Automated Fault Diagnosis at Philips Medical Systems

A Model-Based Approach at Philips Medical Systems

Master’s Thesis

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

A Model-Based Approach at Philips Medical Systems

THESIS

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

SOFTWARE ENGINEERING

by

W.M. Lindhoud

born in Gouda, the Netherlands
Automated Fault Diagnosis at Philips Medical Systems

A Model-Based Approach at Philips Medical Systems

Author: W.M. Lindhoud
Student id: 1053884
Email: wouter.lindhoud@philips.com

Abstract

As our machines get faster, better and cheaper every day the increase in complexity of these systems is huge. This is no 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 exists, so far, there has not been any research about how to set up a diagnostic system. This work is a first exploration of the benefits that such a technique could have for the diagnosis of the Philips Cardio-Vascular X-Ray System. This work 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 seems to suit the goals and qualities the best, because it is able to utilize all relevant information. Shannon's entropy is used as a heuristic to quantify the uncertainty of the diagnoses. By means of a case study of a subsystem, it is shown that a Model-Based approach is able to achieve lower uncertainty in its diagnoses than other automated approaches.

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
Preface

Funny start and acknowledgements.

W.M. Lindhoud
Delft, the Netherlands
May 15, 2006
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>iii</td>
</tr>
<tr>
<td>Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
</tbody>
</table>

### 1 Introduction
1.1 Fault Diagnosis ................................................. 2
1.2 Automated Fault Diagnosis ..................................... 4
1.3 Model-Based Fault Diagnosis ................................. 5
1.4 Problem Statement ............................................... 6
1.5 Outline of the Thesis .......................................... 7

### 2 State-of-the-practice
Fault diagnosis at PMS .............................................. 9
2.1 Preliminaries .................................................... 9
2.2 Overview Today’s Practice ...................................... 12
2.3 Manual Fault Diagnosis ........................................ 15
2.4 Optimal Fault Diagnosis ........................................ 15

### 3 Automated Fault Diagnosis
3.1 Overview Techniques ............................................. 21
3.2 Data Mining Approach Using Historical Data (BB-1) .......... 23
3.3 Classical Approach (CA) ........................................ 24
3.4 Data Mining Approach Using Structure (BB-2) ............... 25
3.5 Model-Based Approach (MBD) ................................... 26
3.6 Evaluation of the AFD-approaches ............................... 27

### 4 Model-Based Fault Diagnosis
4.1 Fundamentals ..................................................... 30
4.2 Model-Based Diagnosis with LYDIA ............................ 32
4.3 Modeling .......................................................... 35
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 Automated Fault Diagnosis approaches . . . . . . . . . . 22
3.2 Power Supply - Black Box view. . . . . . . . . . . . . . . . . . . . . . 23
3.3 Partial architecture of the power supply (repetition of figure 2.2). . . . . 24

4.1 3-inverters example . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
4.2 4-inverter example . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

5.1 Jobsheet referring to problems with the beam propeller movement. . . 43
5.2 Beam Propeller Movement of the frontal stand . . . . . . . . . . . . . 46
5.3 Partial architecture of the FRUs that implement the beam propeller move- ment. Signals enclosed by brackets are not observable, the others are. . . . . 52
5.4 Modeling the control loops . . . . . . . . . . . . . . . . . . . . . . . . 53
5.5 Modeling the error signal . . . . . . . . . . . . . . . . . . . . . . . . . 54

B.1 The parts of the frontal stand . . . . . . . . . . . . . . . . . . . . . . . 63
B.2 Forces and torques on the frontal stand. . . . . . . . . . . . . . . . . . . 65
## List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Partial diagnosis look-up table of the Power Supply system for single faults (startup = true)</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>input matrix for data analysis of the example fault scenario</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>Evaluation approaches.</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Mode catalog of the 4-inverter model, when the observations, x=1 and y=0, have been made.</td>
<td>40</td>
</tr>
<tr>
<td>5.1</td>
<td>Example of a log entry (when error 11 occurs).</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>The components that are excluded from the diagnostic process, described in this chapter</td>
<td>47</td>
</tr>
<tr>
<td>5.3</td>
<td>The FRUs and their associated health variables with a priori component failure probabilities</td>
<td>48</td>
</tr>
<tr>
<td>5.4</td>
<td>Categorizing fault scenarios based on observations. This data has been taken from fault situations as logged in the error 11 messages.</td>
<td>49</td>
</tr>
<tr>
<td>5.5</td>
<td>Partial diagnosis look-up table of the Beam Propeller Movement for single faults</td>
<td>50</td>
</tr>
<tr>
<td>5.6</td>
<td>One way to discretize the observables.</td>
<td>55</td>
</tr>
<tr>
<td>B.1</td>
<td>The meaning of the system data that can be found in an error 11 entry.</td>
<td>64</td>
</tr>
</tbody>
</table>
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 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. Otherwise, the costs and effort that we devote to manually diagnosing the systems, and keep them up and running in a safe way, are just not acceptable anymore.

The next section points out the role of a fault diagnostic 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

\(^{1}\)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 is one step later; it 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 points 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.
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 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. 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.

Remote Monitoring

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\(^2\) has been started within Philips Cardio-Vascular Development \([18][17][6]\). 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, had been shortened. This is because more information could be combined, and experts were able to interpret it all remotely, without traveling to the hospital where the system is located. However, the application of this technique is very new, and the fault diagnosis practice of today still has its drawbacks.

Drawbacks of Today’s Practice

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

\(^2\)the project is currently known as ServiceWax
1.2 Automated Fault Diagnosis

Introduction

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 customer downtime 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, lots of tools, code and mechanism that support fault diagnosis require re-implementation.

Remote monitoring improves the diagnostic performance by providing lots of useful data. However, experts still have to interpret this data manually. Also knowledgeable experts need to devote lots of 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 lots of potential benefits compared to manual approaches. Automated fault diagnosis approaches automate the interpretation of raw data, in order to produce diagnoses. It is 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 it 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 a lot of data that possesses information about faulty components. This is a practical environment to examine various automated approaches to fault diagnosis. The large amount of data that remote monitoring produces suggests:

• Data mining techniques. Such techniques search for correlations between log data and faulty components.

Data mining falls in category of black-box (no decomposed behavioral description available) approaches. Alternatively, it is possible to apply an approach in which experts use their white-box information (a decomposed behavioral description) about the system, such as:

• The classical approach to automated fault diagnosis. Experts manually define a mapping between symptoms and diagnoses. The mapping is implemented by application-specific code or, more generally, by using expert systems. This thesis refers to this approach as the classical approach, because most literature
considers this approach as the more traditional way of doing automated fault diagnosis (because it is the most common approach to automated fault diagnosis in industry).

Another approach that uses white-box information is described in the next section. Instead of restricting itself to effect-to-causes reasonings, like in the classical approach, this technique uses a model that could also define cause-to-effect relationships. The next section introduces this technique.

### 1.3 Model-Based Fault Diagnosis

Model-Based fault Diagnosis (MBD) is a technique for doing fault diagnosis based on the 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 [13] and continued by de Kleer, Mackworth and Reiter[10].

The idea behind the technique is shown in 1.2. The cloud represents the real system, and its runtime operation. The model formally defines structure, as well as nom-
inal and 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][15]. 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 semiconductor industry [5][2][3].

