Fault Scenarios of the Beam Propeller Movement

Table 5.4 shows (part of) the fault scenarios that have been analyzed. These eleven fault scenarios, \( S1 \) till \( S11 \), occurred during system operation in hospitals. This means that the data origins from a real case setting. This is possible, because the shown system data could be extracted from the remote monitoring tools of PMS. Experts interpreted the system data of each fault scenario, and produced a list of possible diagnoses, ordered by probability. Table 5.4 shows the system data that experts examined in combination with the proposed diagnoses. The fault scenarios can be characterized

---

\(^3\)Table B.2 of Appendix B shows all fault scenarios, that are considered in the work of this thesis.
Diagnosing the Beam Propeller Movement of the Frontal Stand

5.3 Off-line Inference by Experts

by four fault categories. Fault category \( C1 \) is defined as all fault scenarios in which the error variable is current (from now referred to as \( \text{CURRENT\_ERROR} \)). It consists of fault scenarios \( S1 \) till \( S4 \). Fault category \( C2 \) is characterized by fault scenarios in which the error signal is speed (from now referred to as \( \text{SPEED\_ERROR} \)), and consists of fault scenarios \( S5 \) till \( S8 \). Fault category \( C3 \) is defined as fault scenarios in which the error variable is position (from now referred to as \( \text{POSITION\_ERROR} \)), and \( \text{Pact} \) and \( \text{Pset} \) do not differ more than a certain threshold. It consists of fault scenarios \( S9 \) and \( S10 \). Fault category \( C4 \) characterizes fault scenarios in which the error variable is pos. val (from now on referred to as \( \text{POSVAL\_ERROR} \)), for example fault scenario \( S11 \).

<table>
<thead>
<tr>
<th>FC</th>
<th>Signal</th>
<th>Imin</th>
<th>Imax</th>
<th>Iact</th>
<th>Iset</th>
<th>Vact</th>
<th>Vset</th>
<th>Pact</th>
<th>Pset</th>
<th>Error</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C1 )</td>
<td>( S1 )</td>
<td>–2070</td>
<td>6456</td>
<td>409</td>
<td>6135</td>
<td>14</td>
<td>29</td>
<td>444</td>
<td>443</td>
<td>current</td>
<td>current (</td>
</tr>
<tr>
<td>( C1 )</td>
<td>( S2 )</td>
<td>–8462</td>
<td>64</td>
<td>–214</td>
<td>–8724</td>
<td>–21</td>
<td>–41</td>
<td>109</td>
<td>107</td>
<td>current</td>
<td>current (</td>
</tr>
<tr>
<td>( C1 )</td>
<td>( S3 )</td>
<td>–4879</td>
<td>2001</td>
<td>43</td>
<td>–4390</td>
<td>50</td>
<td>58</td>
<td>137</td>
<td>132</td>
<td>current</td>
<td>current (</td>
</tr>
<tr>
<td>( C1 )</td>
<td>( S4 )</td>
<td>1673</td>
<td>8611</td>
<td>141</td>
<td>8614</td>
<td>–54</td>
<td>–39</td>
<td>289</td>
<td>288</td>
<td>current</td>
<td>current (</td>
</tr>
<tr>
<td>( C2 )</td>
<td>( S5 )</td>
<td>4316</td>
<td>12486</td>
<td>3584</td>
<td>3632</td>
<td>257</td>
<td>–155</td>
<td>324</td>
<td>295</td>
<td>speed</td>
<td>speed (</td>
</tr>
<tr>
<td>( C2 )</td>
<td>( S6 )</td>
<td>1548</td>
<td>9778</td>
<td>702</td>
<td>751</td>
<td>252</td>
<td>–53</td>
<td>–279</td>
<td>–300</td>
<td>speed</td>
<td>speed (</td>
</tr>
<tr>
<td>( C2 )</td>
<td>( S7 )</td>
<td>–13655</td>
<td>–5485</td>
<td>–4621</td>
<td>–4622</td>
<td>–260</td>
<td>302</td>
<td>784</td>
<td>819</td>
<td>speed</td>
<td>speed (</td>
</tr>
<tr>
<td>( C2 )</td>
<td>( S8 )</td>
<td>6089</td>
<td>14259</td>
<td>5183</td>
<td>5171</td>
<td>275</td>
<td>–253</td>
<td>102</td>
<td>66</td>
<td>speed</td>
<td>speed (</td>
</tr>
<tr>
<td>( C3 )</td>
<td>( S10 )</td>
<td>–3421</td>
<td>3217</td>
<td>–104</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>–37</td>
<td>–37</td>
<td>position</td>
<td>position (</td>
</tr>
<tr>
<td>( C4 )</td>
<td>( S11 )</td>
<td>4104</td>
<td>10984</td>
<td>7186</td>
<td>6868</td>
<td>–1</td>
<td>–45</td>
<td>17</td>
<td>12</td>
<td>pos. val</td>
<td>PEU</td>
</tr>
</tbody>
</table>

Table 5.4: Fault scenarios, and their adjudged diagnoses.

5.3.2 Symptoms of a Malfunctioning Beam Propeller Movement

This subsection explains and presents the symptoms of a malfunctioning beam propeller movement. To give the reader some understanding in the results shown in Table 5.4 some behavior of the target system is explained. This is necessary to understand why symptoms identify certain components as being at false. A thorough explaining of the system behavior is given in Appendix B.3. As said before, there are three nested control loops. Figure 5.3, that we constructed for the MBD approaches that are discussed in the remainder of this chapter, validates the diagnoses that are given for each malfunctioning control loop. The error detection mechanism is able to detect inconsistencies in each control loop, and generates a different error for each case:

1. If \( Iact \) and \( Iset \) differ more than a certain threshold, the \( \text{CURRENT\_ERROR} \) signal becomes true. The current loop consists of LUC Extension (controller), MVR (forward), Motor/Brake and MVR (feedback). So, these are the components that could be broken if the \( \text{CURRENT\_ERROR} \) signal (= true) indicates something has gone wrong in the control loop for current.

2. If \( Vact \) and \( Vset \) differ more than a certain threshold, the \( \text{SPEED\_ERROR} \) signal turns true. The speed loop consists of LUC Extension (controller), MVR (forward), Motor/Brake, Stand and MVR (feedback). So, these components are the suspicious components if \( \text{SPEED\_ERROR} \) is true.

3. If \( \text{Pact} \) and \( \text{Pset} \) differ more than a certain threshold, the \( \text{POSITION\_ERROR} \) becomes true. The position loop consists of LUC Extension (controller), MVR (forward), Motor/Brake,
5.3 Off-line Inference by Experts

Stand and Potmeter/Encoder and LUC_Extension (feedback). So, these components are possibly at false if POSITION_ERROR is true.

It is important to notice that, these three clarifying predictions of system behavior can only be made by means of such a concise representation of the system as Figure 5.3. Without it you would have to trust the expert's inference of system behavior. It is true that some experts know this information by head, and they use it to diagnose the fault scenarios. However, many times the information is not as comprehensive as Figure 5.3 shows. Moreover, an expert is unaware of using some information. Therefore, this knowledge is obtained by many interviews and by studying documentation [17, 16, 19].

The fault categories used in Table 5.4 can be explained by the introduced knowledge. Fault categories C1 and C2 describe resp. an unhealthy control loop and an unhealthy speed loop. The fault category C3 describes an occurrence of a POSITION_ERROR that seems to reveal an unhealthy position loop. However, the values of the Pact and Pset variables do not differ more than the threshold, and therefore the LUC_Extension falsely generates the POSITION_ERROR. It is possible that the error is caused by a malfunctioning PEU. From now on, the fact whether or not Pact and Pset differ more than the threshold is denoted by the variable e_pos. When e_pos is 1 the threshold is violated, otherwise e_pos is 0. Fault scenario C4 is an occurrence of a POSVAL_ERROR. This indicates that the PEU is broken. The symptoms described above can be described more formally as follows:

\[
\begin{align*}
\text{CURRENT_ERROR} & = 1 \lor \\
\text{SPEED_ERROR} & = 1 \lor \\
\text{POSITION_ERROR} & = 1 \land e_{\text{pos}} = 1 \lor \\
\text{POSITION_ERROR} & = 1 \lor \\
\text{POSVAL_ERROR} & = 1
\end{align*}
\]

The variables that are used in the specification of the symptoms are the observables on the system: CURRENT_ERROR, SPEED_ERROR, POSITION_ERROR, POSVAL_ERROR, and e_pos. The identification of the symptoms enables the construction of the table that defines the mapping of these symptoms on diagnoses. This table is presented in the next subsection.

5.3.3 Results of Off-line Inference by Experts

The LUT is constructed by manually performed deductive and abductive reasoning of all possible values of the observables. Table 5.5 shows the result. For each possible observation, the table defines a mapping on a set of consistent diagnoses. For presentation purposes, the table shows only single faults. The table could be implemented by some expert system. In such an automated fault diagnosis approach, the monitored system data is searched in the table, inserted into the expert system, and the expert system produces a list of possible diagnoses. However, in the approach of this case study, the constructed LUT is just an intermediate step; Table 5.5 is used for constructing a consistency-based model.

The entropy of the approach to fault diagnosis ‘off-line inference’ can be calculated, by using the complete (including multiple faults) LUT. This yields the following result for the entropy (H) that experts are able to achieve:

\[
H_{\text{expert}} = 0.9453
\]

The next section presents a MBD implementation, that achieves the same entropy.
Diagnosing the Beam Propeller Movement of the Frontal Stand

5.4 MBD Implementation 1

This section presents the third step of the case study approach. In this step, the first MBD implementation (MBD-1), that reproduces the results and entropy that experts are able to achieve, is presented. The first part of this section discusses the model. Then, the diagnosis and entropy are discussed.

5.4.1 The Modeled Information

In this subsection, the model of the first MBD implementation is constructed. As explained before, the LUT that experts derived is used as valuable information for constructing a consistency-based model. Other sources of information are tools and documentation that are developed and written at PMS. In summary, the following sources of information have been used for the construction of the model that is presented in this section:

- The result of abductive and deductive reasoning by experts, as presented by the LUT shown in Table 5.5.
- Documentation about the meaning of the system data (see Table B.1).
- Technical drawings.
- Fault Isolation Procedures.
- Some design documents, specific to the beam propeller movement [17].
- Interviews with experts on the system.

<table>
<thead>
<tr>
<th>Y</th>
<th>Observations</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>e-pos</td>
<td>P.V. POS. SP. CUR.</td>
</tr>
<tr>
<td>Y1</td>
<td>- 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>Y2</td>
<td>- 0 0 0 1</td>
<td></td>
</tr>
<tr>
<td>Y3</td>
<td>- 0 0 1 0</td>
<td></td>
</tr>
<tr>
<td>Y4</td>
<td>- 0 0 1 1</td>
<td></td>
</tr>
<tr>
<td>Y5</td>
<td>1 0 1 0 0</td>
<td></td>
</tr>
<tr>
<td>Y6</td>
<td>0 0 1 0 0</td>
<td></td>
</tr>
<tr>
<td>Y7</td>
<td>- 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>Y8</td>
<td>- 0 1 1 0</td>
<td></td>
</tr>
<tr>
<td>Y9</td>
<td>- 0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>Y10</td>
<td>- 1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>Y11</td>
<td>- 1 0 0 1</td>
<td></td>
</tr>
<tr>
<td>Y12</td>
<td>- 1 0 1 0</td>
<td></td>
</tr>
<tr>
<td>Y13</td>
<td>- 1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>Y14</td>
<td>- 1 1 0 0</td>
<td></td>
</tr>
<tr>
<td>Y15</td>
<td>- 1 1 0 1</td>
<td></td>
</tr>
<tr>
<td>Y16</td>
<td>- 1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>Y17</td>
<td>- 1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Partial diagnosis look-up table of the Beam Propeller Movement for single faults
Figure 5.3: Partial architecture of the FRUs that implement the beam propeller movement. Signals enclosed by brackets are not observable, the others are.
5.4.2 The Modeled Structure

The structure of the model is shown in Figure 5.3. In short, this block diagram describes the following. The input of the target system is the speed the user requests (\( V_{\text{user}} \)). The controller of the movement controls the output of the target system: the angular speed of the frontal stand (\( \text{speed} \)). It does so, by using three control loops. Note that the fault detection mechanism is part of the architecture. It constitutes of the \( \text{POSITION\_ERROR} \), \( \text{SPEED\_ERROR} \), \( \text{CURRENT\_ERROR} \) and \( \text{POSVAL\_ERROR} \) signals. These signals are also chosen as being outputs of the system. During nominal behavior of the system these four error signals are false (Appendix B.3). The \text{LYDIA} model that defines the nominal behavior is listed in Appendix C.3.5.

This model, that is part of the first MBD implementation, specifies the target system with a certain granularity. The chosen granularity is motivated by the following two considerations:

- the choice of PMS what components can be replaced with new FRUs, and which cannot. This is the reason that the potentiometer and encoder are combined to one entity (\text{PEU}). The motor and brake are also combined (\text{MBU}), because it is impossible to replace one without the other.

- the possibility to use system data to distinguish between faults. For example, a fault of the \text{Stand} could, on a lower level of abstraction, be caused by numerous anomalies: oil, a power from the environment, etc. In some cases, these ‘lower’ faults can be recovered by the service engineer, but the system data cannot reveal them. Thus, these low-level faults are not represented by health variables in the model.

The model listed in C.3.5 is not compositional. Thus, the structure is not explicitly modeled. This is a practical form of the model during development, but in order to allow long-term maintenance, a compositional model is much more appropriate. Appendix C.3.2 shows a compositional version of the model belonging to the MBD-1 implementation. In this model, each health variable belongs to separate component. Each component is modeled as a separate entity. The structure is defined by the top level entity of the model; \text{FS\_Beam\_Propeller\_Movement}. Each component defines part of the behavior. The behavior is discussed in the following subsection.

