Decision Theoretic Troubleshooting Using Bayesian Networks

Guided troubleshooting of a diesel injection system

H E L E N A S U N D B E R G

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2008

Decision Theoretic Troubleshooting Using Bayesian Networks

Guided troubleshooting of a diesel injection system

H E L E N A S U N D B E R G

Master's Thesis in Computer Science (30 ECTS credits)
at the School of Engineering Physics
Royal Institute of Technology year 2008
Supervisors at CSC were Magnus Rosell and Henrik Eriksson
Examiner was Johan Håstad

TRITA-CSC-E 2008:026
ISRN-KTH/CSC/E--08/026--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se

Abstract

This thesis presents a study of how the theory of decision theoretic troubleshooting can be applied to the troubleshooting of a truck and the issues attached to this problem. The main objective of the project, which involves two other theses, is to find a cost-efficient approach to troubleshooting trucks with newly developed complex systems.

This study is concentrated on development and evaluation of a troubleshooter prototype. The prototype was evaluated by simulating a repair process of a new diesel injection system, here modelled by a Bayesian network. Due to objectives of adapting the existing theory of troubleshooting to the specific troubleshooting of a truck, some relaxations from the original assumptions were made. These relaxations were then used in the evaluation of the performance of the troubleshooter prototype which was evaluated by a simulated repair process where the prototype was used to find the faulty component. The costs of the simulated repairs were computed and compared to the costs of two other predefined ways of choosing an order in which to repair components. Considering the average costs for the repair procedures, the prototype presented the lowest average cost in three out of five simulation cycles.

Referat

Kostnadseffektiv felsökning med Bayesianska nätverk

Guidad felsökning av ett dieselinsprutningssystem

Detta examensarbete är en studie av hur beslutsteori och sannolikhetsteori kan användas för att felsöka en lastbil. Examensarbetet är, tillsammans med två andra examensarbeten, en del i ett större projekt med målet att hitta ett kostnadseffektivt sätt att felsöka lastbilar där de inbyggda systemen blir mer och mer komplexa.

Detta arbete koncentrerades på utveckling och utvärdering av en programprototyp för felsökning. Prototypen utvecklades för felsökning av ett dieselinsprutningssystem som modellerats med hjälp av ett Bayesianskt nätverk. Som ett led i anpassningen till felsökning i lastbilar, har vissa antaganden i den existerande teorin relaxerats och prototypen är utvärderad efter dessa relaxeringar. Prototypens funktion utvärderades genom att simulera en reparationsprocess där programmet kunde användas för att söka efter ett simulerat fel. Simuleringar gjordes för tre olika sätt att sortera komponenterna i en reparationsordning för vilken en fiktiv reparationskostnad beräknades. I jämförelsen mellan de olika sorteringarna visade sig programprototypen ge den lägsta medelkostnaden i tre av fem simuleringscykler.

Preface

This report describes a master's thesis carried out at the School of Computer Science and Communication (CSC) at the Royal Institute of Technology in Stockholm. The work was performed during the spring and summer of 2007 and corresponds to 20 academic points. This thesis is part of a project together with two other master's theses and the work was carried out in collaboration with Katja Lotz and Jonatan Mossberg. The mandator was Scania CV AB in Södertälje and supervisor at Scania was Anna Pernestål. Supervisors at CSC were PhD student Magnus Rosell and Associate Professor Henrik Eriksson and examiner was Professor Johan Håstad.

Acknowledgements

First I would like to thoroughly thank my co-workers Katja Lotz and Jonatan Mossberg for support, encouragement and great work. Jonatan, for all our intense discussions and for your wonderful humour, always making me laugh. Katja, for being able to understand me. Not only my way of reasoning but also my constant need for food. It has been a great pleasure working with both of you.

I want to thank Magnus Rosell, Henrik Eriksson and Johan Håstad, my supervisors and examiner at KTH, for detailed and insightful comments upon my report and my supervisor at Scania, Anna Pernestål, for valuable feedback and long and interesting discussions. Thanks to Jonas Biteus and Mattias Nyberg for good comments and many questions that helped develop the work, and to all other Scania employees that in different ways have helped me in the course of the work.

A special thank you to Klas Telborn and Tomas Flink for all the time spent answering questions and writing excel sheets and for great patience when something had to be done all over again.