The diagnostic engine of LYDIA has already been constructed, and is only 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 Problem Statement

The previous sections lead to the hypothesis that automated fault diagnosis, and in particular the model-based approach, is able to improve fault diagnosis. Improvement means that the alternative fault diagnosis technique should achieve higher dependability than today’s practice does. In other words, the alternative approach should have a higher diagnostic performance.

This work explores the possible benefits of automated fault diagnosis of the most recent Cardio-Vascular X-Ray Systems at Philips Medical Systems. The problem encountered in this thesis is to present a proof-of-concept of the model-based approach to fault diagnosis, aimed at the Philips Cardio-Vascular X-Ray system. In order to achieve this, the aims of the project is to

1. uncover the drawbacks of today’s practice to fault diagnosis (chapter 2).
2. show which automated techniques to fault diagnosis could possibly increase diagnostic performance (chapter 3).
3. find criteria for developing a model-based approach (section 2.4).
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 learn typically occur when MBD is applied in the industrial domain (chapter 5).
5. try to figure out how LYDIA is able to model specific dynamics, as well as the use of the accompanying tool set in a real-life scenario (chapter 5).
The result of the project should be a comparison of the diagnostic performance of today’s practice to fault diagnosis, and the model-based approach. This can best be done by using a metric for diagnostic accuracy. Accuracy is the extent to which diagnoses agree with reality, and is considered to be the most important criterion for diagnostic performance. For this reason, the experimental methodology requires that diagnoses made by the diagnostic engine could be compared with what was actually broken. However, due to a missing link between observations and real diagnoses, this is not possible. Consequently, it is only possible to draw statistical conclusions (chapter 6).

1.5 Outline of the Thesis

The outline of the thesis is as follows. In the next chapter, the current approach to fault diagnosis is reviewed and analyzed by means of an example. The final section of chapter 2 presents criteria to estimate diagnostic performance. This enables an evaluation 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 the problem, and uses the criteria of chapter 2, for the 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 that MBD is a solution to the problem. Chapter 6 concludes this thesis with implications of the solution, conclusions and recommendations.
Chapter 2

State-of-the-practice Fault diagnosis at PMS

This chapter describes the manual practice to fault diagnosis, that industry nowadays applies. The fault diagnosis process, aimed at the Philips Cardio-Vascular X-Ray system, is used to clarify this approach. The outline of this chapter is as follows. The first section introduces the preliminaries that are needed to understand this and subsequent chapters. The second section gives an overview of today’s means and procedures within PMS for doing fault diagnosis. The third section concretizes the current practice by applying it on a real example. 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 its diagnostic performance.

2.1 Preliminaries

This section introduces terminology and concepts that play an important role throughout the thesis. The foreknowledge improves the reader’s understanding of the line of thought. 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.

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.
2.1 Preliminaries

State-of-the-practice

Fault diagnosis at PMS

Introduction Running Example: Power Supply

This section introduces a running 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).

This 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 within Philips Cardio-Vascular X-Ray system. The following preliminary section also uses the example, that is introduced in this section, in order to introduce some fault diagnosis related concepts.
Faults, Errors, Failures

The previous section introduced a subsystem of Philips Cardio-Vascular X-Ray system. This section uses this example to introduce notions related to fault diagnosis. 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 [12]. The function of a system is defined as what the system is intended to do. 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. In an "unhealthy" case, the system shows other behavior than the nominal behavior.

This unexpected behavior is preceded by a transition of correct functioning to malfunctioning. This is called a failure. 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. This means that the external system state deviates from the correct system state. Suppose the deviation is that TBCB, CRCB and Collimator are off, while they should be on. This is called the error. The cause of this error can be that (part of) the system is used outside its specification. This is likely in a company that uses many third party components in its systems. However, this writing focusses on the other possibility, namely that one or more components have been broken. If one component is adjudged or hypothesized to be broken, this 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. In general, a fault is the cause of error.

A fault that could cause the above mentioned error (TBCB, CRCB and Collimator are off, while they should be on), is that CableB has been broken. 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” has been broken” suffices. Usually, there is not one possibility for a certain error. A multiple fault explanation of the error is that the FRUs LV_PS2 and FuseD have been broken. TODO: fault scenario. Another important thing 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). Internal errors could be dealt with by other subsystems. Fault diagnosis could aim at the identification of all faults. However, the work in this thesis aims at the identification of active faults that result in a failure. As long as a fault has not been resolved, the user(s) cannot depend on the system. TODO: Faults, errors, and failures could occur on various subsystem levels. TODO?: figure shows the error propagation, pathology of faults, error and failures. TODO?: fault modes, symptom, observations, etc?

TODO: rewrite the following paragraph. It is assumed that some diagnostic process out of the scope of this example (either a manual search or some automated fault diagnosis approach) already pinpoints\(^1\) the search to the components shown in figure 2.2. A diagnostic process should identify CableB as a suspicious component. In this chapter this example subsystem is used to discuss the way the current practise addresses a concrete fault scenario. The chapter (Chap. 4), that introduces the new technique, uses the same example to enable a direct comparison.

### 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.

**procedure**

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 send 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

\(^1\)Various cables, fuses, other powered components and the part between the powered components and the outputted signals (among them: CAN buses, SIB and Host) are left out for sake of simplicity.
State-of-the-practice
Fault diagnosis at PMS

2.2 Overview Today’s Practice

Figure 2.3: The current procedure for doing fault diagnosis (and repair).

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. Finally, people that developed the system are called to help the service engineer or solve the problem themselves. The red lines, all, indicate actions that have to be taken for doing the fault diagnosis. The blue lines represent the efforts of Service Innovation to get the knowledge, prior to this entire process, from the developers to the service engineer (and to the help desks). They try to prevent as many problems as possible from propagating to the C/V development department. The green lines represent the supply of substitutable FRUs.

means

The list below is an overview of all means that a troubleshooter\(^2\) 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 pin-points the search for the malfunctioning FRU to a certain subsystem or collection of FRUs. It is also possible that he finds out that the failing 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.

---

\(^2\)In this section we use the term **troubleshooter**, instead of **service engineer**, to emphasize the fact that these means are meant for all troubleshooters.
2.2 Overview Today’s Practice

State-of-the-practice

Fault diagnosis at PMS

Checking the log The log contains information about past system behavior in the form of error messages, warnings and other informational parameters. They all could pinpoint to a particular subsystem. However, the service engineer has to understand this information.

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 artefact 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 (although they tend to be quite ambiguous).

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, FIPs are all artifacts made by Service Innovation, and illustrated in figure 2.3 by the books in the actor’s hands.

---

3\textit{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.
Fault diagnosis at PMS

2.3 Manual Fault Diagnosis

This section describes the current approach by using the fault scenario in the power supply that states that CableB has been broken. As section 2.2 described, a service engineer has to visit the system 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.

Drawbacks

The described process above, shows that finding a simple broken cable takes lots of time. In the real case, there are even more cables, fuses and components. All of them must be checked, and this is very time consuming: more or less an hour. If one of the more complex components (the components depicted on the right of 2.2) are broken, and LEDs do not indicate malfunctioning, the identification of these as the wrongdoers takes lots more time. In these situations, employees from development need to help and add their knowledge.