5.4.3 Behavior of the Model

A block diagram as shown in Figure 5.3 only shows structure, and only developers of the system are able to understand how these components emit the intended behavior. The components of the compositional model \text{LYDIA} model do specify how the components implement the beam propeller movement of the frontal stand (see Appendix B.3 for a description of nominal behavior in natural language). It is a weak fault model. None is said about what happens if, for example, the \text{MBU} malfunctions. To give insight in how the models specifies the nominal behavior of the system, the writing below describes two important building blocks of the model: a control loop and an error signal.

Modeling a Control Loop

A very interesting issue of the beam propeller movement subsystem is that it consists of control loops. Because of their recursive nature the model should embed state and time. However, these issues are still topic of ongoing research. Therefore, the text below suggests an abstraction of a control loop that does not depend on time and state. Figure 5.4 shows successively the block diagram of one, two and three control loops.
5.4 MBD Implementation 1

Diagnosing the Beam Propeller Movement of the Frontal Stand

Figure 5.4: Modeling the control loops

Figure 5.4(a) shows the basic concept (C = controller, S = system). Suppose the setpoint (A) is \( H \) and the actual value (B) is \( L \). The controller has to adjust its output in order to control B to \( H \). This requires integrating the output of the controller over time. After a certain number of values of B that are still \( L \) it would eventually turn \( H \). This stable moment at which the setpoint and actual value coincide is being described by the model. The corresponding Lydia code is:

\[
( h_c \land h_s ) \Rightarrow (B = A);
\]

So, modeling it this way means there has been made an assumption that the controller is at its setpoint. Consequently, the Lydia code states that if both the controller and system are functioning correctly the setpoint and actual value coincide.

Figure 5.4(b) shows how two nested control loops should be modeled, using the same assumption as in the case of the single control loop, namely the controller is in its stable state. The inner control loop is now part of the system being controlled by the second controller. Possibly the controlled system contains more elements than just the inner control loop. This is indicated by the block \( S_2 \). Again, the actual value \( Q \) equals the setpoint value \( P \) if the controller \( (C_2) \) is functioning
correctly, is at its setpoint and the system’s \((C1, S1, S2)\) behavior is conform the prediction. The Lydia code is now:

\[
( h_c2 \text{ and } h_c1 \text{ and } h_s1 \text{ and } h_s2 ) \Rightarrow (Q = P);
\]

In the same way Figure 5.4(c) shows how a third control loop can be nested over the other two. The reader is encouraged to see how the three control loops concept has been embedded in Figure 5.3. Again, a correct functioning controller \((C3)\) and system \((C2, C1, S1, S2 \text{ and } S3)\) implicate that the actual value \(Y\) equals the setpoint value \(X\). This can be described by the following Lydia code:

\[
( h_c3 \text{ and } h_c2 \text{ and } h_c1 \text{ and } h_s1 \text{ and } h_s2 \text{ and } h_s3 ) \Rightarrow (Y = X);
\]

**Modeling the Error Signal**

As said before there is an error detection mechanism in place inside the *beam propeller movement*. Three errors are related to the control loops. If the setpoint and actual value differ too much the error signal becomes true. Figure 5.5 shows the concept.

\[
\text{ERROR} = (A \neq B)
\]

The square indicates the operation that determines whether or not input values of the controller are valid.

![Figure 5.5: Modeling the error signal](image)

**5.4.4 Discretization**

The observables are CURRENT_ERROR, SPEED_ERROR, POSITION_ERROR, POSVAL_ERROR, and e_pos. The first four are already in the boolean domain. e_pos should be derived from the values of Pact and Pset. These two variables are integers with many possible values, and its derivative should be discretized. As said before, the only thing that is of interest about the position of the frontal stand is whether Pact and Pset do, or do not exceed their threshold. The discretization is defined as follows:

\[
e_{\text{pos}}(B) = \begin{cases} 
0, & \text{if } (Pset - Pact) \in [-\text{POS\_THRESHOLD}, \text{POS\_THRESHOLD}] \\
1, & \text{all other cases}
\end{cases}
\]
5.5 MBD Implementation 2

5.4.5 Results of the MBD-1 Implementation

The goal of this step was to define a MBD implementation to fault diagnosis that achieves the same entropy as experts are able to achieve. The LUT, that specifies the outcome of the LYDIA diagnostic engine for all possible observations, is exactly equal to Table 5.5. Consequently, the entropy (H) of the MBD implementation (MBD-1) is the same:

\[ H_{MBD-1} = 0.9453 \] (5.4)

5.5 MBD Implementation 2

This section presents a second iteration of steps 2, 3, and 4 of the case study approach. The intended result is that the uncertainty decreases without having to make additional measurements, but by improving the model. The idea is as follows. A good diagnostic approach takes advantage of every possible inference that could be drawn out of all available system data. Therefore, in this iteration, experts and modeler search for more symptoms within the examined system data.

If one tries to ‘discover’ more symptoms in the same system data, it is useful to discretize the system data. Most data of the fault scenarios shown in Table 5.4 is in the domain of integers. Humans experience difficulties to reason about faulty components using such a level of detail. As explained in the previous chapter, humans are better able to perform fault diagnosis using a domain with few members. The applied discretization is defined by source domain \( N \) and target domain \( M \).

The target domain \( M \) is defined as follows:

\[
M = \begin{cases} 
  TL, & \text{: Too Low. The threshold has been violated.} \\
  L, & \text{: Low. Negative polarity of the value.}\\
  N, & \text{: Nominal. Close to zero.}\\
  H, & \text{: High. Positive polarity of the value.}\\
  TH, & \text{: Too High. The threshold has been violated.}\\
\end{cases}
\]

The source domain is the domain of natural numbers \( N \). The target domain values can be chosen in many ways. In this case these particular values (\( L \) denotes negative polarity, \( N \) denotes close to zero, etc.) are meaningful when reasoning from first principles about the target system. The mapping of values in the source domain \( N \) to the target domain \( M \) is shown in Appendix B.5. Table 5.6 shows the discretized values of the observations. Note that the table shows other variables than shown in Table 5.4. Discretizing the variables \( \text{Pact} \) and \( \text{Pset} \) is not useful, because the system does not show different behavior for different angles of the frontal stand. The variables \( \text{Imin} \) and \( \text{Imax} \) are used to calculate the polarity of \( \text{Iset} \) and \( \text{Iact} \) (see Appendix B.5), and do not have value on their own.

The values of Table 5.6 are much easier to interpret than the raw system data of Table 5.4. Much of the discretized variables have values that experts expect (see Section 5.3). A current error implies that the \( \text{Iact} \) and \( \text{Iset} \) differ. A speed error implies that \( \text{Vact} \) and \( \text{Vset} \) differ. From system behavior, it is also expected that the value of \( \text{Iset} \) has the same polarity as the value of \( e_{sp} \). This does not hold for fault scenario 3, as can be seen in Table 5.6. In this fault scenario, the inequality of \( \text{Iset} \) and \( e_{sp} \) is a symptom of a malfunctioning system. Figure 5.3 shows that \( \text{CTR} \text{.speed} \) must be at false. This unit is of lower than the chosen granularity, and therefore the symptom identifies that the LUC_Extension is malfunctioning.

62
Table 5.6: Discretization of the system data of fault scenario 1 till 11 to domain M.

The specification of the new model should specify additional behavior of the LUC_Extension, so that the diagnostic engine is able to derive the new symptom-on-diagnosis mapping. The cause-to-effect relation that specifies the behavior is as follows:

\[ h_{\text{EXT}} \Rightarrow (I_{\text{set}} = e_{\text{sp}}) \]

There are various ways to fulfil the addition to the model. One of the possibilities is to define \( I_{\text{set}} \) and \( e_{\text{sp}} \) as observables in domain \( M \), and insert their value by an automated tool that performs the mapping of the system data to domain \( M \) values. This unnecessarily complicates the code of the model. A better solution is to add an observable variable \( \text{ctr\_speed} \), that is derived from the variables \( I_{\text{set}} \) and \( e_{\text{sp}} \). In this implementation, the model must also define how the value of \( \text{ctr\_speed} \) is derived from the system data:

```c
// definition derivative ctr_speed
ctr_speed = (I_{\text{set}} != e_{\text{sp}});
```

The tool responsible for applying the discretization from domain \( N \) values to domain \( M \) values, and to calculate the value of the \( \text{ctr\_speed} \) variable. When \( \text{ctr\_speed} = 1 \), the MBD engine derives the diagnosis that the LUC_Extension is at false.

### 5.5.1 Results of the MBD-2 Implementation

Table 5.7 shows the new LUT that the LYDIA diagnostic engine produces when all possible observations are inserted. It includes one extra observation, namely \( Y_{17} \).

The entropy \( H_{\text{MBD-2}} \) of the model described in this section is:

\[ H_{\text{MBD-2}} = 0.9453 \] (5.5)

Despite the fact that we expected to achieve an entropy gain, there is no entropy gain at all. The equation \( H_{\text{MBD-2}} = H_{\text{MBD-1}} \) holds for 20 significant digits. This seems quite peculiar. MBD-2 is able to diagnose the fault scenario \( S3 \) with hundred percent certainty. After all, the list of diagnoses that the LYDIA engine produces contains only 1 item, namely LUC_Extension. At first sight it seems strange that MBD-2 does not yield any entropy gain compared to MBD-1. However, the values of the observables for fault scenario \( S3 \) are very rare, and for both implementation the
Diagnosing the Beam Propeller Movement of the Frontal Stand

5.6 MBD Implementation 3

This section presents how to use entropy to determine which additional measurements on the system can increase the diagnostic accuracy the most. Adding measurements means increasing the observation space. The observation space can be divided in a spatial dimension and a temporal dimension [21]. The spatial dimension refers to the number of observation variables (sensors). The temporal dimension refers to the number of samples per observation variable in time. Here, the spatial dimension of the beam propeller movement system is examined.

Figure 5.3 shows all components, observables and internal variables with respect to the beam propeller movement. The observation space of the model used for implementation MBD-2 is determined by all permutations of the following observables: CURRENT_ERROR, SPEED_ERROR, e_pos, POSITION_ERROR, POSVAL_ERROR and ctr_speed. It is interesting to known what variables shown in the block diagram of Figure 5.3 are best to add to this observation space. In other words, an engineer is interested what measurements of the target system improve the accuracy of the MBD outcome the most. This is interesting because there are costs for each measurement that is added to the observation space. For example, adding the variable real_speed requires that a sensor is added to the target system, the sensor signal is being logged, possibly discretized, and finally inserted...
Diagnosing the Beam Propeller Movement of the Frontal Stand

5.6 MBD Implementation

into the diagnostic engine. Other variables do not require the addition of a sensor, but require the implementation of the discretizer defined in Appendix B.5 (e.g., \( I_{set}, V_{act} \), etc.). The MBD-3 implementation is shown in Appendix C.3.6, which lists a modified version of the model in which all variables of Figure 5.3 are declared, and part of a behavioral rule. This model declares, in addition to the six variables that were already declared as observables, 8 other variables as being observable, namely \( I_{mvr}, I_{to\_motor}, I_{from\_motor}, real\_speed, V_{act}, V_{set}, I_{act} \) and \( I_{set} \). These are chosen, because it is most reasonable that these internals can be measured by means of some sensor or discretizer. The total amount of possible sensor set-ups are all permutations of these 8 variables (e.g., \([I_{mvr}, I_{set}], [real\_speed], [I_{mvr}, I_{to\_motor}, I_{from\_motor}]\)).

In order to calculate the best possible sensor set-up, given that the 6 observables of MBD-2 are already defined, the values of these 6 variables should be known. These values are different for many fault scenarios, and different values imply other values for the entropy of a particular sensor set-up. To overcome this problem, the values of these 6 observables are given the observable values defined by the fault categories. For each fault category, the set of values for the 6 observables of MBD-2 is added to the model of Appendix C.3.6 (this version of the model shows the addition of the fault scenario C1-values). Then, the entropy gain for all possible sensor set-ups is calculated for that fault category. The entropy gain of a particular sensor set-up is obtained by weighting the outcome of a fault category by the number of real-case fault scenarios that are in that particular fault category. Ideally, the calculation considers all permutations of the 8 additional observables. Because there is no tool that automates this extensive search process available, 5 sensor set-ups are arbitrarily chosen. The outcome, of the examination that is presented below, is a list of these 5 sensor set-ups that is ordered by the ability to increase diagnostic accuracy of the target system.

Table 5.8 shows the entropy gain of the 5 chosen sensor set-ups in case a fault scenario in fault category C1 occurred. It shows that adding the three additional sensors, \( I_{mvr}, I_{to\_motor} \) and \( I_{from\_motor} \), have the highest entropy gain, for this particular fault category. It is also interesting to notice that adding a sensor for the \( real\_speed \) has equal entropy gain as adding the two sensors for \( I_{from\_motor} \) and \( V_{act} \). Table 5.9 shows the entropy gain of the same sensor set-ups in case a fault scenario in fault category C2 occurred. This time adding sensors for \( I_{mvr} \) and \( I_{set} \) has higher entropy gain than adding the three three sensors. Table 5.10 shows the entropy gain in case a fault scenario in fault category C3 occurred. Finally, Table 5.11 shows the entropy gain in case a fault scenario in fault category C4 occurred.

<table>
<thead>
<tr>
<th>Additional measurements</th>
<th>Entropy gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_{mvr}, I_{to_motor}, I_{from_motor} )</td>
<td>0.2945</td>
</tr>
<tr>
<td>( I_{mvr}, I_{set} )</td>
<td>0.2589</td>
</tr>
<tr>
<td>( I_{from_motor}, V_{act} )</td>
<td>0.2407</td>
</tr>
<tr>
<td>real_speed</td>
<td>0.2407</td>
</tr>
<tr>
<td>( I_{from_motor} )</td>
<td>0.2407</td>
</tr>
</tbody>
</table>

Table 5.8: Some additional measurements for fault scenarios in fault category C1 (current error)

The entropy gain of a sensor set-up is calculated by weighting the entropy gains of the Tables 5.8, 5.9, 5.10 and 5.11 by the number of times a fault scenario from that category occurred in a real-case setting. From Table B.2 the number of occurrences of each fault category (for S1-S11 and S17-S42) is calculated:

- **C1:** 16 (\( CURRENT\_ERROR = true \))
- **C2:** 7 (\( SPEED\_ERROR = true \))
5.7 Results of the Case Study

Diagnosing the Beam Propeller Movement of the Frontal Stand

- C3: 12 (POSITION_ERROR = true and e_pos = false)
- C4: 1 (POSVAL_ERROR = true)
- Unknown fault category: 1 (error is status).