And to Johan, for reading and correcting my writing but most importantly for making the problem with missing index files become just a minor issue. Thank you.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Objectives . . . 2
- 1.3 Report Outline . . . 2

2 Introduction to Bayesian Networks **5**
- 2.1 The Bayesian Approach to Probabilities . . . 5
- 2.2 Bayes' Theorem . . . 6
- 2.3 The Structure of Bayesian Networks . . . 6
- 2.4 Probabilities in the Network . . . 8
- 2.5 The State of Knowledge . . . 8
- 2.6 Conditional Probability Tables . . . 9

3 Decision Theory **13**
- 3.1 The Decision Maker and the State of Nature . . . 13
- 3.2 Maximising Payoff . . . 14
- 3.3 Bayes' Decision Rule . . . 14
- 3.4 The Introduction of Experimentation . . . 14
- 3.5 Decision Making Applied to this Thesis . . . 14

4 The Troubleshooting Problem **17**
- 4.1 Decision Theoretic Troubleshooting . . . 17
- 4.2 Actions . . . 17

5 Theory Of Expected Cost Of Repair **19**
- 5.1 General Assumptions . . . 19
- 5.2 Expected Cost of Repair . . . 20
- 5.3 The Optimal Sequence . . . 21
- 5.4 Expected Cost of Repair Including Tests . . . 22
- 5.5 The Value of Information . . . 23

6 Algorithms for Troubleshooting **25**
- 6.1 The Greedy Algorithm . . . 25
- 6.2 The One Step Look Ahead Algorithm . . . 26

6.2.1 A Myopic Approach 26
6.2.2 General Description of the One Step Algorithm 27

7 Costs **29**
7.1 Independent and Dependent Costs 29
7.2 Level of Disassembly 29
7.3 Costs of Transition 31
7.4 Functional Control 31
7.5 Determining a Strategy using Dependent Costs 31
7.6 The Expected Cost of Repair using Dependent Costs 32

8 The Extreme High Pressure Injection System **35**
8.1 The Diesel Engine 35
8.2 Operating Principle of the XPI System 37
8.2.1 Fuel Transport in the XPI System 37
8.2.2 On-Board Diagnosis and Diagnostic Trouble Codes 38

9 A Model of the XPI System **41**
9.1 Components 41
9.2 Observability 42
9.3 The Single Fault Constraint Node 43
9.4 Diagnostic Trouble Codes 43
9.5 The Bayesian Model of the XPI System 44
9.6 Determining the Levels of Disassembly 44
9.6.1 Level of Disassembly 44
9.6.2 Transition Costs 45

10 Simulating the Repair Process **47**
10.1 Simulation Objectives 47
10.2 Component Sorting Criteria 47
10.3 Carrying Out the Simulations 48
10.3.1 Simulating Component Failure 48
10.3.2 Simulating Test Outcome 49
10.4 Simulation Results 50
10.4.1 Simulation Cycle A - Component 15 Faulty 50
10.4.2 Simulation Cycle B - Component 16 Faulty 53
10.4.3 Simulation Cycle C - Component 17 faulty 56
10.4.4 Simulation Cycle D - Component 4 Faulty 58
10.4.5 Simulation Cycle E - Component 9 Faulty 60
10.4.6 The Average Cost of Repair 61
10.5 Conclusions 62
10.5.1 Importance of Test Outcome 62
10.5.2 Specificity of DTCs 63

11 Discussion and Future Work **65**
11.1 Performance of the ECR Theory 65
11.2 Relaxations of General Assumptions 65
11.2.1 Dependent Costs 65
11.2.2 Performance of a Functional Control 66

Bibliography **67**

Appendices **68**

A List of Notation **69**

B Basics of Probability Theory **71**
B.1 The Basic Rules 71
B.2 The Joint Probability 72
B.3 The Marginal Probability 72
B.4 The Chain Rule 72

C List of Diagnostic Trouble Codes **73**

D List of Tests **75**

List of Figures

2.1 A simple Bayesian network 7
2.2 The Bayesian network for Example 1. 9

6.1 Flowchart of the Greedy algorithm 26
6.2 Flowchart of the One Step Look Ahead algorithm 28

7.1 Levels of disassembly in a general case. 30

8.1 A schematic picture of a cylinder and the important parts 36
8.2 A schematic picture of the four strokes of a diesel engine 36
8.3 A schematic picture of the XPI system 38
8.4 A sketch of the on-board diagnosis. 38

9.1 The model of the XPI system as a Bayesian network 44
9.2 Transition costs for simple transitions in the truck. 45

10.1 The unit interval in relation to the marginalized test probability 49
10.2 The active model for simulation A 51
10.3 The cost distributions for simulation cycle A 52
10.4 The active model for simulation B 54
10.5 The cost distributions for simulated repair of component 16 55
10.6 The active model for simulation C 56
10.7 The cost distributions for simulated repair of component 17 57
10.8 The active model for simulation D 58
10.9 The cost distributions for simulated repair of component 4 59
10.10 The active model for simulation E 60
10.11 The cost distributions for simulated repair of component 9 61