2.4 Optimal Fault Diagnosis

This section defines and clarifies the problem that this thesis encounters precisely. The former described the current approach to fault diagnosis, as you see most in industry. In order to evaluate the drawbacks, a definition of the "perfect fault diagnosis process" is needed. This definition determines the goals of a fault diagnosis technique. It counts for PMS as well as for all companies constructing embedded systems. From the definition, items are derived that make a fault diagnosis approach perfect. The final subsection shows what items could possibly be introduced, or improved, in order to solve the problem.

Perfect Fault Diagnosis

Here, an answer is given to the question what we consider as a "perfect" fault diagnosis process. A definition is given below.

**Definition.** A perfect fault diagnosis process is a practice of doing fault diagnosis that 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.

For achieving this situation items that make it fit this definition are derived. These can be used to evaluate diagnostic approaches. The following items would make a fault diagnosis process "perfect":

15
2.4 Optimal Fault Diagnosis

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:

   - *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 4.
   - *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 *Shannon's 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. **Independent of its environment.** If the successful working of the diagnostic process highly depends on forces outside the sphere of influence of the company, obviously risks increase.

5. **Development costs.** Development costs are all the costs that have to be made prior to the start of the diagnostic process. This includes the development of all supporting artifacts, training sessions for troubleshooters and all 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.

The following items would add to perfectness but are out of scope of this thesis:

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

   - *Novelty Identifiability.* The ability to detect and diagnose faults that have not occurred before.

   - *Ability to deal with multiple faults.*

   - *Reasonable storage and computational requirement.*

---

4At PMS this means that a component that is diagnosed as unhealthy will be replaced.
State-of-the-practice
Fault diagnosis at PMS

2.4 Optimal Fault Diagnosis

Evaluation Current Approach

The suboptimality of the diagnostic process has developed gradually through the years. When the first Cardio-Vascular systems were placed at 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. The skills and knowledge of service engineers perfectly fitted the diagnostic tasks. Therefore, analog measuring was the most efficient way for diagnosing these systems. 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 perfectness of the diagnostic process. Below, the presence in the current approach of the items that make fault diagnosis perfect is discussed.

Accuracy

The current process and available tools do not allow the determination of how accurate the diagnosis is. The only way to estimate it, 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, quantification is not possible in the current practice.

Independent of its Environment

Section 2.3 described the actions that employees perform for solving a failure of the power supply. It shows that, nowadays, many diagnostic knowledge depends on cur-
2.4 Optimal Fault Diagnosis

State-of-the-practice Fault diagnosis at PMS

rent employees. Therefore, the reliance on many people is called 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 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. This is done in section 3.6. Also, development costs typically outweigh the runtime costs.

Runtime Costs

The runtime costs of the current diagnostic approach are high compared to other approaches. 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 very adequate at the moment.

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 objections 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. 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. 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 solve this problem. 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. The next chapter shows the possible automated approaches. The items of 2.4 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 manual fault diagnosis approach, currently used, does not meet the desired criteria. 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 needs to be interpreted. Consequently, the knowledge of the system that most troubleshooters is not enough for interpreting those large amount 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.

This chapter gives an overview of the possible automated approaches to fault diagnosis. The first section introduces a logical overview of the categories automated approaches are in. The sections that follow give a view on how a particular approach would work out on the power supply example. Finally, these different approaches are assessed by using the criteria introduced in section 2.4. This chapter is the rationale for choosing a specific automated approach.

3.1 Overview Techniques

Automating the 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 many possible sorts of automated fault techniques. The only scientific attempt trying to provide an overview is done by Dash and Venkata-subramanian [7]. 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 about explaining the (abnormal) behavior of a system.

Consider figure 3.1. It shows the categorization of automated fault diagnosis approaches. It is possible to differentiate between white box and black box techniques (above and below the X-axis). A black box technique only takes input and output of the entire system to account, and no structure or internal working is used. This thesis briefly discusses two black box approaches. Section 3.2 introduces the first, and can be described as:

\(^1\)Note that an automated approach does not exclude the use of the observations that can still - and only - be obtained manually.
3.1 Overview Techniques

**Figure 3.1: Categorization Automated Fault Diagnosis approaches**

**BB-1:** A black box approach that uses historical data. Historical data are observations and diagnoses of (many) previous days that the system was operational. A data mining approach attempts to find correlations between observations and diagnoses made by service engineers (e.g. job sheets).

The other black box approach might not be purely black-box according to some definitions of the black box concept, for it also uses the structure of the system. **Section ??** presents the second black box technique that is discussed in this thesis, and is

**BB-2:** A black box approach that uses system structure. This data mining approach uses passed/failed-statuses of requests to pinpoint to one or more suspected components.

A white box technique distinguishes itself from a black box technique in its use of behavioral knowledge about its internal components (above the X-axis of figure 3.1). It not only uses a constructional decomposition of the system. ("the power supply consists of a PDU, two cables, three low voltage power supplies, five fuses.") but also defines the function for each of its parts ("a fuse cuts of the electrical current, if a abusive condition occurs."). When a white box description is available, it is possible to distinguish between consistency-based and abductive reasoning. The abductive
approach reasons from effect to causes, or - in the context of fault diagnosis - from symptoms to causes. The classical approach to fault diagnosis is in this context.

**CA:** The classical approach. Experts manually define a mapping between symptoms and diagnoses. The mapping is implemented by application-specific code or, more generally, by using expert systems.

A consistency-based approach only uses the more natural way of reasoning from causes to effect. It is discussed in chapter 4, and could be summarized as:

**MBD:** Model-Based fault Diagnosis. Model-Based fault Diagnosis (MBD) is a technique for doing fault diagnosis based on the model of a system. The model defines cause-to-effect relationships based on first principles. The inference mechanism uses this model in combination with real-life observations to isolate the root cause of a failure.

Below, the idea behind each of the above mentioned approaches is shortly clarified by applying them on the power supply example (as introduced in section 2.1). Figure 3.3 repeats the figure, that was previously shown in chapter 2.

### 3.2 Data Mining Approach Using Historical Data (BB-1)

The first data mining approach, that is discussed, searches for correlations between log data and job sheets. Each time a failure occurs two kinds of information become available. Firstly, the log contains observations. Secondly, the service engineer replaces a certain FRU to solve the problem and documents on his/her actions in a job sheet. The idea of this technique is to use these two pieces of information by searching correlations between one another. 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 (as the previous section showed):
A well written data mining program would easily find this (linear) correlation between this observation and a replacement of CableB. Then, a supporting diagnosis engine could use found correlations to build a Diagnosis Lookup Table (analogous to table ??) 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 Classical Approach (CA)

This section presents the classical approach to the fault diagnosis problem, elaborated on the example of the power supply. The idea is to take advantage of experts their knowledge of the system. Given observations on the system, most practically embedded in the log, he is able to draw more conclusions about possible malfunctions than the average service engineer. Let us now consider the power supply example, we used for discussing the current (manual) approach, as depicted in figure 2.2. Suppose the on/off-statuses of the five components on the right of the figure could be known by

<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>

Figure 3.3: Partial architecture of the power supply (repitition of figure 2.2).
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, the symptom is then:

\[
\forall x \in V \mid \neg (FD = TBCB = CRCB = Collimator = Chiller = StartUp)
\]

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 contrary, the expert’s knowledge about the system enables him to distinguish much more symptoms. This way he/she is able to define a mapping of symptoms on more useful diagnoses. The results of this process is shown in table 3.1. It can easily be verified by considering the block diagram of figure 2.2.