Table 5.12 shows the entropy gain after weighting for each fault category. For example, the entropy gain of the sensor set-up \([I_{mvr}, I_{to\_motor}, I_{from\_motor}]\) is calculated by:

\[
H_{gain} = \frac{16 \times 0.2945 + 7 \times 0.3729 + 12 \times 0.2711 + 1 \times 0.1883}{36} = 0.2990
\] (5.6)

The conclusion is that from the considered sensor set-ups, \([I_{mvr}, I_{to\_motor}, I_{from\_motor}]\) has the highest entropy gain, thus leads to the highest increase in diagnostic accuracy. Note that the decision to choose a particular sensor set-up also depends on the costs of adding those specific sensors. An expert would be unable to order a list of sensor set-ups with respect to higher diagnostic performance. For example, it is very likely that an expert expects higher improve of accuracy from the sensor set-up \([I_{from\_motor}, Vact]\) than from sensor set-up \([I_{mvr}, I_{set}]\), while this is the other way around. Ordering a list of sensor set-ups is especially complicating if one likes to consider all additional sensor set-ups that are possible.

5.7 Results of the Case Study

This section discusses why it is not possible to present the experimental results of the case study. The reason for this is as follows. The experimental methodology requires that for each fault scenario it is known what is broken. Unfortunately, for fault scenario’s S1 till S11 and S17 till S39 this is not possible\(^4\). These fault scenarios are extracted from the remote monitoring tools, and it is impossible to determine what components were really broken for those cases. Thus, it is impossible to calculate the accuracy of the MBD implementations, based on real-life fault scenarios.

In order to still be able to calculate the accuracy of the MBD implementations, 5 faults are injected in the system in a test situation. This resulted in 5 error messages. Table 5.13 shows the observations, adjudged broken components, and the outcome of MBD-1 (or MBD-2), for these 5 fault scenarios; S12 till S16. A value of 1 in column A of Table 5.13 means the fault scenario is accurately diagnosed. The vector that was used to denote the observations is \((e\_pos, control\_speed, POSVAL\_ERROR, POSITION\_ERROR, SPEED\_ERROR, CURRENT\_ERROR)\).

Using Formula 5.1, the accuracy of the MBD-1 and MBD-2 implementations over these 5 fault scenarios is calculated:

\[A_{\text{MBD}\_1} = A_{\text{MBD}\_2} = \frac{3}{5} = 60\%\] (5.7)

Of course, this result should be looked at with care. Only 5 fault scenarios could be used, and these did not occur in a real case setting. If the set of fault scenarios would be representative for a large group of real-life fault scenarios, an accuracy of 60% could be interpreted as that the MBD engine produces the actual diagnosis as the first item in the list for 60% of the fault scenarios. For the other 40% of the fault scenarios, the actual broken component is in the list, but not as the first item. For example, the outcome of the LYDIA diagnostic engine for fault scenario S12 lists the actual broken component as the fourth item. And the outcome of the LYDIA diagnostic engine for fault scenario S16 lists the actual candidate as the third item.

\(^4\) Appendix B.4 lists all fault scenarios.

66
### 5.7 Results of the Case Study

#### Diagnosing the Beam Propeller Movement of the Frontal Stand

<table>
<thead>
<tr>
<th>Additional measurements</th>
<th>Entropy gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_{mvr}, I_{to_motor}, I_{from_motor} )</td>
<td>0.3729</td>
</tr>
<tr>
<td>( I_{mvr}, I_{set} )</td>
<td>0.3755</td>
</tr>
<tr>
<td>( I_{from_motor}, V_{act} )</td>
<td>0.3431</td>
</tr>
<tr>
<td>( \text{real_speed} )</td>
<td>0.3431</td>
</tr>
<tr>
<td>( I_{from_motor} )</td>
<td>0.3431</td>
</tr>
</tbody>
</table>

Table 5.9: Some additional measurements for fault scenarios in fault category C2 (speed error)

<table>
<thead>
<tr>
<th>Additional measurements</th>
<th>Entropy gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_{mvr}, I_{to_motor}, I_{from_motor} )</td>
<td>0.2711</td>
</tr>
<tr>
<td>( I_{mvr}, I_{set} )</td>
<td>0.2877</td>
</tr>
<tr>
<td>( I_{from_motor}, V_{act} )</td>
<td>0.2657</td>
</tr>
<tr>
<td>( \text{real_speed} )</td>
<td>0.2657</td>
</tr>
<tr>
<td>( I_{from_motor} )</td>
<td>0.2657</td>
</tr>
</tbody>
</table>

Table 5.10: Some additional measurements for fault scenarios in fault category C3 (unjustified position error)

<table>
<thead>
<tr>
<th>Additional measurements</th>
<th>Entropy gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_{mvr}, I_{to_motor}, I_{from_motor} )</td>
<td>0.1883</td>
</tr>
<tr>
<td>( I_{mvr}, I_{set} )</td>
<td>0.1890</td>
</tr>
<tr>
<td>( I_{from_motor}, V_{act} )</td>
<td>0.1858</td>
</tr>
<tr>
<td>( \text{real_speed} )</td>
<td>0.1858</td>
</tr>
<tr>
<td>( I_{from_motor} )</td>
<td>0.1858</td>
</tr>
</tbody>
</table>

Table 5.11: Some additional measurements for fault scenarios in fault category C4 (pos. val. error)

<table>
<thead>
<tr>
<th>Additional measurements</th>
<th>Entropy gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_{mvr}, I_{to_motor}, I_{from_motor} )</td>
<td>0.2990</td>
</tr>
<tr>
<td>( I_{mvr}, I_{set} )</td>
<td>0.2892</td>
</tr>
<tr>
<td>( I_{from_motor}, V_{act} )</td>
<td>0.2674</td>
</tr>
<tr>
<td>( \text{real_speed} )</td>
<td>0.2674</td>
</tr>
<tr>
<td>( I_{from_motor} )</td>
<td>0.2674</td>
</tr>
</tbody>
</table>

Table 5.12: Entropy gain of 5 arbitrarily chosen additional sensor set-ups.
5.7 Results of the Case Study

Diagnosing the Beam Propeller Movement of the Frontal Stand

### Table 5.13: 5 additional fault scenarios, of which adjudged broken components are known.

<table>
<thead>
<tr>
<th>S</th>
<th>Observations</th>
<th>$D_{real}$</th>
<th>$D_{MBD-2}$</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>S12</td>
<td>(0,0,0,1,0,0)</td>
<td>$\neg h_{MBU}$</td>
<td>$\neg h_{Stand}$</td>
<td>0</td>
</tr>
<tr>
<td>S13</td>
<td>(0,0,1,0,0,0)</td>
<td>$\neg h_{PEU}$</td>
<td>$\neg h_{PEU}$</td>
<td>1</td>
</tr>
<tr>
<td>S14</td>
<td>(0,0,1,0,0,0)</td>
<td>$\neg h_{PEU}$</td>
<td>$\neg h_{PEU}$</td>
<td>1</td>
</tr>
<tr>
<td>S15</td>
<td>(0,0,0,1,0,0)</td>
<td>$\neg h_{Stand}$</td>
<td>$\neg h_{Stand}$</td>
<td>1</td>
</tr>
<tr>
<td>S16</td>
<td>(0,0,0,0,0,1)</td>
<td>$\neg h_{MBU}$</td>
<td>$\neg h_{Stand}$</td>
<td>0</td>
</tr>
</tbody>
</table>

Suppose this value is considered too low by the management of PMS. What can be done to increase it? This could be done by improving models, or extending observability. Entropy can be used to estimate the diagnostic performance of the fault scenarios. Even if no fault scenarios are available, entropy can be used to predict if the accuracy increases on future fault scenarios. The relation between entropy and accuracy is that solutions with lower entropy have equal or higher accuracy on the same set of fault scenarios [20].
Chapter 6

Conclusions and Recommendations

The complexity crisis is most visible in complex, multi-disciplinary embedded systems. Although individual components might work well separately, the system as a whole may exhibit unexpected faults. More and more components are bought third party and might not be used as the supplier expected when developing them. There is a complex dynamic interaction between subsystems, and this is what makes fault diagnosis difficult and time consuming. The Philips Cardio-Vascular X-Ray system is a complex, multi-disciplinary embedded system. The current efforts for doing fault diagnosis at PMS is increasing; the portion of the PMS employees working on service instead of development grows. This thesis presents possible improvement when using a model-based approach to fault diagnosis.

6.1 Conclusions

In this thesis, it is shown that the current practice to fault diagnosis at PMS is not optimal with respect to the items that an ideal fault diagnosis technique would have; accuracy, speed of diagnosis, uncertainty of diagnosis, context independency, development costs, runtime costs, explanation facility, and adaptability. Furthermore, model-based diagnosis is chosen as the automated solution that could improve fault diagnosis, because this technique to automated fault diagnosis is likely to possess these items that, when present in fault diagnosis, improve dependability of the target system.

Unfortunately, it is not possible to conclude that MBD improves fault diagnosis with respect to higher dependability of Philips Cardio-Vascular X-Ray Systems. In order to so, it must be possible to determine the accuracy of both the current practice to fault diagnosis, and the proposed technique MBD. This cannot be done. The reason for this is that it is impossible to find out what faults caused a certain failure. Although, in many cases, it is possible to determine the observations at the moment of failure (the error detection mechanisms in the Philips Cardio-Vascular X-Ray System produce clear log entries), it is impossible to determine the actual fault that caused the specific observable values. Without this information, the accuracy of the current practice, and a MBD implementation, cannot be quantified. In other words, the significant problem is the inability to determine what components recovered a certain failure.

Entropy is a valuable metric to show the improvement of different MBD implementations. In the case study, the diagnostic performance of three MBD implementations that can diagnose the beam propeller of the frontal stand is estimated. The entropy gain of implementation MBD-2 compared to MBD-1 is zero, thus implementing MBD-2 is not useful. Adding extra sensors to the target system
(MBD-2) does lead to entropy gain, and might be worth the costs of placing these sensors. Entropy can be used to decide which additional measurements are best.

6.2 Recommendations

The following recommendations help in establishing a fault diagnosis approach at PMS, that achieves higher dependability of Philips Cardio-Vascular Systems. Appendix D shows the ideal practice of a fault diagnosis process at PMS.

- Currently, it is impossible to determine what fault(s) caused a certain failure. This must be solved, in order to evaluate the diagnostic performance of the current practice, and enable comparison with new approaches to fault diagnosis, such as MBD. The observations when a failure occurred are known and stored in the logs of each system, and by means of remote monitoring in a central database. The problem could be solved when Service Engineers insert their diagnosis of a failure, and what it did to recover the failure, into the central database. A tool must use the time of this insertion, and knowledge about which log entries contain observations about the failure, to link the diagnosis to the observations. For example, suppose a service engineers diagnoses a failure of the beam propeller movement of the frontal stand, namely fault scenario S13 (see Table 5.13). The database contains an ‘error 11’ log entry. In the proposed situation, the service engineer records the failure ‘beam propeller movement of the frontal stand’, the diagnosis that MBU is at fault, the replacement of the MBU, and the time of these actions. If the failure does not reoccur within short time, the automated tool concludes that the observables of the error 11 entry, with a time stamp close to the entry inserted by the service engineer, recovers the failure. Consequently, the central database must save the link between both the ‘error 11’ entry and the entry inserted by the service engineer. This enables PMS to know what fault caused the failure.

- Entropy is an important metric when developing MBD implementations. Its use as a feedback mechanism for constructing an optimal model, and optimal measurement points is very valuable. Research on how to achieve the highest entropy gain in fault diagnosis of the Philips Cardio-Vascular X-Ray Systems is valuable for research as well as business. Research is interested in the use of entropy in an industrial domain. Business could use entropy of diagnoses for developing very dependable medical systems, or in general, embedded systems.

- Currently, the log entries of the Philips Cardio-Vascular X-Ray System logs do not contain sufficient data that can be used as observables in a MBD implementation. MBD and entropy are suitable mechanisms to decide which variables should be added to the log, or which additional sensors lead to the highest improvement of diagnostic performance. The approach for achieving this consists of the following steps: construct the model of a subsystem. Perform an entropy study to decide which measurements are best. If necessary, add sensors to the target system. Implement the mechanism to log the chosen variables. During runtime operation, the model and the logged observables must be fed to the MBD engine, resulting in a higher diagnostic performance.

- The case study examined fault scenarios in which only one component was at false (single fault). Fault scenarios in which a failure is caused by multiple faults are likely to occur in reality, and implementing a MBD implementation that is able to accurately diagnose such fault scenarios could lead to higher dependability of the target system.
Conclusions and Recommendations 6.2 Recommendations

- In the case study, it is assumed that faults occur independently of each other. In real-case fault scenarios, it is common that faults are dependent. In other words, one fault is caused by other faults. Identifying the root cause of dependent faults increases a dependable operation of the target system. Therefore, it is valuable to examine MBD implementations for such fault scenarios.

- Practice at PMS shows that the number of faults and failures is highest just after a new system release. A MBD implementation that identifies faults in the implementation, that did not occur until that point scores high on the item ‘novel identifiability’. A MBD model is able to model fault modes that did not occur yet, but is is essential that the model is still robust. Consequently, a research on novel identifiability and the robustness of MBD models is valuable for PMS, and industry in general.

- Currently, for many cases, the time to diagnose is long because it is not known if the cause of a failure is caused by hardware or software. Therefore, it is valuable to examine the benefits of MBD in such a case.