List of Tables

2.1 The CPT for the node S in Example 1. 9
2.2 The CPT for the node H in Example 1. 9
2.3 The CPT for the node B in Example 1 10

7.1 Level of disassembly for the components in Example 2 33

9.1 Levels of disassembly at which repairs and observations can be made. . 46

10.1 The sorting criteria used for the different simulations 48
10.2 Components and corresponding DTCs used in the simulations. 49
10.3 Generated test outcome by a random variable, r. 50
10.4 The obtained repair sequences in simulation cycle A. 52
10.5 The obtained repair sequences in simulation cycle B. 54
10.6 The average cost of repair for all simulation cycles 62

Chapter 1

Introduction

To introduce the reader to this master's thesis and to the report this chapter gives a brief view of the background and a description of the thesis outline.

1.1 Background

Modern truck development contains as much advanced computer programming as it contains construction and design of mechanical parts. All systems in a truck, e.g. the brake system and the hydraulic system, are controlled by control units, designed to discover malfunctioning behaviour. These control systems make the operating diagnosis effective. As a consequence of computerised diagnosis, the repair process at the workshop becomes more aggravated since the cause of a failure may be difficult to find without help from specially developed software to retrieve information from the control systems.

An extension of such software, which does not only retrieve information from the truck but also acts as a help in the repair process, is of great interest to the truck industry as a means to obtain a more efficient repair procedure. Guided troubleshooting, which means searching for a cause of failure following a pre-defined strategy, can be a competitive edge since it contributes to efficient repair and thereby good customer service. Guided troubleshooting may even become a necessity in the future since many new systems tend to be more and more complex.

Troubleshooting software forms the interface between parts of the mechanical devices in the truck, the electrical diagnosis system and the mechanic at the workshop. The software must be designed to handle both disturbances in diagnosis data and the risk of human errors occurring in the mechanic's interpretation of the software output. To create such a program, deep knowledge of the mechanical system in question must be used.

1.2 Objectives

This thesis is part of a project aiming to construct a prototype for guided troubleshooting of a diesel injection system. The goal of the troubleshooting program is to minimise the expected cost of repairing a faulty truck. The study is concentrated on evaluations of algorithms adapted to theory of troubleshooting [Heckerman et al. 1995] using a different approach to costs of repair. In parallel to this thesis two complementary studies were made; an evaluation of Bayesian modeling of a diesel injection system [Mossberg 2007] and a study of how tests can be implemented into the troubleshooting process [Lotz 2007].

The main objectives of this thesis are to

- Adapt existing troubleshooting theory to the troubleshooting of trucks.
- Present and discuss an application on including costs into troubleshooting theory.
- Study the performance of a troubleshooting algorithm with respect to cost-efficiency, using a different cost approach.

1.3 Report Outline

This thesis can be divided into three parts. The first part contains the basic theory of Bayesian networks, decision theory and the troubleshooting problem. The middle part describes the cost approach used in this thesis followed by an introduction and thereafter the modelling of the Extreme High Pressure Injection system. The last part presents the results from the simulations of a truck repair procedure.

First we present the theory of this thesis. Chapters 2 and 3 are to make the reader aquainted to the basics of Bayesian networks and decision theory. The theory of Bayesian networks is based on probability theory and some readers might consider it useful to read the basics of probability theory in Appendix B. Chapter 4 introduces the concept of troubleshooting and its mathematical formulation. In this thesis the cost of the troubleshooting steps are important and one theory for this approach, the expected cost of repair, is presented in Chapter 5. An introduction to some algorithms based on the expected cost theory is found in Chapter 6.

Part two begins with Chapter 7, which describes how the costs of repairing a truck are handled in this thesis and how this differs from existing theory. Chapter 8 presents the diesel injection system in a Scania truck and the Bayesian model of this injection system, used in the computations in this thesis, is presented in Chapter 9.

The third part contains the results of the evaluations of the troubleshooter prototype. Chapter 10 presents and discusses the results of the simulations aimed to study whether the expected cost theory is cost-efficient or not. Discussion and suggestions for future work are found in Chapter 11.

Chapter 2

Introduction to Bayesian Networks

Bayesian networks are graphical models used for probabilistic reasoning.
In [Murphy 2001] Michael Jordan is cited describing Bayesian networks as "a marriage between probability theory and graph theory, providing a natural tool for dealing with two problems that occur throughout applied mathematics and engineering - uncertainty and complexity".