This table can then be used each time a failure occurs and serves as a lookup table. Suppose CableB has been broken again. The observation, that could be extracted from the log, would be as follows:

<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 lookup in the table shows that this valuation of the observables corresponds to the entry identified by fault scenario 7. The corresponding diagnosis is that either the PDU or CableB is broken.

### 3.4 Data Mining Approach Using Structure (BB-2)

This section discusses the second data mining approach. 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 a black box approach would work. The outputs can be considered as error detection mechanisms. This supplies us with 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 3.2 shows the traces that a broken CableB yields. This information is then subject to statistical analysis to find possibly malfunctioning components. This statistical technique is called data clustering, and finds the components which are mostly correlated with failures. In the example, LC_PS2 is two times member of a failed trace. However, the PDU and CableB are both three time members of failed traces.
3.5 Model-Based Approach (MBD)  

Automated Fault Diagnosis

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Trace</th>
<th>Passed/Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PDU-CableA-LV_PS1-FuseA-Flat Detector</td>
<td>passed</td>
</tr>
<tr>
<td>2</td>
<td>PDU-CableB-LV_PS2-FuseB-TBCB</td>
<td>failed</td>
</tr>
<tr>
<td>3</td>
<td>PDU-CableB-LV_PS2-FuseC-CRCB</td>
<td>failed</td>
</tr>
<tr>
<td>4</td>
<td>PDU-CableB-LV_PS3-FuseD-Collimator</td>
<td>failed</td>
</tr>
<tr>
<td>5</td>
<td>PDU-CableC-Chiller</td>
<td>passed</td>
</tr>
</tbody>
</table>

Table 3.2: input matrix for data analysis of the example fault scenario

traces, and therefore one of them is most likely to be at fault. This brings us the same result as suggested by an expert (see fault scenario 7 of table 3.1).

3.5 Model-Based Approach (MBD)

Model-Based fault Diagnosis is, like the classical method, a white box technique, but defines the behavior of the system in terms of cause-to-effect rather than effect-to-cause. To do so, it uses a behavioral model of the system (as opposed to just using a structural model, like the technique of the previous section). 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 falsely made 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 (see the next chapter). 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. For each diagnoses a probability can be calculated. These could be used by a service engineer to prioritize his/her diagnostic activities.

### 3.6 Evaluation of the AFD-approaches

Table 3.3 shows an evaluation of the approaches of this chapter, using the items of section 2.4 as criteria.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Accuracy</th>
<th>Speed</th>
<th>Diagnostic Resolution</th>
<th>Independence</th>
<th>Development Costs</th>
<th>Runtime Costs</th>
<th>Adaptability</th>
<th>Explanation Facility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BB-1</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CA</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>BB-2</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</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>
</tr>
</tbody>
</table>

Table 3.3: 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 artefact. 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. Lots of 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
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 behavior rules can be formalized to propositional logic:

\[
\begin{align*}
h_A & \Rightarrow (x \Leftrightarrow \neg w) \\
h_B & \Rightarrow (y \Leftrightarrow \neg x) \\
h_C & \Rightarrow (z \Leftrightarrow \neg w)
\end{align*}
\] (4.1)

This formalization from the concept of 3 inverters to these behavior 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 [TODO].

### Diagnostic Engine

The second artefact 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 known variables; only \( w, y \) and \( z \) are known. These are called the observables of the system, and substituting them 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:

\[
(\neg h_A \lor \neg x) \land (\neg h_B \lor x) \land (\neg h_C \lor \neg x)
\] (4.3)

This can be rewritten to DNF-form:

\[
\neg h_A \neg h_C x \lor \neg h_B \neg x = 1
\] (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 4.3 yields:

\[ (-h_A \lor -h_B) \land (-h_B \lor -h_C) = 1 \] (4.6)

Then, using De Morgan’s Laws:

\[
\neg(-h_A \lor -h_B) \land (-h_B \lor -h_C) = 0 \\
\neg(-h_A \lor -h_B) \lor \neg(-h_B \lor -h_C) = 0 \\
-h_A -h_B \lor -h_B -h_C = 0
\] (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 [8], 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 sYst- tem DIAgnosis), and the corresponding LYDIA toolkit. Other diagnostic systems are GDE of de Kleer [TODO], Sherlock [TODO] and Livingston by Williams and Nayak [TODO]. See [16] 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 artefacts 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. The propositional system of equations of 4.2 can easily be translated to LYDIA. The behavioral rule for one inverter (in proposition logic: \(h \Rightarrow (i \leftrightarrow \neg o)\)) is defined in LYDIA as follows:

```lydia
system inverter(bool i, h, o) {
    h => (i = !o); // If healthy, output equals inverse of the input
}
```
Then, 3 of such inverters can be connected by the following structural description:

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

system inverter3 (w: bool, hA: bool, hB: bool, hC: bool, y: bool, z: bool)
    // Inputs, outputs, and healths are parameterized to
    // to make the definition compositional.
{
    // Define a priori probabilities for all health variables
    probability ( hA = false ) = 0.01;
    probability ( hB = false ) = 0.01;
    probability ( hC = false ) = 0.01;

    // Connect the 3 inverters
    inverter ( w, hA, x);
    inverter ( x, hB, y);
    inverter ( x, hC, z);
}
```

The three lines starting with `inverter` 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. The lines that start with the keyword `probability` specify the probabilities that the inverters are broken. These 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 4.4 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. Appendix TODO presents the use of Lydia tools in this thesis. 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][16][15][9]). 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.
4.2 Model-Based Diagnosis with LYDIA

**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.

The previous writing 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 for creating a solution to the problem: presenting a proof-of-concept of a model-based approach to fault diagnosis, aimed at the Philips Cardio-Vascular X-Ray System. The 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 [11] and [9] is in this area. Currently, the diagnostic engine of LYDIA could operate, within reasonable time, on models that have at most $10^2$ variables.

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 observability, and increase diagnostic accuracy.

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 also 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 4.3 introduces types of models, and ways to take on the modeling problem. In order to optimize the solution to the problem, section 4.4 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, an promising advantage is the following: the metric introduced in section 4.4 allows to study where 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, modeling is the most important issue within the scope of this thesis. The next section introduces its ingredients.
4.3 Modeling

The previous section concluded by pointing out the importance of a good model. Engineers and scientists 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. Scientists try to refine their models, in order to remove differences. Engineers try to search for anomalies, 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 conclusion about the health of components. 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.

Types of models

Information about a system could be described in different ways. There are four types of models:

**Structural Model** Description of the system that only defines the set of components, its interconnections and facts that state which observations fail. No behavioral information is included, but facts whether or not an observation is allowed in healthy system operation should be added. In the 3-inverter example, the behavioral rule for one inverter is not specified, but pass/fail outcomes for each possible observation are added. The new definition becomes:

```plaintext
system inverter(bool i, h, o)
{
    h => (i = o); // If healthy, a correct input results in a correct output
}

system inverter3 (w: bool, hA: bool, hB: bool, hC: bool, y: bool, z: bool)
// Inputs, outputs, and healths are parameterized to
// to make the definition compositional.
{
    // Define a priori probabilities for all health variables
    probability ( hA = false ) = 0.01;
    probability ( hB = false ) = 0.01;
    probability ( hC = false ) = 0.01;

    // Connect the 3 inverters
    inverter ( w, hA, x);
    inverter ( x, hB, y);
    inverter ( x, hC, z);
```

// Define all observation that 'fail'
((y=0) and (z=0)) => !(hA and hB and hC)
((y=0) and (z=1)) => !(hA and hB and hC)
((y=1) and (z=0)) => !(hA and hB and hC)
}

**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. Examples are stuck-at-zero, etc. A strong model of the 3-inverters is:

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

**Abductive Model** MBD diagnosis does not forbid to use abductive description of the systems. 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. A (partial) abductive model, 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)) => {
    !hB
    or (!hA and !hC)
}
```

In theory, the expressive power is the same. The only difference is that way how relevant information is specified. This affects (as explained in chapter 3) time to create the model, it is error-prone, and inflexible in case of a design change. But is some cases there is no other possibility.