- The storage and computational requirements to calculate the entropy of models, and entropy gain of additional measurements, is very high for most models. Therefore, it is useful to examine implementations of the diagnostic engine that approximate entropy in less time and space.
Bibliography


BIBLIOGRAPHY


Appendix A

Glossary of terms

In this appendix we give an overview of frequently used terms and abbreviations.

• **Abduction**: reasoning based on the principle of 'inference to the best explanation'.

• **Abductive approach**: an approach to fault diagnosis that defines a diagnosis as a set of abnormality assumptions that covers (or, in terms of logic, implies) the observations. [18, 24]

• **Abductive Model**: see *abductive approach*. It is the part of the *strong model* that defines the false modes of operation.

• **Accuracy**: the extent to which *diagnoses* agree with reality. Used as a criterion of *diagnostic performance* in this thesis.

• **Adaptability**: a criterion of *diagnostic performance* that is used in this thesis, namely the ability of a diagnostic process to cope with design changes of the *target system*.

• **A priori probability**: the probability that a component fails ($h=0$), without having made any observations.

• **Arm**: term of mechanics that is defines as the distance between the centre of buoyancy and the the lever’s fulcrum.

• **Automated Fault Diagnosis**: approaches to *fault diagnosis*, in which the interpretation of system data to produce diagnoses is automated.

• **Backpanel Control Unit**: a FRU of the Philips C/V X-Ray system that is comparable with the motherboard of a PC; by means of *PCI* it connects various hardware units.

• **Beam propeller movement**: a mechanical (rotational) movement of a *stand* of the Philips Cardio-Vascular X-Ray System.

• **Behavior**: what the system does to implement its function, and is described by a sequence of states [4].

• **Black Box**: A model that describes the externally observable behavior, but does not state anything about the structure of the system or behavior of internal components of the system.

• **Build-In-Self-Test (BIST)**: a test of one or more FRUs that can be performed in Field Service mode.
Glossary of terms

- **Cardio-Vascular**: the business unit within PMS responsible for the development, sales and service of Philips Cardio-Vascular X-Ray Systems.

- **Cardio-Vascular X-Ray System**: system is used to enable diagnosis and treatment of patients with cardiac and vascular diseases by using the X-ray imaging technique.

- **Cause-to-Effect Reasoning**: see *consistency-based approach*.

- **Chameleon**: a FRU that is used in the power supply example; it needs voltage.

- **Chiller**: a FRU that is used in the power supply example; it needs voltage.

- **Collimator**: a device made from radiation absorbent material such as lead or tungsten, designed to limit and define the direction and angular divergence of the radiation beam. This FRU is part of the environment for the *beam propeller movement*, and used in the power supply example; it needs voltage.

- **Component**: one element of a larger system.

- **Consistency-based Approach**: an approach to fault diagnosis that defines a diagnosis as a set of assumptions about a system component’s abnormal behavior such that observations of one component’s misbehavior are consistent with the assumption that all the other components are acting correctly [11, 25].

- **Consistency-based Model**: see *Consistency-based Approach*.

- **CRCB**: a FRU that is used in the power supply example; it needs voltage.

- **Customer downtime**: see *repair time*.

- **C/V**: See *Cardio-Vascular*.

- **Data Mining**: automatically searching large stores of data for correlations between variables.

- **Deduction**: inference in which the conclusion is of no greater generality than the premises.

- **Dependability** (dependable): a property that a successful system must have. It can be decomposed into the more lower level attributes *availability, integrity, maintainability, reliability* and *safety* [4].

- **Development Costs**: a criterion of *diagnostic performance* that is used in this thesis, namely all the costs that have to made prior to the start of the diagnostic process (e.g., supporting artifacts, training sessions for troubleshooters, other prepare actions that precede the operational phase).

- **Diagnosis**: see *fault*.

- **Diagnostic Approach**: the specific characteristics of a fault diagnosis process (e.g., reasoning scheme, online or off-line reasoning, automated or manual inference, etc.).

- **Diagnostic Engine**: MBD engine that produces diagnoses based on a model and real-life observations.
Glossary of terms

• **Diagnostic Performance**: the quality of an approach to fault diagnosis in terms of dependability of the target system, costs, and context independency.

• **Diagnostic Resolution**: the extent to which a diagnosis process is able to minimize the set of suspicious components.

• **Diagnosis Process**: process that enables fault diagnosis.

• **Discretization**: a mapping of variables within the system data, that are in a many-valued domain, on observables, that are in a few-valued domain.

• **Effect-to-Cause Reasoning**: see abductive approach.

• **Entropy**: measure for uncertainty.

• **Entropy Gain**: decrease in uncertainty.

• **Error**: the deviation between external system state and correct system state [4].

• **Error 11**: A known error in the subsystem *Geometry* of the Philips C/V X-Ray System, that is logged in case one of the mechanical movements malfunctions.

• **Error Detection**: identifies the presence of error [4].

• **Error-on-Solution Database**: tool used for fault diagnosis at PMS; a database that contains error messages and their corresponding cause and solution.

• **Explanation Facility**: a criterion of diagnostic performance that is used in this thesis, namely the justification of a diagnoses.

• **Failure**: an event that occurs when the delivered service deviates from correct service [4].

• **Fault**: the adjudged or hypothesized cause of an error. [4].

• **Fault Category**: A set of fault scenarios that have the same values for (a subset of) the observables.

• **Fault Diagnosis**: the process of identifying the root cause(s) of a failure. According to [4]: fault diagnosis identifies and records the cause(s) of error(s), in terms of both location and type.

• **Fault Isolation Procedure (FIP)**: a tree-like graph that can used by a service engineer to repair a malfunctioning part of the system. (usually it is created by the department Service Innovation of PMS C/V). Also known as test tree.

• **Fault Tolerant**: a system property that is present in systems that remain operational in the presence of faults.

• **Fault Tree**: a tree-like graph that hierarchically subdivides a failure in its causes.

• **Fault Recovery**: transforms a system state that contains one or more errors and (possibly) faults into a state without detected errors and without faults that can be activated again [4].
Glossary of terms

- **Fault Scenario**: A particular failure of the system. In this thesis, it is characterized by the *system data* that is logged at the moment a failure occurs, in addition with the actual broken component.

- **Field Replaceable Unit (FRU)**: a component of a system that can be replaced by a substitute.

- **Field Service mode**: special use of the system, especially made for diagnosing it.

- **First Principles**: a set of cause-to-effect relations between system parameters.

- **Flat Detector (FD)**: a device to measure the presence and amount of x-ray.

- **Frontal Stand**: the mechanical body that has the *collimator* and *flat detector* on its far ends. It can be positioned relative to the patient in order to transmit X-ray through the patient.

- **Geometry**: a subsystem of the Philips Cardio-Vascular X-Ray System, as defined by [16], that is responsible for mechanical movements of the system. It consists of mechanics as well as the related hardware and software components.

- **Granularity**: level of abstraction and detail.

- **Health variable**: state of the system, either healthy or unhealthy.

- **Health vector**: vector with the *health variables* of all components in the *target system* as elements.

- **Human inference**: the activity of humans to use their knowledge about nominal and faulty system behavior to reason about possibly faulty components.

- **Job sheet**: a document type within PMS C/V in which a service engineer documents on the history of diagnosis and repair of a particular system.

- **Junction movement**: see *beam propeller movement*.

- **Log data (log/logging)**: *system data* in the form of a digital file that consists of data that a system produces during real-time operation.

- **LUC**: Local Universal Controller. This is dedicated hardware unit that implements the generic part of a controller for mechanical movements.

- **LUC Extension (Ext)**: Extension of a Local Universal Controller. This is a dedicated hardware unit that implements the part of the controller of a mechanical movement that is specific for that particular movement.

- **Look-Up Table (LUT)**: a table that defines a mapping between symptoms and consistent diagnoses.

- **LYDIA (Language for sYstem DIAgnosis)**: a language for specifying the model for *MBD*, developed at Delft University of Technology.

- **Mode of Operation**: a state of a component in which it obeys an unique behavioral rule.

- **Mode Catalog**: a table that specifies all possible observations for each *health vector*, and can be derived from a *consistency-based model* of a system.
Glossary of terms

- **Model**: an abstract representation of a system from a particular viewpoint, chosen by the modeler (in this thesis, the model can be both written down on some medium, or could remain only in the knowledge of an individual).

- **Model-Based Diagnosis (MBD)**: a reasoning technique to isolate root causes of failures, that uses a clearly separated model of a system.

- **Modality**: diagnostic equipment that uses a medical imaging technique, such as X-ray, Nuclear Medicine, ultrasound and MRI.

- **Motor/Brake Unit (MBU)**: FRU of Philips C/V X-Ray System. This component consists of a motor and a brake. The motor is the actuator of the mechanical movement of the frontal stand.

- **Motor Voltage Regulator (MVR)**: hardware unit that controls the current and voltage towards the motor.

- **Nominal behavior**: the intended behavior of the system.

- **Observability**: the extent to which internal states of a system can be inferred by knowledge of its external outputs.

- **Observable**: a property of the system that can be observed.

- **Online inference**: the MBD model is solved for \( h \), using realtime observations, at the moment fault diagnosis is performed.

- **Off-line inference**: the MBD model is solved for \( h \), using expected observations, priori to the moment that fault diagnosis is performed.

- **PCI**: Peripheral Component Interconnect. This is the most common I/O bus in use today. It provides a shared data path between a number of peripheral controllers in an embedded system.

- **Philips**: One of the world’s biggest electronics companies (Royal Philips Electronics).

- **Philips Cardio-Vascular X-Ray System**: the *Cardio-Vascular X-Ray System* that is developed and serviced by Philips Medical Systems.

- **Philips Medical Systems (PMS)**: a subsidiary of Philips that develops, sales and services a wide range of medical systems.

- **Power-On-Self-Test (POST)**: a test of one FRU that is automatically performed at startup.

- **Potentiometer**: component of the Philips C/V X-Ray System (not a FRU), used as a position sensor for various mechanical movements.

- **Potmeter/Encoder Unit (PEU)**: FRU of Philips C/V X-Ray System that consists of a Potentiometer and an encoder.

- **Power Distribution Unit (PDU)**: most important power supply of the Philips C/V X-Ray System. This FRU is used in the power supply example; it provides voltage.
• **Recovery**: see *Fault Recovery*.

• **Repair time**: the time between the moment of *failure* and the moment the *system is recovered*.

• **Remote Monitoring**: the extraction of system parameters (log data) from systems all over the world to a central database.

• **Robustness**: dependability with respect to external faults, which characterizes a system reaction to a specific class of faults [4].

• **Runtime Costs**: a criterion of *diagnostic performance* that is used in this thesis, namely all the costs that the company makes to keep the diagnostic process up and running.

• **Service**: in this thesis *service* refers to diagnosing and repairing of a system.

• **Service Delivery**: correct functioning of the system (no failures occurred), in which it performs its intended function.

• **Service Engineer**: the person that diagnoses and repairs systems. In this thesis the term refers to employees of PMS that diagnose and repair Philips Cardio-Vascular X-Ray Systems.

• **Service Innovation**: a department of PMS - C/V that is responsible to provide service engineers with information that is necessary for diagnosing Philips Cardio-Vascular X-Ray Systems.

• **Service Specialist**: a person that is specialized in the service of a system. In this thesis the term refers to employees of PMS that diagnose and repair the most rare failures of the Philips Cardio-Vascular X-Ray System.

• **ServiceWAX**: current name of the project with as goal to develop the remote monitoring tools.

• **Speed of diagnoses**: a criterion of *diagnostic performance* that is used in this thesis, namely the time between the moment that a failure occurs and the moment that the root cause of that failure is identified.

• **SPR**: a FRU of the Philips C/V System that contains the *MBU* and the *PEU*.

• **Stand**: the mechanical body that has the collimator and flat detector on its far ends. It can be positioned relative to the patient in order to transmit x-ray through the patient.

• **Strong model**: description of the system that defines all *modes of operation*.

• **Structure**: what enables the system to generate the behavior [4].

• **Subsystem**: a subset of interrelated *components*.

• **Supervisory Controller**: entity that uses a diagnosis to recover the system (e.g., replace a component, clean rotating parts, invoke redundant component, etc.).

• **Symptom**: see *error*.

• **Symptom-Cause-and-Solution sheets**: documents used for fault diagnosis at PMS; quick lookup sheets, each containing symptoms, and their corresponding cause and solution.
Glossary of terms

- **System**: an entity that interacts with other entities, i.e., other systems, including hardware, software, humans, and the physical world with its natural phenomena [4].

- **System data**: all unprocessed digital data that the target system produces.

- **Target System**: a system that is subject to fault diagnosis.

- **TBCB**: a FRU that is used in the power supply example; it needs voltage.

- **Technical Drawing**: drawings that define the structure of the Philips C/V X-Ray System at various levels of detail (e.g., of cable- and FRU-connections).

- **Troubleshooter**: a person that diagnoses and repairs systems (e.g., a service engineer).

- **Weak model**: a model that only defines the nominal behavior of the system, and does not state anything about false modes of operation.

- **White Box**: a model that describes the internally non-observable behavior as well as externally observable behavior. The behavior of the whole system is defined by the structure and the behavior of internal components.

- **X-ray**: a form of ionising radiation used to image some internal structures of the body.
Appendix B

Case Study Details

In this appendix details of the case study (Chap. 5) can be found. The outline of this appendix is as follows. Section B.1 describes the architecture of the most important parts of the beam propeller movement. Section B.2 lists some information that has been used to construct the model. Section B.3 explains the model of the beam propeller movement in natural language. Section B.4 lists all fault scenarios that could be extracted from the ServiceWax application, during the time span of the project. Section B.5 lists the discretization that has been used to examine further improvement of the model.

B.1 Architecture

The architecture that implements the beam propeller movement consists of three main parts. Figure B.1 shows the physical location of these parts. They comprise of the following:

Backpanel control unit: This is the place where the control software and hardware are located. It is comparable with the motherboard of a PC; by means of PCI it connects the hardware. In this case the hardware consists of dedicated hardware boards (the so-called LUCs and its extensions) and Motor Voltage Regulators (MVRs). The former are processing boards with hardware and embedded software. They come in pairs: when an extension is connected to

Figure B.1: The parts of the frontal stand
### B.2 System Data

#### Table B.1: The meaning of the system data that can be found in an error 11 entry.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imin</td>
<td>minimum motor current towards the hardware in milliampere</td>
</tr>
<tr>
<td>Imax</td>
<td>maximum motor current towards the hardware in milliampere.</td>
</tr>
<tr>
<td>Iact</td>
<td>actual motor current from the hardware in milliampere.</td>
</tr>
<tr>
<td>Iset</td>
<td>actual set motor current from the hardware in milliampere.</td>
</tr>
<tr>
<td>Vact</td>
<td>actual speed from the hardware derived by emk sensing of the motor in 0.1°/s or 0.1mm/s.</td>
</tr>
<tr>
<td>Vset</td>
<td>set speed towards the hardware in 0.1°/s or 0.1mm/s.</td>
</tr>
<tr>
<td>Pset</td>
<td>actual position from hardware in 0.1° or 0.1mm.</td>
</tr>
<tr>
<td>Pact</td>
<td>the set position of the motion controller in 0.1° or 0.1mm.</td>
</tr>
<tr>
<td>State</td>
<td>the movement state of the motion controller.</td>
</tr>
<tr>
<td>Note</td>
<td>indicates whether the units are 0.1° or mm.</td>
</tr>
<tr>
<td>Error(s) found</td>
<td>This is a readable representation of the error status register.</td>
</tr>
</tbody>
</table>

#### B.2 System Data

A LUC the pair is made specific for a certain task. Each pair forms the control unit of some movements. The MVR is part of the actuator path and is used to amplify the electronic signal from the processing boards to the motor and brake.

**SPR:** The unit that contains the *motor, brake, potentiometer* and *encoder*. The former two are the actuators of the movement. The latter two are used as sensors and constitute the start of the sensor feedback path of the control loop.

**Frontal Stand:** The mechanical body that has the collimator and flat detector on its far ends. It can be positioned relative to the patient in order to transmit X-ray through the patient.

### B.3 Behavior

Below follows a description of the behavior of the system in natural language. A formal specification is the corresponding Lydia code that is listed in appendix C. The desired behavior is realized by three nested control loops. The writing below follows the convention to use *this font* for all referenced names of signals and FRUs (also shown in Figure 5.3). The first control loop controls the position. The setpoint for the position (Pset) is determined by integrating the requested speed (V_user). The (nominal) error (e_pos) is calculated by subtracting the actual position (Pact). This error is used by position controller CTR_pos to calculate a setpoint for the speed (Vset) in such a way that the stand passes position Pset at the requested speed. Therefore, Vset depends on e_pos, V_user and Vfw. The latter is a feed forward variable that is determined during calibration. The next control loop controls the speed. Figure B.2 shows the dynamics that should be thought of when controlling the speed. It can be divided in a left and right part, both parts representing a lever.
Case Study Details

B.4 Full List of Examined Fault Scenarios

Gravity exerts torques on both the left (torque \( t_1 \)) and right (torque \( t_2 \)) lever of the frontal stand \( \text{(Stand)} \). These two torques should be complemented with a third torque exerted by the motor in order to control the angular acceleration of the stand. This acceleration is used by the speed controller \( \text{CTR}_\text{speed} \) to enable the controlling of the speed. This controller uses a table to determine what current it has to set \( I_{\text{set}} \) in order to achieve a certain acceleration at a particular position \(^1\). So, the difference between the speed setpoint \( V_{\text{set}} \) and the actual speed \( V_{\text{act}} \) results in a certain desired acceleration \( e_{\text{sp}} \). This acceleration is combined with the current position \( P_{\text{act}} \) in order to lookup the associated current setpoint \( I_{\text{set}} \). Here starts the third and final control loop. It is responsible for controlling the current at setpoint \( I_{\text{set}} \). In order to do so it requests a certain current of the \( \text{MVR} \text{(forward)} \)\(^2\) that has a circuit with the \( \text{Motor/Brake} \). The actual current \( I_{\text{act}} \) on this circuit is measured by means of an ampere meter and led back to the current controller \( \text{CTR}_\text{current} \). Because the current is flowing on this circuit the motor exerts a torque \( T_m \). The torque resultant makes that the stand will rotate at a certain speed to a certain position. The speed can be measured by the \( \text{MVR} \text{(feedback)} \). And by leading the signal \( V_{\text{act}} \text{analog} \) back to the controller \( V_{\text{act}} \) the speed control loop has been closed. The closing of the outer position loop is done by measuring the real position. For this purpose a \text{potentiometer} and \text{encoder} \text{(Potmeter/Encoder)} have been assembled upon the rotating axle. When rotated, these components output a voltage. The signals of the potentiometer and encoder are combined in order to achieve a certain accuracy (and to do error detection). The \( \text{LUC Extension(feedback)} \) contains a AD-converter to digitize the analog signal and lead it back to the controller \( \text{LUC} \text{Extension(controller)} \). The position controller will use this value \( P_{\text{act}} \) once again to calculate the (nominal) position error.

B.4 Full List of Examined Fault Scenarios

Table B.2 shows the fault scenarios that are considered in the work for this thesis:

\( \text{S1-S11:} \) presented in the thesis, table 5.4 and table 5.6.

\(^1\)Note that when the Stand rotates the arms of the left and right gravitational torques change. In order to keep the speed constant the torque that the motor has to generate has to change likewise.

\(^2\)Note that both occurrences of \( \text{MVR} \) denote the same FRU. It is only a logical decomposition. The same holds for the LUC Extension FRU.
### Table B.2: All fault scenarios, and their system data.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Imin</th>
<th>Imax</th>
<th>Iact</th>
<th>Iset</th>
<th>Vact</th>
<th>Vset</th>
<th>Pact</th>
<th>Pset</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>−2070</td>
<td>6456</td>
<td>6409</td>
<td>6135</td>
<td>14</td>
<td>29</td>
<td>444</td>
<td>443</td>
<td>current</td>
</tr>
<tr>
<td>S2</td>
<td>−8462</td>
<td>64</td>
<td>−214</td>
<td>−8724</td>
<td>−21</td>
<td>−41</td>
<td>−109</td>
<td>−107</td>
<td>current</td>
</tr>
<tr>
<td>S3</td>
<td>−4879</td>
<td>2001</td>
<td>43</td>
<td>−4390</td>
<td>50</td>
<td>58</td>
<td>137</td>
<td>132</td>
<td>current</td>
</tr>
<tr>
<td>S4</td>
<td>1673</td>
<td>8611</td>
<td>141</td>
<td>8614</td>
<td>−54</td>
<td>−39</td>
<td>289</td>
<td>288</td>
<td>current</td>
</tr>
<tr>
<td>S5</td>
<td>4316</td>
<td>12486</td>
<td>3584</td>
<td>3632</td>
<td>257</td>
<td>−155</td>
<td>324</td>
<td>295</td>
<td>speed</td>
</tr>
<tr>
<td>S6</td>
<td>1548</td>
<td>9778</td>
<td>702</td>
<td>751</td>
<td>252</td>
<td>−53</td>
<td>−279</td>
<td>−300</td>
<td>speed</td>
</tr>
<tr>
<td>S7</td>
<td>−13655</td>
<td>−5485</td>
<td>−4621</td>
<td>−4622</td>
<td>−260</td>
<td>302</td>
<td>784</td>
<td>819</td>
<td>speed</td>
</tr>
<tr>
<td>S8</td>
<td>6089</td>
<td>14259</td>
<td>5183</td>
<td>5171</td>
<td>275</td>
<td>−253</td>
<td>102</td>
<td>106</td>
<td>speed</td>
</tr>
<tr>
<td>S9</td>
<td>−6357</td>
<td>1843</td>
<td>−1899</td>
<td>−1887</td>
<td>−54</td>
<td>−55</td>
<td>−506</td>
<td>−507</td>
<td>position</td>
</tr>
<tr>
<td>S10</td>
<td>−3421</td>
<td>3217</td>
<td>−104</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>−37</td>
<td>−37</td>
<td>position</td>
</tr>
<tr>
<td>S11</td>
<td>4104</td>
<td>10984</td>
<td>7186</td>
<td>6868</td>
<td>−1</td>
<td>−45</td>
<td>17</td>
<td>12</td>
<td>pos. val</td>
</tr>
<tr>
<td>S12</td>
<td>0</td>
<td>0</td>
<td>55</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>77</td>
<td>26</td>
<td>position</td>
</tr>
<tr>
<td>S13</td>
<td>0</td>
<td>0</td>
<td>−43</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>78</td>
<td>78</td>
<td>pos. val</td>
</tr>
<tr>
<td>S14</td>
<td>−2335</td>
<td>4545</td>
<td>1130</td>
<td>1056</td>
<td>135</td>
<td>133</td>
<td>−1100</td>
<td>−1097</td>
<td>pos. val</td>
</tr>
<tr>
<td>S15</td>
<td>−5996</td>
<td>884</td>
<td>1728</td>
<td>1703</td>
<td>−192</td>
<td>−154</td>
<td>291</td>
<td>301</td>
<td>position</td>
</tr>
<tr>
<td>S16</td>
<td>−1140</td>
<td>7386</td>
<td>18</td>
<td>3437</td>
<td>252</td>
<td>217</td>
<td>143</td>
<td>145</td>
<td>current</td>
</tr>
<tr>
<td>S17</td>
<td>−6113</td>
<td>767</td>
<td>−2717</td>
<td>−2644</td>
<td>62</td>
<td>73</td>
<td>432</td>
<td>432</td>
<td>speed</td>
</tr>
<tr>
<td>S18</td>
<td>−6707</td>
<td>1819</td>
<td>2607</td>
<td>2643</td>
<td>−270</td>
<td>−248</td>
<td>160</td>
<td>166</td>
<td>position</td>
</tr>
<tr>
<td>S19</td>
<td>−7836</td>
<td>690</td>
<td>−4170</td>
<td>−4060</td>
<td>−68</td>
<td>−64</td>
<td>131</td>
<td>131</td>
<td>current</td>
</tr>
<tr>
<td>S20</td>
<td>6258</td>
<td>14428</td>
<td>5391</td>
<td>5488</td>
<td>257</td>
<td>−140</td>
<td>869</td>
<td>846</td>
<td>speed</td>
</tr>
<tr>
<td>S21</td>
<td>−3616</td>
<td>2956</td>
<td>214</td>
<td>128</td>
<td>−4</td>
<td>−5</td>
<td>−232</td>
<td>−232</td>
<td>position</td>
</tr>
<tr>
<td>S22</td>
<td>−4428</td>
<td>2452</td>
<td>−177</td>
<td>−4341</td>
<td>27</td>
<td>14</td>
<td>−171</td>
<td>−173</td>
<td>current</td>
</tr>
<tr>
<td>S23</td>
<td>−2514</td>
<td>6012</td>
<td>495</td>
<td>6111</td>
<td>10</td>
<td>27</td>
<td>451</td>
<td>450</td>
<td>current</td>
</tr>
<tr>
<td>S24</td>
<td>−6865</td>
<td>1661</td>
<td>−1044</td>
<td>−6844</td>
<td>−21</td>
<td>−44</td>
<td>22</td>
<td>24</td>
<td>current</td>
</tr>
<tr>
<td>S25</td>
<td>−1164</td>
<td>7362</td>
<td>971</td>
<td>7430</td>
<td>31</td>
<td>58</td>
<td>−453</td>
<td>−456</td>
<td>current</td>
</tr>
<tr>
<td>S26</td>
<td>−6042</td>
<td>2128</td>
<td>−1667</td>
<td>−1655</td>
<td>−30</td>
<td>−29</td>
<td>365</td>
<td>364</td>
<td>position</td>
</tr>
<tr>
<td>S27</td>
<td>−8854</td>
<td>328</td>
<td>92</td>
<td>−9371</td>
<td>13</td>
<td>−38</td>
<td>−311</td>
<td>−307</td>
<td>current</td>
</tr>
<tr>
<td>S28</td>
<td>−7538</td>
<td>988</td>
<td>−7967</td>
<td>−30</td>
<td>−61</td>
<td>−307</td>
<td>−303</td>
<td>−303</td>
<td>current</td>
</tr>
<tr>
<td>S29</td>
<td>−7561</td>
<td>965</td>
<td>−1764</td>
<td>−7576</td>
<td>0</td>
<td>−37</td>
<td>−237</td>
<td>−236</td>
<td>current</td>
</tr>
<tr>
<td>S30</td>
<td>−6976</td>
<td>1550</td>
<td>−67</td>
<td>−7015</td>
<td>−8</td>
<td>−26</td>
<td>−109</td>
<td>−108</td>
<td>current</td>
</tr>
<tr>
<td>S31</td>
<td>−3826</td>
<td>3054</td>
<td>6</td>
<td>−3498</td>
<td>−24</td>
<td>−35</td>
<td>−299</td>
<td>−297</td>
<td>current</td>
</tr>
<tr>
<td>S32</td>
<td>−7189</td>
<td>−309</td>
<td>−128</td>
<td>−7662</td>
<td>6</td>
<td>−27</td>
<td>−246</td>
<td>−246</td>
<td>current</td>
</tr>
<tr>
<td>S33</td>
<td>−2659</td>
<td>5367</td>
<td>1300</td>
<td>1227</td>
<td>37</td>
<td>37</td>
<td>−205</td>
<td>−205</td>
<td>position</td>
</tr>
<tr>
<td>S34</td>
<td>0</td>
<td>0</td>
<td>−43</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>−22</td>
<td>−22</td>
<td>position</td>
</tr>
<tr>
<td>S35</td>
<td>5186</td>
<td>13356</td>
<td>4353</td>
<td>4377</td>
<td>252</td>
<td>−82</td>
<td>339</td>
<td>321</td>
<td>position</td>
</tr>
<tr>
<td>S36</td>
<td>−3974</td>
<td>4226</td>
<td>−43</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>−1</td>
<td>−2</td>
<td>position</td>
</tr>
<tr>
<td>S37</td>
<td>4060</td>
<td>12230</td>
<td>3230</td>
<td>3217</td>
<td>252</td>
<td>−131</td>
<td>329</td>
<td>303</td>
<td>speed</td>
</tr>
<tr>
<td>S38</td>
<td>−3959</td>
<td>4241</td>
<td>−104</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>position</td>
</tr>
<tr>
<td>S39</td>
<td>0</td>
<td>0</td>
<td>−79</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>−35</td>
<td>−35</td>
<td>position</td>
</tr>
<tr>
<td>S40</td>
<td>−1110</td>
<td>3360</td>
<td>857</td>
<td>1004</td>
<td>133</td>
<td>134</td>
<td>1083</td>
<td>1084</td>
<td>status</td>
</tr>
<tr>
<td>S41</td>
<td>−4440</td>
<td>3978</td>
<td>3718</td>
<td>0</td>
<td>−114</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>current/position</td>
</tr>
<tr>
<td>S42</td>
<td>1673</td>
<td>8611</td>
<td>141</td>
<td>8614</td>
<td>−54</td>
<td>−39</td>
<td>289</td>
<td>288</td>
<td>current</td>
</tr>
</tbody>
</table>

S12-S16: injected faults.