**Granularity**

The first step of making a model is deciding the granularity, in other words 'level of detail', of the model. There are two ways to view granularity.
Firstly, in depth, it is possible to tune the level of abstraction. For each system, there are 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 only 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. This is more expensive than a model that specifies the system at a lower level of granularity. 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.

Secondly, in breadth, 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, and environment.

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 use for recovering the system? Does this component ever brake down? Could it state directly or indirectly be observed?)

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.

Discretization of Observations

The output of sensors or other sources of data in a system usually contain a lot of 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. The variables of the few-valued domain are specified by the model, and determine the observability of the system. Discretization removes irrelevant information, in order to decrease complexity of the model. Removing too much resolution decreases diagnostic performance. There is a certain optimum.

The system data of the 3-inverter example, as well as the example of section 4.5, only consists of variables that are already in the boolean domain. The next chapter presents an complex example that shows how a discretization simplifies modeling.

### Compositional of Subsystems

Another important issue is to make subsystems entities compositional. This is one of the characteristics model-based computing. The 3-inverter example presented above is specified in a compositional model. A non-compositional model would be:

```plaintext
system inverter3 ()
{
  // Define a priori probabilities for all health variables
  probability ( hA = false ) = 0.01;
  probability ( hB = false ) = 0.01;
  probability ( hC = false ) = 0.01;

  // Define the behavioral rules for the 3 inverters
  hA => (w = !x);
  hB => (x = !y);
  hC => (x = !z);
}
```

By the way, this is an intermediate step within the LYDIA compiler. It does not change the semantics of the model. Compositional models improve flexibility, allow for reuse, and improve readability. Suppose inverter B is implemented by resistive-drain (that is, using one transistor and one resistor), and inverter B and C use the CMOS implementation of an inverter (using two transistors). For some reason, the supervisory controller is interested if these transistors and resistors are healthy or unhealthy. TO DO: finish

### 4.4 Entropy Gain

This section discusses a metric that can be used to estimate the diagnostic performance of a specific MBD implementation, namely entropy. Entropy is a heuristic, introduced by Shannon [14], to quantify information. The Kleer and Williams suggested to use entropy as a heuristic for quantifying the uncertainty of diagnosis [8]. 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. This allows for comparison of various MBD implementations. In other words, 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. Then, entropy is used to estimate the diagnostic performance of the structural, weak, and strong model of the 3 inverter example.

Best Next Measurements

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 considered points are referred to by input \( x \), intermediates \( a, b, c \), and output \( y \).

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:

\[
p(\neg h_A) = 0.01 \\
p(\neg h_B) = 0.01 \\
p(\neg h_C) = 0.01 \\
p(\neg h_D) = 0.01
\]  

(4.8)

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. So, the number of possible diagnoses is \( 2^4 = 16 \). This information, the existence of 16 possible diagnosis, 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) \( h_A = \text{true}, h_B = \text{true}, h_C = \text{false}, h_D = \text{true} \) 
(0.00970299) \( h_A = \text{true}, h_B = \text{true}, h_C = \text{true}, h_D = \text{false} \) 

... 

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
4.4 Entropy Gain Model-Based Fault Diagnosis

<table>
<thead>
<tr>
<th>Nr.</th>
<th>x</th>
<th>hA</th>
<th>a</th>
<th>hB</th>
<th>b</th>
<th>hC</th>
<th>c</th>
<th>hD</th>
<th>y</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.00000099</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00000099</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00000099</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.00970299</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00000099</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.00970299</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.00970299</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00970299</td>
</tr>
</tbody>
</table>

Table 4.1: Mode catalog of the 4-inverter model, when the observations, \( x=1 \) and \( y=0 \), have been made.

Calculated by using the formula:

\[
p(\{h_A, h_B, h_C, h_D\}) = p(h_A) \ast p(h_B) \ast p(h_C) \ast p(h_D)
\]

(4.9)

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. Let \( i \) be a health vector from the list of diagnosis. The entropy (H) is defined as:

\[
H = - \sum p_i \log p_i,
\]

(4.10)

where \( p_i \) is the probability of health vector \( i \) given a specific set of observations. This is a cost function to estimate the expected cost of identifying the actual candidate. TODO: explain formula.. If no observations have been made, the 4-inverter system has 16 health vectors, and applying the formula for entropy yields: \( H(h) = 0.27 \), where \( h \) is the set of all health vectors.

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-inverters model, as listed in appendix TODO, is a strong model, and the fault mode of 1 inverter is defined by: \( \neg h = \rightarrow (x = i) \). The mode catalog of this 4-inverter model contains 32 entries (16 entries for \( x=0 \) and 16 entries for \( x=1 \), when nothing has been observed. The mode catalog contains 8 entries when the input and output of the 4-inverter system are known, and this mode catalog is shown in table 4.1. It can be obtained from the a priori mode catalog, by removing all entries where \( x \neq 1 \) or \( y \neq 0 \). The resulting number of health vectors is 8. This information, the existence of 8 health vectors that are consistent with \( x=1 \) and \( y=0 \), can be stored in 3 bits, as opposed to the 4 bits that were needed without observing \( x \) and \( y \). The entropy

Consider that \( a \) is being measured. There are two possibilities: \( a=0 \) or \( a=1 \). If \( a=0 \), there are 4 health vectors still consistent with the observations. The mode catalog specifies the a priori probabilities of these health vectors. TODO: finish!
Model-Based Fault Diagnosis

4.5 MBD on the Power Supply

TODO: sequential fault diagnosis [3].

Fundamentals

— short abstract - will be replaced, removed or rewritten to the actual text —
Fundamentals of an entropy study using the four inverter example. Of course the section contains references to [8] and [3].

4.5 MBD on the Power Supply

This section introduces more notions of the MBD theory by means of the example that was also used in the previous chapter.

Summary

...(very short: e.g. the MBD-approach to testing) …There are various techniques for solving this NP-hard search problem [8].
Chapter 5

Diagnosing the Beam Propeller Movement of the Frontal Stand

The text of figure 5.1 has been taken from job sheets. 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\(^2\). 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.

---

\(^1\)“Junction movement” is the old term within PMS for “Beam propeller movement”.
\(^2\)If we assume that faults occur independently.