S17-S19: injected faults. unrealistic.

S20-S39: fault scenarios, like S1-S19, extracted from FMT tool\(^3\).

\(^3\)FMT is the former version of the ServiceWAX tool.
S40-S42: fault scenarios, extracted from ServiceWAX tool.

### B.5 Discretization of Implementation MBD-2

The implementation MBD-2 of the case study explores the benefits of a discretization. This discretization, the mapping of the integer values on the domain $M$ values, is defined as follows:

$$I_{\text{constant}INT} = \frac{I_{\text{min}} + I_{\text{max}}}{2}$$

$$I_{\text{set}INT} = I_{\text{set}INT} - I_{\text{constant}INT}$$

$$I_{\text{act}INT} = I_{\text{act}INT} - I_{\text{constant}INT}$$

$I_{\text{set}} = \begin{cases} 
TL, & \text{if } e_j \geq TH \_J, I_{\text{set}INT} \in [-\infty, 0], e_{sp} \in [H, TH]; \\
L, & \text{if } I_{\text{set}INT} \in [-\infty, -500], e_j < TH \_J; \\
N, & \text{if } I_{\text{set}INT} \in [-500, 500], e_j < TH \_J; \\
H, & \text{if } I_{\text{set}INT} \in [500, \infty], e_j < TH \_J; \\
TH, & \text{if } e_j \geq TH \_J, I_{\text{set}INT} \in [0, \infty], e_{sp} \in [L, TL]; 
\end{cases}$

$I_{\text{act}} = \begin{cases} 
TL, & \text{if } e_j \geq TH \_J, I_{\text{act}INT} \in [-\infty, 0], I_{\text{set}} \in (\not{\in})[TL, TH]; \\
L, & \text{if } I_{\text{act}INT} \in [-\infty, -500], e_j < TH \_J; \\
N, & \text{if } I_{\text{act}INT} \in [-500, 500], e_j < TH \_J; \\
H, & \text{if } I_{\text{act}INT} \in [500, \infty], e_j < TH \_J; \\
TH, & \text{if } e_j \geq TH \_J, I_{\text{act}INT} \in (0, \infty), I_{\text{set}} \in (\not{\in})[TL, TH]; 
\end{cases}$

$e_{sp} = \begin{cases} 
TL, & \text{if } e_{sp} \geq TH \_V, vset \in [N, H, TH]; \\
L, & \text{if } e_{sp} \in [-\infty, -500], e_j < TH \_V; \\
N, & \text{if } e_{sp} \in [-500, 500], e_j < TH \_V; \\
H, & \text{if } e_{sp} \in [500, \infty], e_j < TH \_V; \\
TH, & \text{if } e_{sp} \geq TH \_V, vset \in [L, TL]; 
\end{cases}$

$V_{act} = \begin{cases} 
TL, & \text{if } V_{\text{act}INT} \in [-\infty, 0], e_{sp} \geq TH \_V; \\
L, & \text{if } V_{\text{act}INT} \in [-\infty, -500], e_j < TH \_V; \\
N, & \text{if } V_{\text{act}INT} \in [-500, 500], e_j < TH \_V; \\
H, & \text{if } V_{\text{act}INT} \in [500, \infty], e_j < TH \_V; \\
TH, & \text{if } V_{\text{act}INT} \in (0, \infty), e_{sp} \geq TH \_V; 
\end{cases}$

$V_{set} = \begin{cases} 
L, & \text{if } V_{\text{set}INT} \in [-\infty, -500], e_j < TH \_V; \\
N, & \text{if } V_{\text{set}INT} \in [-500, 500], e_j < TH \_V; \\
H, & \text{if } V_{\text{set}INT} \in [500, \infty], e_j < TH \_V; 
\end{cases}$

$e_{p} = \begin{cases} 
TL, & \text{if } e_{p} \geq TH \_P, vset \in [N, H, TH]; \\
L, & \text{if } e_{p} \in [-\infty, -500], e_j < TH \_P; \\
N, & \text{if } e_{p} \in [-500, 500], e_j < TH \_P; \\
H, & \text{if } e_{p} \in [500, \infty], e_j < TH \_P; \\
TH, & \text{if } e_{p} \geq TH \_P, vset \in [L, TL]; 
\end{cases}$
B.5 Discretization of Implementation MBD-2

Case Study Details
Appendix C

Lydia Models

This appendix lists the Lydia models that are discussed in this thesis.

C.1 The Synthetic Models that are Used in this Thesis

C.1.1 Structural Model of the 3-inverter Example

```plaintext
system inverter (bool i, h, o)
{
    attribute health(h) = true;
    attribute probability(h) = h ? 0.99 : 0.01;

    // If healthy, a correct input results in a correct output
    h => (i = o);
}

system inverters3
{
    bool correct_w, // input
    bool hA, hB, hC, // healths
    bool correct_y, correct_z // outputs
}
{
    // Declaration internal variables
    bool correct_x;

    // Declaration observables
    attribute observable (correct_y, correct_z) = true;

    // Declaration inverters
    system inverter invA, invB, invC;

    // input is assumed to be correct
```
C.1 The Synthetic Models that are Used in this Thesis

Lydia Models

correct_w = true;

// Connect the 3 inverters
invA (correct_w, hA, correct_x);
invB (correct_x, hB, correct_y);
invC (correct_x, hC, correct_z);

C.1.2 Weak Model of the 3-inverter Example

system inverter (bool i, h, o)
{
    attribute health(h) = true;
    attribute probability(h) = h ? 0.99 : 0.01;

    h => (i = !o); // If healthy, output equals inverse of the input
}

system inverters3
(
    bool w, // input
    bool hA, hB, hC, // healths
    bool y, z // outputs
)
{
    // Declaration internal variables
    bool x;

    // Declaration observables
    attribute observable (w, y, z) = true;

    // Declaration inverters
    system inverter invA, invB, invC;

    // Connect the 3 inverters
    invA (w, hA, x);
    invB (x, hB, y);
    invC (x, hC, z);
}

C.1.3 Strong Model of the 3-inverter Example

system inverter (bool i, h, o)
{
    attribute health(h) = true;
    attribute probability(h) = h ? 0.99 : 0.01;
Lydia Models

C.1 The Synthetic Models that are Used in this Thesis

h => (i = !o); // If healthy, output equals inverse of the input
!h => (o = false); // If unhealthy, the output is stuck-at-zero

system inverters3
{
bool w, // input
bool hA, hB, hC, // healths
bool y, z // outputs
}

// Declaration internal variables
bool x;

// Declaration observables
attribute observable (w, y, z) = true;

// Declaration inverters
system inverter invA, invB, invC;

// Connect the 3 inverters
invA (w, hA, x);
invB (x, hB, y);
invC (x, hC, z);
}

C.1.4 Model of the 4-inverter Example

A strong model of the 4-inverter system.

system inverter (bool h, i, o)
{
    h => (o = !i);
    !h => (o = i);
}

system inverters4 (bool hA, hB, hC, hD, a, b, c, d, e)
{
    system inverter invA, invB, invC, invD;
    invA (hA, a, b);
    invB (hB, b, c);
    invC (hC, c, d);
    invD (hD, d, e);
}
C.2 Model of the Power Supply Example

This subsection lists the MBD model of the power supply example. The power supply example is introduced in Section 2.1.2. Section 4.6 presents the MBD implementation for this example subsystem of the Philips Cardio-Vascular X-Ray System.

C.2.1 Weak Model of the Power Supply Example

```plaintext

system inverts4Main (bool h1, h2, h3, h4, x, a, b, c, y) {
    attribute health(h1) = true;
    attribute probability(h1) = h1 ? 0.99 : 0.01;

    attribute health(h2) = true;
    attribute probability(h2) = h2 ? 0.99 : 0.01;

    attribute health(h3) = true;
    attribute probability(h3) = h3 ? 0.99 : 0.01;

    attribute health(h4) = true;
    attribute probability(h4) = h4 ? 0.99 : 0.01;

    system inverts4 inv4;
    inv4 (h1, h2, h3, h4, x, a, b, c, y);

    attribute observable(a) = true;
    attribute observable(b) = true;
    attribute observable(c) = true;

    x=true;
    y=false;
}
```

C.2 Model of the Power Supply Example

This subsection lists the MBD model of the power supply example. The power supply example is introduced in Section 2.1.2. Section 4.6 presents the MBD implementation for this example subsystem of the Philips Cardio-Vascular X-Ray System.

C.2.1 Weak Model of the Power Supply Example

```plaintext

// // (c) Philips Medical Systems / TU Delft
// // Author : W.M. Lindhoud
// // Filename : power_supply-weak.sys
// // Version : v1.0 (weak model)
// // Date : june, 2006.
// // Description : Weak model of the power supply example,
// as presented in the MSc thesis
```
"Automated Fault Diagnosis at PMS"

// Definition Cable
system Cable
(
    bool p,
    bool q
)
{
    // declaration health variable
    bool h;
    attribute health(h) = true;
    attribute probability(h) = h ? 0.92 : 0.08;

    // definition behavior
    h => ( q = p );
}

// Definition Fuse
system Fuse
(
    bool p,
    bool q
)
{
    // declaration health variable
    bool h;
    attribute health(h) = true;
    attribute probability(h) = h ? 0.90 : 0.10;

    // definition behavior
    h => ( q = p );
}

// Definition Low Voltage Power Supply
system Low_voltage_power_supply
(
    bool input,
    bool out1, out2
)
{
    // declaration health variable
    bool h;
    attribute health(h) = true;
    attribute probability(h) = h ? 0.96 : 0.04;

    // definition behavior
    h => ( ( out1 = input ) and (out2 = input) );
}
// Definition Power Distribution Unit
system PDU
{
    bool startup,
    bool out1, out2, out3
}

// declaration health variable
bool h;
attribute health(h) = true;
attribute probability(h) = h ? 0.95 : 0.05;

h => ( ( out1 = startup ) and ( out2 = startup ) and ( out3 = startup ) );

// Definition of one of a unit that need voltage
// (e.g., collimator, chiller)
system Unit
{
    bool input,
    bool on
}

// declaration health variable
bool h;
attribute health(h) = true;
attribute probability(h) = h ? 0.99 : 0.01;

// definition behavior
h => ( on = input );

// Definition power supply example system
system Power_Supply
{
    bool startup,              // input
    bool status_FD, status_TBCB, status_CRCB, // outputs
    bool status_COL, status_chiller // outputs
}

// declaration intermediate variables
bool cableA_in, cableA_out;
bool cableB_in, cableB_out;
bool cableC_in, cableC_out;
bool fuseA_in, fuseA_out;
bool fuseB_in, fuseB_out;
bool fuseC_in, fuseC_out;
bool fuseD_in, fuseD_out;
bool not_connected1, not_connected2;

// declaration observables
attribute observable (startup, status_FD, status_TBCB) =
    true;
attribute observable (status_CRCB, status_COL, status_chiller) = true;

// declaration components
system Cable cableA, cableB, cableC;
system Fuse fuseA, fuseB, fuseC, fuseD;
system Low_voltage_power_supply LV_PS1, LV_PS2, LV_PS3;
system Unit FD, TBCB, CRCB, collimator, chiller;
system PDU pdu;

// definition structure power supply example
pdu (startup, cableA_in, cableB_in, cableC_in);
cableA (cableA_in, cableA_out);
LV_PS1 (cableA_out, fuseA_in, not_connected1);
fuseA (fuseA_in, fuseA_out);
FD (fuseA_out, status_FD);
cableB (cableB_in, cableB_out);
LV_PS2 (cableB_out, fuseB_in, fuseC_in);
fuseB (fuseB_in, fuseB_out);
TBCB (fuseB_out, status_TBCB);
fuseC (fuseC_in, fuseC_out);
CRCB (fuseC_out, status_CRCB);
LV_PS3 (cableB_out, fuseD_in, not_connected2);
fuseD (fuseD_in, fuseD_out);
collimator (fuseD_out, status_COL);
cableC (cableC_in, cableC_out);
chiller (cableC_out, status_chiller);

C.2.2 Strong Model of the Power Supply Example

/////////////////////////////////////////////////////////////
// (e) Philips Medical Systems / TU Delft
// Author : W.M. Lindhoud
// Filename : power_supply-strong.sys
// Version : v1.0 (strong model)
// Date : june, 2006.
// Description : Strong model of the power supply example,
// as presented in the MSc thesis
// "Automated Fault Diagnosis at PMS"
C.2 Model of the Power Supply Example  

Lydia Models

---

// Definition Cable
system Cable
(
    bool p,
    bool q
)
{
    // declaration health variable
    bool h;
    attribute health(h) = true;
    attribute probability(h) = h ? 0.92 : 0.08;

    // definition behavior
    h => ( q = p );
    !h => ( q = false ); // stuck-at-zero
}