---

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


The remainder of this chapter clarifies how MBD can help.

The previous chapter presented the basic theory of model-based fault diagnosis. It already included an explanation of MBD on a real system. This trivial example already shows benefits, but the main part of 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.

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.

In this case study, the modeling process uses the log data as starting point. First, the scope of the system that is subject to fault diagnosis should be defined. This is the topic of the first section. The second section presents the fault diagnosis that experts are able to do using the log data; the classical approach to automated fault diagnosis. 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 the classical approach. 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 could achieve the highest entropy gain, by considering various extra measurement points.

5.1 Experimental Methodology

Recall that the problem of this thesis is to present a proof-of-concept of the model-based approach to fault diagnosis, aimed at the Philips Cardio-Vascular X-Ray system. This case study presents a solution to this problem. The goal of this case study is to show that MBD is applicable to an example system of 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 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: 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 $D_{jMBD}$. Finally, let $D_{jreal}$ be the adjudged broken component. The accuracy ($A$) of a MBD implementation is then calculated by the formula:

$$A = \sum (D_{jMBD} = D_{jreal})$$

(5.1)
An alternative metric for diagnostic accuracy, that is described by [5], is not used. 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 lots of 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.

Approach

As explained in the previous chapter, developing a MBD implementation only requires the construction of a model. 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 a particular set of log data.
2. Experts interpret log 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 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. Section 5.6 presents the other way to entropy gain; extending observability of the system.

5.2 Scope Target System

This section defines the scope of the system that is subject to diagnosis. The 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.

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 angles of freedom. There is a subsystem defined, called Geometry, that is responsible for all 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. For this reason, and because of the diagnosis problems referred to in the chapter’s introduction, one of them has been chosen for closer examination.
5.2 Scope Target System

Diagnosing the Beam Propeller Movement of the Frontal Stand

Figure 5.2: Beam Propeller Movement of the frontal stand

The chosen movement is called the *beam propeller movement*, and has been chosen because its FRUs are accessible quite easily.

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. The following describes the error that is logged by the error detection mechanism.

**The error**

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 contained information about the system’s state at the time the error occurred. In this case were are quite lucky: there is a Exception Description 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, and this particular subsystem. Not all components that could cause a failure of the *beam propeller movement* are considered. The considerations to decide if components are included or excluded in the target system, are as follows. For all known

---

3Remark: An automated approach has even more benefits on a system when the components are hard to reach, because the diagnosis 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 Target System

---

<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).

components, within the architecture described in appendix B.1, 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 components generates 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.

If, for a certain component, all answers on these questions are positive, the component is included in the scope of the target system.

Table 5.2 shows the components that are excluded (see appendix B.1 for the place of a component within the architecture). 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 extension has, by far, the highest a priori failure probability, it has been chosen to include this component in the target system.

<table>
<thead>
<tr>
<th>Component</th>
<th>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: The components that are excluded from the diagnostic process, described in this chapter.

47
5.3 Classical Approach

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 should be based on historic data, but for presentation purpose the values are only used to order the list of possible diagnosis (the order 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: The FRUs and their associated health variables with a priori component failure probabilities

5.3 Classical Approach

The approach of this case study on MBD requires that the classical approach is constructed. The acquisition of relevant knowledge for a consistency-based model is the most important show stopper for the success if the model-based approach. By constructing an abductive model, the classical approach to automated fault diagnosis, lots of information can be obtained. In this approach, an expert has to examine log data, and draw conclusion about the correct or incorrect functioning of the components.

What information is used by the expert for constructing the symptom-diagnosis mapping? Most of this information is gained by experience, and this is why this modeling step is very useful for acquiring that ‘hidden’ information.

Table 5.4 shows (part of) the fault scenarios that have been analyzed. The table shows that the system data contains four different fault categories, with corresponding diagnoses. The result of the classical approach is the construction of a diagnosis look-up table. In order to achieve this, the expert has to look for the minimal set of symptoms that identifies a particular fault category. The data of table 5.4 somehow maps upon these symptoms. Human inference of nominal and faulty system behavior
Diagnosing the Beam Propeller Movement of the Frontal Stand

5.3 Classical Approach

<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></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></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></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></td>
<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>
</tr>
<tr>
<td></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></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></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></td>
<td>C3</td>
<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></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></td>
<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>
</tr>
</tbody>
</table>

Table 5.4: Categorizing fault scenarios based on observations. This data has been taken from fault situations as logged in the error 11 messages.

showed that the error signals differentiate between the fault scenarios. So, these are chosen as the symptoms of the fault scenarios. The block diagram of figure 5.3, that we constructed for the MBD approach, validates this conclusion:

1. If \( I_{act} \) and \( I_{set} \) differ more than a certain threshold the \( \text{CURRENT\_ERROR} \) signal becomes true. The control 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 indicates something has gone wrong in the current control loop.

2. If \( V_{act} \) and \( V_{set} \) 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).

3. If \( P_{act} \) and \( P_{set} \) differ more than a certain threshold the \( \text{POSITION\_ERROR} \) becomes true. The position loops consists of LUC Extension (controller), MVR (forward), Motor/Brake, Stand and Potmeter/Encoder and LUC Extension (feedback).

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 the block diagram. Without it you would have to trust the expert’s inference of system behavior.

Diagnosing and Entropy

The identification of the symptoms enables the construction of the table that defines the mapping of these symptoms on diagnoses. Table 5.5 shows this mapping for single faults only.

TODO: entropy is ...
This section presents the MBD implementation that reproduces the results and entropy result of the classical approach. This is a transformation from abductive model to consistency-based model. The first part of this section discusses the model. Then, the diagnosis and entropy are discussed.

**The Model**

In this subsection the model is constructed. What resources of information have been used for its construction?

- The abductive model, as presented by the diagnosis look-up table of table 5.5.
- Documentation about the meaning of the system data (see table B.1 of appendix B)
- Technical drawings.
- Other technical documentation.

The result of the modeling activity is shown in figure 5.3. The block diagram shows the structure of the target system.

**Type of the Model**

TODO: weak fault model
Diagnosing the Beam Propeller Movement of the Frontal Stand

5.4 MBD’s Entropy (MBD-1)

Granularity

TODO: improve! This is one of many possible views on the architecture of the target system, and has been chosen, because it includes relevant aspects of the system. The input of the system is the speed. It is led via other components of the Geometry sub-system to this subsystem. The angular speed of the frontal stand that results is chosen as an output signal of the system. However, these two signals are only added for clarity and do not play a role in the diagnostic process. Note that the fault detection mechanism is part of the architecture. It constitutes of the POSITION_ERROR, SPEED_ERROR, CURRENT_ERROR and 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.

Discretization

TODO: not necessary for these signals.

Compositional

TODO: refer to appendixes, that show uncompositional and compositional versions.

Modeling a Control Loop

To give insight in its construction of the model described above, the writing below describes two important building blocks of the model. 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.