// Definition Fuse
system Fuse
(
    bool p,
    bool q
)
{
    // declaration health variable
    bool h;
    attribute health(h) = true;
    attribute probability(h) = h ? 0.90 : 0.10;

    // definition behavior
    h => ( q = p );
    !h => ( q = false ); // stuck-at-zero
}

// Definition Low Voltage Power Supply
system Low_voltage_power_supply
(
    bool input,
    bool out1, out2
)
{
    // declaration health variable
    bool h;
    attribute health(h) = true;
    attribute probability(h) = h ? 0.96 : 0.04;

    // definition behavior
    h => ( ( out1 = input ) and ( out2 = input ) );
Lydia Models

C.2 Model of the Power Supply Example

!h => ( (out1 = false) and (out2 = false) ); // stuck—at—zero

} // Definition Power Distribution Unit
system PDU
(
    bool startup,
    bool out1 , out2 , out3
) {
    // declaration health variable
    bool h;
    attribute health(h) = true;
    attribute probability(h) = h ? 0.95 : 0.05;
    
    h => ( (out1 = startup) and (out2 = startup) and (out3 = startup) );
    !h => ( (out1 = false) and (out2 = false) and (out3 = false) ); // stuck—at—zero

} // Definition of one of a unit that need voltage
// (e.g., collimator, chiller)
system Unit
(
    bool input, 
    bool on
) {
    // declaration health variable
    bool h;
    attribute health(h) = true;
    attribute probability(h) = h ? 0.99 : 0.01;
    
    // definition behavior
    h => ( on = input );
    !h => ( on = false ); // stuck—at—zero

} // Definition power supply example system
system Power_Supply
(
    bool startup, // input
    bool status_FD, status_TBCB, status_CRCB, // outputs
    bool status_COL, status_chiller // outputs
) {
    // declaration intermediate variables
    bool cableA_in, cableA_out;

    // code

C.3 The Models Constructed for the Case Study

C.3.1 Model of a Control Loop

```c
bool cableB_in, cableB_out;
bool cableC_in, cableC_out;
bool fuseA_in, fuseA_out;
bool fuseB_in, fuseB_out;
bool fuseC_in, fuseC_out;
bool fuseD_in, fuseD_out;
bool not_connected1, not_connected2;

// declaration observables
attribute observable (startup, status_FD, status_TBCB) = true;
attribute observable (status_CRCB, status_COL, status_chiller) = true;

// declaration components
system Cable cableA, cableB, cableC;
system Fuse fuseA, fuseB, fuseC, fuseD;
system Low_voltage_power_supply LV_PS1, LV_PS2, LV_PS3;
system Unit FD, TBCB, CRCB, collimator, chiller;
system PDU pdu;

// definition structure power supply example
pdu (startup, cableA_in, cableB_in, cableC_in);
cableA (cableA_in, cableA_out);
LV_PS1 (cableA_out, fuseA_in, not_connected1);
fuseA (fuseA_in, fuseA_out);
FD (fuseA_out, status_FD);
cableB (cableB_in, cableB_out);
LV_PS2 (cableB_out, fuseB_in, fuseC_in);
fuseB (fuseB_in, fuseB_out);
TBCB (fuseB_out, status_TBCB);
fuseC (fuseC_in, fuseC_out);
CRCB (fuseC_out, status_CRCB);
LV_PS3 (cableB_out, fuseD_in, not_connected2);
fuseD (fuseD_in, fuseD_out);
collimator (fuseD_out, status_COL);
cableC (cableC_in, cableC_out);
chiller (cableC_out, status_chiller);
```

C.3 The Models Constructed for the Case Study

C.3.1 Model of a Control Loop
C.3.2 Model of the MBD-1 Implementation

This subsection lists the compositional model of the target system referred to as the beam propeller movement of the frontal stand, implementation MBD-1, as described in Section 5.4.

// Definition control loop
system Control_loop
{
  bool A,
  bool h_c, h_s,
  bool B
}

// definition behavior 1 control loop
(h_c and h_s) => (B = A);

// Definition control loop
system Control_loop
{
  bool A,
  bool h_c, h_s,
  bool B
}

// definition behavior 1 control loop
(h_c and h_s) => (B = A);
// Definition software error
system Error
(
    bool A, B,
    bool ERROR
) {
    // software is assumed to be correct,
    // so the rule is independent of any health
    ERROR = (A != B);
}

// Definition Potmeter/Encoder Unit
system PEU
(
    bool POSVAL_ERROR,
    bool h
) {
    // definition behavior
    POSVAL_ERROR => !h;
}

// Definition LUC_Extension
system LUC_Extension
(
    bool e_pos,
    bool Pset, Pact,
    bool POSITION_ERROR,
    bool h_PEU,
    bool h
) {
    // definition behavior
    (Pact != Pset) => POSITION_ERROR;
    e_pos = (Pact != Pset);   // by definition
    h => (POSITION_ERROR => (e_pos or !h_PEU));
}

// Definition beam propeller movement of the frontal stand.
system FS_Beam_Propeller_Movement
(
    bool e_pos,
    bool CURRENT_ERROR, SPEED_ERROR, POSITION_ERROR,
    bool POSVAL_ERROR,
    bool h_EXT, h_MVR, h_MBU, h_Stand, h_PEU
) {
    // declaration intermediate variables
    bool Iact, Iset, Vaqt, Vset, Pact, Pset;
}
Lydia Models

C.3 The Models Constructed for the Case Study

// declaration observables
attribute observable (CURRENT_ERROR) = true;
attribute observable (SPEED_ERROR) = true;
attribute observable (e_pos) = true;
attribute observable (POSITION_ERROR) = true;
attribute observable (POSVAL_ERROR) = true;

// declaration health variables
attribute health (h_EXT) = true;
attribute probability (h_EXT) = h_EXT ? 0.97 : 0.03;
attribute health (h_MVR) = true;
attribute probability (h_MVR) = h_MVR ? 0.98 : 0.02;
attribute health (h_MBU) = true;
attribute probability (h_MBU) = h_MBU ? 0.99 : 0.01;
attribute health (h_Stand) = true;
attribute probability (h_Stand) = h_Stand ? 0.95 : 0.05;
attribute health (h_PEU) = true;
attribute probability (h_PEU) = h_PEU ? 0.96 : 0.04;

// declaration components
system Control_loop current_loop, speed_loop, position_loop;
system Error current_error, speed_error;
system PEU peu;
system LUC_Extension luc_extension;

// definition structure control loops
current_loop (Iset, h_EXT, h_MVR and h_MBU, Iact);
speed_loop (Vset, h_EXT and h_MVR and h_MBU, h_Stand, Vact);
position_loop (Pset, h_EXT and h_MVR and h_MBU, h_Stand and h_PEU, Pact);

// definition structure error signals
current_error (Iact, Iset, CURRENT_ERROR);
speed_error (Vact, Vset, SPEED_ERROR);
luc_extension (e_pos, Pset, Pact, POSITION_ERROR, h_PEU, h_EXT);
pou (POSVAL_ERROR, h_PEU);

C.3.3 Model of the MBD-1 Implementation (not compositional)

This subsection lists same model as listed in Section C.3.2, but not specified by using compositional components.
system FS_Beam_Propeller_Movement
{
  bool e_pos,
  bool CURRENT_ERROR, SPEED_ERROR, POSITION_ERROR,
  bool POSVAL_ERROR,
  bool h_EXT, h_MVR, h_MBU, h_Stand, h_PEU
}

// declarations intermediate variables
bool Iact, Iset, Vact, Vset, Pact, Pset;

// declaration observables
attribute observable (CURRENT_ERROR) = true;
attribute observable (SPEED_ERROR) = true;
attribute observable (e_pos) = true;
attribute observable (POSITION_ERROR) = true;
attribute observable (POSVAL_ERROR) = true;

// declaration health variables
attribute health(h_EXT) = true;
attribute probability (h_EXT) = h_EXT ? 0.97 : 0.03;
attribute health(h_MVR) = true;
attribute probability (h_MVR) = h_MVR ? 0.98 : 0.02;
attribute health(h_MBU) = true;
attribute probability (h_MBU) = h_MBU ? 0.99 : 0.01;
attribute health(h_Stand) = true;
attribute probability (h_Stand) = h_Stand ? 0.95 : 0.05;
attribute health(h_PEU) = true;
attribute probability (h_PEU) = h_PEU ? 0.96 : 0.04;

// definition behavior
(h_EXT and h_MVR and h_MBU) => (Iact = Iset);
C.3 The Models Constructed for the Case Study

Lydia Models

C.3 The Models Constructed for the Case Study

(h_EXT and h_MVR and h_MBU and h_Stand) => (Vact = Vset);
(h_EXT and h_MVR and h_MBU and h_Stand and h_PEU) => (Pact = Pset);

// definition error signals
CURRENT_ERROR = (Iact != Iset);
SPEED_ERROR = (Vact != Vset);
(Pact != Pset) => POSITION_ERROR;
e_pos = (Pact != Pset); // by definition
h_EXT => (POSITION_ERROR => (e_pos or !h_PEU));
POSVAL_ERROR => !h_PEU;

C.3.4 Model of the MBD-2 Implementation

This subsection lists the compositional model of the target system referred to as the beam propeller movement of the frontal stand, implementation MBD-2, as described in Section 5.5.

// /**/ Definitions of control loop
system Control_loop
( bool A,
  bool h_c, h_s,
  bool B ) {
  // definition behavior 1 control loop
  (h_c and h_s) => (B = A);
}

// Definitions of software error
system Error
( //

// Definitions of control loop
system Control_loop
( bool A,
  bool h_c, h_s,
  bool B ) {
  // definition behavior 1 control loop
  (h_c and h_s) => (B = A);
}

// Definitions of software error
system Error
( //
C.3 The Models Constructed for the Case Study  

Lydia Models

bool A, B,  
bool ERROR
}) {
    // software is assumed to be correct,  
    // so the rule is independent of any health  
    ERROR = (A != B);  
}

// Definition Potmeter/Encoder Unit  

system PEU  
(  
    bool POSVAL_ERROR,  
    bool h
) {
    // definition behavior  
    POSVAL_ERROR => !h;
}

// Definition LUC_Extension  

system LUC_Extension  
(  
    bool e_pos,  
    bool ctr_speed,  
    bool Iset, Pset, Pact,  
    bool POSITION_ERROR,  
    bool h_PEU,  
    bool h_EXT
) {
    // declaration internals  
    bool e_sp;

    // definition derivative ctr_speed  
    ctr_speed = (Iset != e_sp);

    // definition behavior  
    h_EXT => (Iset = e_sp);
    (Pact != Pset) => POSITION_ERROR;
    e_pos = (Pact != Pset);  // by definition  
    h_EXT => (POSITION_ERROR => (e_pos or !h_PEU));
}

// Definition beam propeller movement of the frontal stand.  

system FS_Beam_Propeller_Movement  
(  
    bool e_pos,  
    bool CURRENT_ERROR, SPEED_ERROR, POSITION_ERROR,  
    bool POSVAL_ERROR, ctr_speed,
bool h_EXT, h_MVR, h_MBU, h_Stand, h_PEU

// declaration intermediate variables
bool Iact, Iset, Vact, Vset, Pact, Pset;

// declaration observables
attribute observable (CURRENT_ERROR) = true;
attribute observable (SPEED_ERROR) = true;
attribute observable (e_pos, ctr_speed) = true;
attribute observable (POSITION_ERROR) = true;
attribute observable (POSVAL_ERROR) = true;

// declaration health variables
attribute health(h_EXT) = true;
attribute probability(h_EXT) = h_EXT ? 0.97 : 0.03;
attribute health(h_MVR) = true;
attribute probability(h_MVR) = h_MVR ? 0.98 : 0.02;
attribute health(h_MBU) = true;
attribute probability(h_MBU) = h_MBU ? 0.99 : 0.01;
attribute health(h_Stand) = true;
attribute probability(h_Stand) = h_Stand ? 0.95 : 0.05;
attribute health(h_PEU) = true;
attribute probability(h_PEU) = h_PEU ? 0.96 : 0.04;

// declaration components
system Control_loop current_loop, speed_loop, position_loop;
system Error current_error, speed_error;
system PEU peu;
system LUC_Extension luc_extension;

// definition structure control loops
current_loop(Iset, h_EXT, h_MVR and h_MBU, Iact);
speed_loop(Vset, h_EXT and h_MVR and h_MBU, h_Stand, Vact);
position_loop(Pset, h_EXT and h_MVR and h_MBU, h_Stand and h_PEU, Pact);

// definition structure error signals
current_error(Iact, Iset, CURRENT_ERROR);
speed_error(Vact, Vset, SPEED_ERROR);
luc_extension(e_pos, ctr_speed, Iset, Pset, Pact,
              POSITION_ERROR, h_PEU, h_EXT);
peu(POSVAL_ERROR, h_PEU);
}
C.3.5 Model of the MBD-2 Implementation (not compositional)

This subsection lists same model as listed in Section C.3.4, but not specified by using compositional components.

```plaintext
// // (c) Philips Medical Systems / TU Delft
//
// Author : W.M. Lindhoud
//
// Filename : bpm-mbd2-uc.sys
//
// Version : MBD-2, not compositional
//
// Date : june , 2006.
//
// Description : Model of the beam propeller movement
//
// of the frontal stand, implementation MBD-2,
//
// as presented in the MSc thesis
//
// "Automated Fault Diagnosis at PMS"
//
// Note : This uncompositional version
//
// is used during development
//

system FS_Beam_Propeller_Movement
{
    bool e_pos,
    bool CURRENT_ERROR, SPEED_ERROR, POSITION_ERROR,
    bool POSVAL_ERROR, ctr_speed,
    bool h_EXT, h_MVR, h_MBU, h_Stand, h_PEU
}
```

```
// declaration intermediate variables
bool Iact, Iset, Vact, Vset, Pact, Pset;
bool e_sp;

// declaration observables
attribute observable (CURRENT_ERROR) = true;
attribute observable (SPEED_ERROR) = true;
attribute observable (e_pos, ctr_speed) = true;
attribute observable (POSITION_ERROR) = true;
attribute observable (POSVAL_ERROR) = true;

// declaration health variables
attribute health(h_EXT) = true;
attribute probability(h_EXT) = h_EXT ? 0.97 : 0.03;
attribute health(h_MVR) = true;
attribute probability(h_MVR) = h_MVR ? 0.98 : 0.02;
attribute health(h_MBU) = true;
attribute probability(h_MBU) = h_MBU ? 0.99 : 0.01;
attribute health(h_Stand) = true;
attribute probability(h_Stand) = h_Stand ? 0.95 : 0.05;
```
Lydia Models

C.3 The Models Constructed for the Case Study

attribute health(h_PEU) = true;
attribute probability(h_PEU) = h_PEU ? 0.96 : 0.04;

// definition behavior
(h_EXT and h_MVR and h_MBU) => (Iact = Iset);
(h_EXT and h_MVR and h_MBU and h_Stand) => (Vact = Vset);
(h_EXT and h_MVR and h_MBU and h_Stand and h_PEU) => (Pact = Pset);
h_EXT => (Iset = e_sp);

// definition error signals
CURRENT_ERROR = (Iact != Iset);
SPEED_ERROR = (Vact != Vset);
(Pact != Pset) => POSITION_ERROR;
e_pos = (Pact != Pset); // by definition
ctr_speed = (Iset != e_sp); // definition derivative
ctr_speed
h_EXT => (POSITION_ERROR => (e_pos or !h_PEU));
POSVAL_ERROR => !h_PEU;

C.3.6 Model of the MBD-3 Implementation

This subsection lists the compositional model of the target system referred to as the beam propeller movement of the frontal stand, implementation MBD-3, as described in Section 5.6.

// (c) Philips Medical Systems / TU Delft
// Author : W.M. Lindhoud
// Filename : bpm-mbd3-c.sys
// Version : MBD-3, not compositional
// Date : june, 2006.
// Description : Model of the beam propeller movement of the frontal stand, implementation MBD-3, as presented in the MSc thesis "Automated Fault Diagnosis at PMS"

// Definition software error
system Error
(
    bool A, B,
    bool ERROR
) {
    // software is assumed to be correct,
}
C.3 The Models Constructed for the Case Study

Lydia Models

    // so the rule is independent of any health
    ERROR = (A != B);
}

system PEU
(
    bool POSVAL_ERROR,
    bool h
) {
    // definition behavior
    POSVAL_ERROR => !h;
}

// Definition LUC_Extension

system LUC_Extension
(
    bool e_pos,
    bool ctr_speed,
    bool Iset, Pset, Pact,
    bool POSITION_ERROR,
    bool h_PEU,
    bool h_EXT
) {
    // declaration internals
    bool e_sp;

    // definition derivative ctr_speed
    ctr_speed = (Iset != e_sp);

    // definition behavior
    h_EXT => (Iset = e_sp);
    (Pact != Pset) => POSITION_ERROR;
    e_pos = (Pact != Pset);  // by definition
    h_EXT => (POSITION_ERROR => (e_pos or !h_PEU));
}

// Definition of a component

system Component
(
    bool in_status,
    bool out_status,
    bool h
) {
    // if the component is healthy,
    // a correct in_status results in a correct out_status.
    h => (out_status = in_status);
}
// definition 'cut-open' current loop
system Current_loop
{
    bool Iset, Imvr, I_to_motor, I_from_motor, Iact_analog, 
    Iact, 
    bool h_EXT, h_MVR, h_MotorBrake 
}

// declaration components
system Component EXT_out, MVR_forward, MBU, MVR_backward, 
    EXT_in;

// definition behavior
EXT_out (Iset, Imvr, h_EXT);
MVR_forward (Imvr, I_to_motor, h_MVR);
MBU (I_to_motor, I_from_motor, h_MotorBrake);
MVR_backward (I_from_motor, Iact_analog, h_MVR);
EXT_in(Iact_analog, Iact, h_EXT);

// definition 'cut-open' speed loop
system Speed_loop
{
    bool Vset, Vmvr, V_to_motor, torque, real_speed, 
    Vact_analog, Vact, 
    bool h_EXT, h_MVR, h_MotorBrake, h_Stand 
}

// declaration components
system Component EXT_out, MVR_forward, MBU, stand, 
    MVR_backward, EXT_in;

// definition behavior
EXT_out (Vset, Vmvr, h_EXT);
MVR_forward (Vmvr, V_to_motor, h_MVR);
MBU (V_to_motor, torque, h_MotorBrake);
stand (torque, real_speed, h_Stand);
MVR_backward (real_speed, Vact_analog, h_MVR);
EXT_in(Vact_analog, Vact, h_EXT);

// definition 'cut-open' position loop
system Position_loop
{
    bool Pset, CURRENT_ERROR, SPEED_ERROR, real_position, 
    Pac_analog, e_pos, 
    bool h_Stand, h_PEU, h_EXT 
}

}}
C.3 The Models Constructed for the Case Study

Lydia Models

```plaintext
// declaration components
system Component peu;

// definition behavior
(!CURRENT_ERROR and !SPEED_ERROR) = Pset;
hs Stand => (Pset => real_position);
peu (real_position, Pact_analog, h_PEU);
h_EXT => (e_pos = !Pact_analog);

// Definition beam propeller movement of the frontal stand.
system FS_Beam_Propeller_Movement
{
  bool Iact, Iset,
  bool Vact, Vset,
  bool e_pos,
  bool Imvr, I_to_motor, I_from_motor, Iact_analog,
  bool Vmvr, V_to_motor, torque, real_speed, Vact_analog,
  bool real_position, Pact_analog,
  bool CURRENT_ERROR, SPEED_ERROR, POSITION_ERROR,
  bool POSVAL_ERROR, ctr_speed,
  bool h_EXT, h_MVR, h_MotorBrake, h_Stand, h_PEU
}

// declaration intermediate variables
bool Pact, Pset;

// define variables that are already observed
// currently defined: Fault scenario C1 (current error)
CURRENT_ERROR = true;
SPEED_ERROR = false;
e_pos = false;
ctr_speed = false;
POSITION_ERROR = false;
POSVAL_ERROR = false;

// declaration additional observables
attribute observable (I_mvr, I_to_motor, I_from_motor) = true;
attribute observable (real_speed) = true;
attribute observable (Vact, Vset) = true;
attribute observable (Iact, Iset) = true;

// declaration health variables
attribute health (h_EXT) = true;
attribute probability (h_EXT) = h_EXT ? 0.97 : 0.03;
attribute health (h_MVR) = true;
attribute probability (h_MVR) = h_MVR ? 0.98 : 0.02;
```
attribute health(h_MotorBrake) = true;
attribute probability(h_MotorBrake) = h_MotorBrake ? 0.99 : 0.01;
attribute health(h_Stand) = true;
attribute probability(h_Stand) = h_Stand ? 0.95 : 0.05;
attribute health(h_PEU) = true;
attribute probability(h_PEU) = h_PEU ? 0.96 : 0.04;

// declaration components
system Current_loop current_loop;
system Speed_loop speed_loop;
system Position_loop position_loop;
system Error current_error, speed_error;
system PEU peu;
system LUC_Extension luc_extension;

// definition structure control loops
current_loop (Iset, Imvr, I_to_motor, I_from_motor,
  Iact_analog, Iact, h_EXT, h_MVR, h_MotorBrake);
speed_loop (Vset, Vmvr, V_to_motor, torque, real_speed,
  Vact_analog, Vact, h_EXT, h_MVR, h_MotorBrake, h_Stand);
position_loop (Pset, CURRENT_ERROR, SPEED_ERROR,
  real_position, Pact_analog, e_pos, h_Stand, h_PEU,
  h_EXT);

// definition structure error signals
current_error (Iact, Iset, CURRENT_ERROR);
speed_error (Vact, Vset, SPEED_ERROR);
luc_extension (e_pos, ctr_speed, Iset, Pset, Pact,
  POSITION_ERROR, h_PEU, h_EXT);
pceu (POSVAL_ERROR, h_PEU);

C.3.7 Model of the MBD-3 Implementation (not compositional)

This subsection lists same model as listed in Section C.3.6, but not specified by using compositional components.

// (c) Philips Medical Systems / TU Delft

// Author : W.M. Lindhoud
// Filename : bpm-mbd3-uc.sys
// Version : MBD-3, not compositional
// Date : june, 2006.
C.3 The Models Constructed for the Case Study

Lydia Models

// Description: Model of the beam propeller movement
// of the frontal stand, implementation MBD−3,
// as presented in the MSc thesis
// "Automated Fault Diagnosis at PMS"
// Note: This uncompositional version
// is used during development

system FS_Beam_Propeller_Movement
(
  bool Iact, Iset,
  bool Vact, Vset,
  bool e_pos,
  bool I_mvr, I_to_motor, I_from_motor, Iact_analog,
  bool V_mvr, V_to_motor, torque, real_speed, Vact_analog,
  bool real_position, Pact_analog,
  bool CURRENT_ERROR, SPEED_ERROR, POSITION_ERROR,
  bool POSVAL_ERROR, ctr_speed,
  bool h_EXT, h_MVR, h_MotorBrake, h_Stand, h_PEU
)
{
  // declarations intermediate variables
  bool Pact, Pset;
  bool e_sp;

  // define variables that are already observed
  // currently defined: Fault scenario C1 (current error)
  CURRENT_ERROR = true;
  SPEED_ERROR = false;
  e_pos = false;
  ctr_speed = false;
  POSITION_ERROR = false;
  POSVAL_ERROR = false;

  // declaration additional observables
  attribute observable (I_mvr, I_to_motor, I_from_motor) =
    true;
  attribute observable (real_speed) = true;
  attribute observable (Vact, Vset) = true;
  attribute observable (Iact, Iset) = true;

  // declaration health variables
  attribute health(h_EXT) = true;
  attribute probability(h_EXT) = h_EXT ? 0.97 : 0.03;
  attribute health(h_MVR) = true;
  attribute probability(h_MVR) = h_MVR ? 0.98 : 0.02;
  attribute health(h_MotorBrake) = true;

Lydia Models

C.3 The Models Constructed for the Case Study

\begin{verbatim}
attribute probability(h_MotorBrake) = h_MotorBrake ? 0.99 : 0.01;
attribute health(h_Stand) = true;
attribute probability(h_Stand) = h_Stand ? 0.95 : 0.05;
attribute health(h_PEU) = true;
attribute probability(h_PEU) = h_PEU ? 0.96 : 0.04;

// definition behavior current loop
h_EXT => (I_mvr = I_set);

h_MVR => (I_to_motor = I_mvr);

h_MotorBrake => (I_from_motor = I_to_motor);

h_MVR => (Iact_analog = I_from_motor);

h_EXT => (Iact = Iact_analog);

// definition behavior speed loop
h_EXT => (V_mvr = Vset);

h_MVR => (V_to_motor = V_mvr);

h_MotorBrake => (torque = V_to_motor);

h_Stand => (real_speed = torque);

h_MVR => (Vact_analog = real_speed);

h_EXT => (Vact = Vact_analog);

// definition behavior position loop
(!CURRENT_ERROR and !SPEED_ERROR) = Pset;

h_Stand => (Pset => real_position);

h_PEU => (Pact_analog = real_position);

h_EXT => (epos = !Pact_analog);

// specified behavior internal of the LUC_Extension
h_EXT => (Iset = e_sp);

// definition error signals
CURRENT_ERROR = (Iact != Iset);
SPEED_ERROR = (Vact != Vset);
(Pact != Pset) => POSITION_ERROR;
epos = (Pact != Pset);    // by definition
ctr_speed = (Iset != e_sp);    // definition derivative
ctr_speed
h_EXT => (POSITION_ERROR => (epos or !h_PEU));
POSVAL_ERROR => !h_PEU;
\end{verbatim}
Appendix D

State-of-the-Future
Fault Diagnosis at PMS

In this appendix, the proposed procedure for applying the model-based approach to fault diagnosis at PMS is presented. This appendix is best understood when Chapter 1 till Chapter 5 are read. Figure 2.3 shows the current procedure to fault diagnosis. The proposed procedure is shown in Figure D.1. The help desks and Service Innovation departments shown in the figure that describes the current approach are not shown in Figure D.1. This is because the new figure focusses on an ideal situation, in which the fault diagnosis process is not dependent on the activities of these departments. Of course, this ideal situation cannot be reached, because there will always be unexpected faults and failures.

The lines and objects of Figure D.1 show the following. The system remotely monitors the log data on a system. The data is stored in a database, and a web-based application shows the data to supervisory users. These are users that for some reason need to interpret the data (e.g., fault diagnosis, marketing, user profiling, developing a MBD implementation, etc.). Another application extracts data from the database, discretizes it to a domain with few members, and inserts it into the diagnostic engine LYDIA. The other input of this diagnostic engine are LYDIA models of the Philips Cardio-Vascular X-Ray System. These models are specified by experts (e.g., developers, designers, service specialists, etc). At regular time intervals, the inserted observations and models are used to generate diagnoses. If the diagnostic engine identifies unhealthy components a service engineer is notified to repair the system. The service engineer can use the diagnostic system for examining details about the system state. If necessary, the service engineer refines the diagnosis produced by the diagnostic engine. At the moment that the diagnostic engine identified the unhealthy components the logistic department is notified to supply the appropriate FRUs to the right hospital.

There are some alternative decisions to the situation described above. The monitor and diagnostic engine could be implemented on each Philips Cardio-Vascular X-Ray System. An advantage is that service engineers are used to use special tools on the system. More importantly, this enables the implementation of automated system recovery, because in this set-up, a failure can be diagnosed milliseconds after it occurred. The reason for choosing the set-up that includes remote monitoring is that it provides a convenient environment to develop and research the applicability of the model-based approach at Philips Medical Systems. For example, the database of the remote monitoring tool set provides a huge amount of fault scenarios that can be used to for constructing the model. Moreover, the fault scenarios indicate which faults occur very often and might need a MBD implementation for diagnosing them.
Figure D.1: Proposed procedure for doing fault diagnosis (and repair) at PMS.