Figure 5.4(a) shows the basic concept. 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:

\[
\text{( h\_c and h\_s ) => (B = A)};
\]

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; 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 S2. Again, the actual value Q equals the setpoint value P if the controller (C2) is functioning correctly, is at its setpoint and the system’s (C1, S1, S2) behavior is conform the prediction. The Lydia code is now:
Figure 5.3: Partial architecture of the FRUs that implement the beam propeller movement. Signals enclosed by brackets are not observable, the others are.
Diagnosing the Beam Propeller Movement of the Frontal Stand

5.4 MBD’s Entropy (MBD-1)

Figure 5.4: Modeling the control loops
5.5 MBD’s Entropy Gain - Current Sensors (MBD-2)

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 and S3) implicate that the actual value Y equals the setpoint value X. This can be described by the following Lydia code:

\[( h_3 \text{ and } h_c2 \text{ and } h_c1 \text{ and } h_s1 \text{ and } h_s2 \text{ and } h_s3 ) \Rightarrow (Y = X) \; \]

Modeling the Error Signal

As said before there is an error detection mechanism in place inside the beam propeller movement. Three 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.

![Diagram](a) One control loop

Figure 5.5: Modeling the error signal

Diagnosing and Entropy

TODO: table showing that each fault scenario has same diagnosis list. = \text{entropy is the same: ...}

5.5 MBD’s Entropy Gain - Current Sensors (MBD-2)

Obviously, a good diagnostic approach takes advantage of every possible inference that could be drawn out of all available observables. There are more symptoms within the examined fault scenarios. Consider table 5.4. The values that the observables of the first eight columns can take are in the infinite domain of integers. It is hard to understand the exact meaning of these integers with respect to the system behavior.
Therefore a clarifying step can be to abstract from these integer values and chose a discrete domain with less members. In this case we defined a domain \( M \):

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

(5.2)

Table 5.6 repeats the examples of fault scenarios from table 5.4. But this time some other variables lifted out and the named variables are in a different domain. See B.4 for the discretization that has been applied. Table 5.6 shows the result of this discretization.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Fault Scenario</th>
<th>Iact,( \mu )</th>
<th>Iset,( \mu )</th>
<th>Iact</th>
<th>Iset</th>
<th>e_sp</th>
<th>Vact</th>
<th>Vset</th>
<th>e,p</th>
<th>C</th>
<th>S</th>
<th>P</th>
<th>PV</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ex. A1</td>
<td>-1784</td>
<td>3942</td>
<td>TL</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>N</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>ex. A2</td>
<td>3985</td>
<td>-4525</td>
<td>TH</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>N</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>ex. A3</td>
<td>1482</td>
<td>-2951</td>
<td>H</td>
<td>TL</td>
<td>H</td>
<td>H</td>
<td>N</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>ex. A4</td>
<td>-5001</td>
<td>3472</td>
<td>TL</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>N</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>B</td>
<td>ex. B1</td>
<td>-4817</td>
<td>-4769</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>TH</td>
<td>L</td>
<td>L</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>ex. B2</td>
<td>-4961</td>
<td>-4912</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>TH</td>
<td>L</td>
<td>L</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>ex. B3</td>
<td>4949</td>
<td>4948</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>TL</td>
<td>H</td>
<td>H</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>ex. B4</td>
<td>-4991</td>
<td>-5003</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>TH</td>
<td>L</td>
<td>L</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>C</td>
<td>ex. C1</td>
<td>358</td>
<td>370</td>
<td>H</td>
<td>H</td>
<td>N</td>
<td>L</td>
<td>L</td>
<td>N</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>ex. C2</td>
<td>-2</td>
<td>102</td>
<td>L</td>
<td>H</td>
<td>N</td>
<td>H</td>
<td>N</td>
<td>N</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>ex. D1</td>
<td>-358</td>
<td>-676</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>N</td>
<td>N</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Table 5.6: One way to discretize the observables.

Notice that some variables of table 5.4 are removed: 1) \( \text{Pact} \) and \( \text{Pset} \) are not usable because a different position does not implicate other behavior. 2) \( \text{Imin} \) and \( \text{Imax} \) are used to calculate the polarity of \( \text{Iset} \) and \( \text{Iact} \). The first two columns show this intermediate step. Clearly, these domain M values are much easier to reason with than the integer values. For example, from the system behavior we expect that the polarity of \( \text{Iset} \) and \( \text{e_sp} \) coincide. However, this does not hold for the third example of what we categorized as a fault scenario A occurrence. Looking at figure 5.3 this indicates the CTR_speed must be at false. So, the LUC_Extension is malfunctioning. This way, we identified fifth fault scenario. Table TODO shows the new diagnoses.

5.6 MBD’s Entropy Gain - Extra Sensors (MBD-3)

MBD can be used as a probing strategy for extra sensors...as section TODO .... to do ...
5.7 results

TODO: diagnostic accuracy not possible. Only for these 3 fault scenarios. SHOW.

However, entropy...blabla Model-Based approach achieves the same entropy as experts are able to achieve, using only the current set of sensors. Improving the model, but using the same set of sensors, leads to an entropy gain of .... Furthermore, examining the placement of extra sensors could predict potential extra entropy gains of ..., ..., and ...
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 be used outside their original specifications. There is a complex dynamic interaction between subsystems, and this is what makes fault diagnosis difficult and time consuming. Philips Cardio-Vascular X-Ray system is such a system. It also includes an increasing amount of third party components. The effort put in diagnosing is obviously increasing, as the portion of the PMS employees working on service instead of development grows. The diagnostic process aimed at the Philips Cardio-Vascular X-Ray system has been as a case study for solving . . .

6.1 Reflection

- missing link between observations and real diagnoses
- it all comes down to entropy

- conclusions on "case-study-level". - generalization to "PMS-level"-statements. - generalization to "Embedded Systems-level" statements.

To do . . .

6.2 Recommendations

To do . . .
Bibliography


Appendix A

Glossary of terms

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

**Arm:** Distance between the centre of buoyancy and the lever's fulcrum.

**Beam propeller movement:** . . .

**C/V:** Cardio Vascular. The business unit within PMS responsible for the Cardio-Vascular solutions.

**Error:** The difference between desired and actual performance [?].

**Extension:** . . .

**Failure:** The lack of ability of a component, equipment, sub system, or system to perform its intended function as designed. Failure may be the result of one or many faults [?].

**Fault:** An abnormal condition or defect at the component, equipment, or sub-system level which may lead to a failure [?].

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

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

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

**Geometry:** . . .

**Host:** . . .

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

**Image Processing (IP):** . . .
**Glossary of terms**

**Image Detection (ID):**  

**Jobsheet:**

**Junction movement:** See *beam propeller movement*.

**LUC:** Local Universal Controller

**Modality:**

**MVR:** Motor Voltage Regulator

**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.

**PCI:**

**Philips Cardio-Vascular X-Ray system:**

**Service Innovation:**
Appendix B

Case Study Details

In this appendix details of the case study described in chapter 5 can be found.

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 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.

![Diagram of the parts of the frontal stand](image)

**Figure B.1:** The parts of the frontal stand
### B.2 Used Target System Documentation

<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>

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

**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.2 Used Target System Documentation

### 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. Read appendix D for a brief introduction on the required (control) theory. 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. Gravity exerts torques on both the left (torque t1) and right (torque t2) lever of the frontal stand (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 CTR_speed to enable the controlling of the speed. This controller uses a table to determine what current it has to set (I_{set}) in order to achieve a certain acceleration at a particular position \(^{1}\). So, the difference between the speed setpoint (V_{set}) and the actual speed (V_{act}) results in a certain desired acceleration (e_{sp}). This acceleration is combined with the current position (P_{act}) in order to lookup the associated current setpoint I_{set}. Here starts the third and final control loop. It is responsible for controlling the current at setpoint I_{set}. In order to do so it requests a certain current of the MVR\(^{2}\) that has a circuit with the Motor/Brake. The actual current (I_{act}) on this circuit is measured by means of a ampere meter and led back to the current controller CTR_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 MVR\(^{2}\). And by leading the signal V_{act\_analog} back to the controller (V_{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 potentiometer and encoder (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 do error detection). The LUC_extension\(^{2}\) contains an AD-converter to digitize the analog signal and lead it back to the controller (LUC_extension(controller)). The position controller will use this value (P_{act}) once again to calculate the (nominal) position error.

### B.4 Discretization MBD-2

So, I_{act}, I_{set}, e_{sp}, V_{act}, V_{set}, e_{p}, C, S, P, PV are all in the domain M. And C, S, P, PV are in the boolean domain. The mapping of the integer values on the domain M values is defined as TODO: TO APPENDIX???? follows:

\(^{1}\)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 MVR denote the same FRU. It is only a logical decomposition. The same holds for the LUC Extension FRU.
B.4 Discretization MBD-2 Case Study Details

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

\[ I_{\text{set INT}} = I_{\text{set INT}} - I_{\text{constant INT}} \]

\[ I_{\text{lact INT}} = I_{\text{lact INT}} - I_{\text{constant INT}} \]

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

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

\[ e_{sp} = \begin{cases} 
TL, & \text{if } e_{sp} \geq T H V, vset \in [N,H,T H]; \\
L, & \text{if } e_{sp} \in [\infty, -500], e_i < T H V; \\
N, & \text{if } e_{sp} \in [-500, 500], e_i < T H V; \\
H, & \text{if } e_{sp} \in [500, \infty], e_i < T H V; \\
T H, & \text{if } e_{sp} \geq T H V, vset \in [L,T L]; 
\end{cases} \]

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

\[ V_{\text{set}} = \begin{cases} 
L, & \text{if } V_{\text{set INT}} \in [\infty, -500], e_i < T H V; \\
N, & \text{if } V_{\text{set INT}} \in [-500, 500], e_i < T H V; \\
H, & \text{if } V_{\text{set INT}} \in [500, \infty], e_i < T H V; 
\end{cases} \]

\[ e_{p} = \begin{cases} 
TL, & \text{if } e_{p} \geq T H P, vset \in [N,H,T H]; \\
L, & \text{if } e_{p} \in [\infty, -500], e_i < T H P; \\
N, & \text{if } e_{p} \in [-500, 500], e_i < T H P; \\
H, & \text{if } e_{p} \in [500, \infty], e_i < T H P; \\
T H, & \text{if } e_{p} \geq T H P, vset \in [L,T L]; 
\end{cases} \]
Appendix C

Lydia models

In this appendix some lydia models are listed:

Control loop

```lydia
system control_loop
(
    bool A, B,  \/// inputs
    bool ERROR,  \/// output error
    bool h_c, h_s  \/// health variables
) {
    attribute observable (ERROR) = true;

    (h_c and h_s) => (B = A);
    ERROR = (A != B);

    // health Extension
    attribute health(h_c) = true;
    attribute probability(h_c) = h_c ? 0.99 : 0.01;
    attribute health(h_s) = true;
    attribute probability(h_s) = h_s ? 0.95 : 0.05;
}

Beam propeller movement (not compositional)

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
) {
    // intermediate variables
    bool Iact, Iset, Vact, Vset, Pact, Pset;
```
Lydia models

// observables that can be extracted from the 'error' logentry
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;

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

// errors signals (in software, so independent of any health)
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;

// 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_Sand) = true;
attribute probability(h_Sand) = h_Sand ? 0.95 : 0.05;
attribute health(h_PEU) = true;
attribute probability(h_PEU) = h_PEU ? 0.96 : 0.04;

} 

Beam propeller movement (100% observational/ not compositional)

system FS_Beam_Propeller_Movement (}
Lydia models

bool Iact, Iset,
bool Vact, Vset,
bool e_pos,
bool I_mvr, I_to_motor, I_from_motor, I_act_analog,
bool V_mvr, V_to_motor, torque, real_speed, V_act_analog,
bool P_ctr, real_position, P_act_analog,
bool CURRENT_ERROR, SPEED_ERROR, POSITION_ERROR,
bool POSVAL_ERROR,
bool h_EXT_in, h_EXT_out, h_MVR_forward, h_MVR_backward, h_MotorBrake, h_Stand, h_PEU
)
{
  // intermediate variables
  bool Pact, Pset;

  // observables that can be extracted from the 'error 11'-logentry
  attribute observable (Iact, Iset) = true;
  attribute observable (CURRENT_ERROR) = true;
  attribute observable (Vact, Vset) = true;
  attribute observable (SPEED_ERROR) = true;
  attribute observable (e_pos) = true;
  attribute observable (POSITION_ERROR) = true;
  attribute observable (POSVAL_ERROR) = true;

  // possible additional observables
  attribute observable (I_mvr, I_to_motor, I_from_motor, I_act_analog) = true;
  attribute observable (V_mvr, V_to_motor, torque, real_speed, V_act_analog) = true;
  attribute observable (P_ctr, real_position, P_act_analog) = true;

  //--------------- definition behavior

  // current loop
  h_EXT_out => (I_mvr = Iset);
  h_MVR_forward => (I_to_motor = I_mvr);
  h_MotorBrake => (I_from_motor = I_to_motor);
  h_MVR_backward => (I_act_analog = I_from_motor);
  h_EXT_in => (Iact = Iact_analog);

  // speed loop
  h_EXT_out => (V_mvr = Vset);
  h_MVR_forward => (V_to_motor = V_mvr);
Lydia models

h_MotorBrake => (torque = V_to_motor);
h_Stand => (real_speed = torque);
h_MVR_backward => (Vact_analog = real_speed);
h_EXT_in => (Vact = Vact_analog);

// position loop
(!CURRENT_ERROR and !SPEED_ERROR) = Pset;
h_Stand => (Pset => real_position);
h_PEU => (Pact_analog = real_position);
h_EXT_in => (e_pos = !Pact_analog);

// error signals
CURRENT_ERROR = (Iact != Iset);
SPEED_ERROR = (Vact != Vset);
(Pact != Pset) => POSITION_ERROR;

e_pos = (Pact != Pset);  // by definition
h_EXT_in => (POSITION_ERROR => (e_pos or !h_PEU))

POSVAL_ERROR => !h_PEU;

/* ------------ health variables */

attribute health(h_EXT_out) = true;
attribute probability(h_EXT_out) = h_EXT_out ?
    0.97 : 0.03;
attribute health(h_MVR_forward) = true;
attribute probability(h_MVR_forward) =
    h_MVR_forward ? 0.98 : 0.02;
attribute health(h_EXT_in) = true;
attribute probability(h_EXT_in) = h_EXT_in ? 0.97
    : 0.03;
attribute health(h_MVR_backward) = true;
attribute probability(h_MVR_backward) =
    h_MVR_backward ? 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;
Appendix D

Control theory

In this appendix some the basics of control theory are explained. To do (one page) . . .