The injection system of a diesel engine is a complex and complicated system. It consists of many different parts with dependencies difficult to grasp. When a symptom of a fault occurs, we cannot be sure of which component is broken. Instead, we can estimate how probable different components are to be faulty. In order to create a troubleshooter prototype for the injection system, we need a model that handles both the high complexity and the uncertainty. Fortunately, Bayesian networks have the ability to handle probabilistic reasoning in an efficient way, even when systems get more complicated.

2.1 The Bayesian Approach to Probabilities

In order to understand Bayesian networks we have to understand the Bayesian way of approaching probabilities. Following Heckerman, the Bayesian probability of an event E, $P(E)$, can be considered as a person's *degree of belief* in that event. Instead of making (infinitely) many experiments to estimate the probability of an event, the Bayesian probability is assigned with respect to the knowledge at hand [Heckerman 1995].

Heckerman points out that the Bayesian definition of probabilities as degrees of belief has been criticised because it seems arbitrary [Heckerman 1995]. This is discussed somewhat further by Pernestål [Pernestål 2007]. Pernestål refers to Jaynes [Jaynes 2001], who has thoroughly discussed this way of viewing probability.

Jaynes treats probability from the point of three basic requirements, called the *Basic Desiderata*. These are:

1. Probability should be represented by real numbers.
2. Probability should qualitatively correspond to common sense.
3. Probability must satisfy the following criteria of consistency
    a) If a probability can be derived in more than one way, then every possible way must lead to the same result.
    b) All evidence relevant to a question must always be taken into account and no information can be arbitrarily ignored.
    c) Equivalent states of knowledge must be represented by equivalent probabilities.

Jaynes shows that the laws of probability (c.f. Appendix B) can be derived directly from these desiderata. This supports the Bayesian view of probabilities as degrees of belief and, conversely, this indicates that probability can be used to measure beliefs [Pernestål 2007, Jaynes 2001].

2.2 Bayes' Theorem

One of the most frequently used probability formulas is Bayes' theorem. It can be used to compute $P(X = x|Y = y)$ from $P(Y = y|X = x)$ and is useful when computing probabilities in Bayesian networks.

$$P(X = x|Y = y) = \frac{P(Y = y|X = x)P(X = x)}{P(Y = y)} \tag{2.1}$$

Bayes' theorem is, as Pernestål states, fundamental in probabilistic reasoning [Pernestål 2007]. The case of troubleshooting a malfunctioning truck is no exception. With only one or a few observations at hand, we want to determine which component is faulty. To do this we want to compute the probabilities of the different faults being present, given the observations.

Typically it is easier to determine the probability that a fault causes a certain observation, rather than the probability that a certain fault is present given an observation. Bayes' theorem enables us to use the information easier obtained to determine the probability we are actually interested in [Pernestål 2007].

2.3 The Structure of Bayesian Networks

A Bayesian network is a directed acyclic graph (DAG) which can be used to model probabilistic relationships between variables. The network consists of *nodes*, each

representing a variable, and directed *edges* which connect the nodes. Each variable is assigned to be discrete and have a finite set of mutually exclusive states. The edges are most often directed from cause to effect [Jensen 2001].

In a directed acyclic graph there is at least one node which does not receive input from any other node. This node is often called the *root*. There must also be at least one node which does not give input to any other node, and nodes of this type are often called *leaves*. The direction of the edges implies a dependence between the nodes such that the state of a node N depends on the state of its *parents*, $Pa(N)$ [Jensen 2001]. Acyclic graphs have, as the name suggests, no cycles. This means that there exists no directed path that ends in its own starting point [Jensen 2001].

The small Bayesian network in Figure 2.1 is a simplified example of a Bayesian network used in this thesis. In this network there is a problem-defining node, labelled *Alarm*. This node tells us whether the system is working or not. The problem defining node in this case can be in one of two possible states; *yes*, if the system is working properly, and *no* if it exhibits some faulty behaviour.

The two nodes *Comp 1* and *Comp 2* are called cause nodes because they have a causal (physical) impact on the state of *Alarm*. *Comp 1* also affects the node *Test 1*. These dependencies are illustrated by the directed edges.

The direction of each edge indicates how the physical influence can spread between the nodes connected by the edge. The physical influence can only spread in the direction given by the edges, and not in the opposite way; performing *Test 1* does not change the actual state of the node *Comp 1*.

For changes in probability the situation is different from that of physical influence. Probabilities can spread in both directions of an edge. This is easy to understand by considering the case where the outcome of *Test 1* indicates that it is very unlikely that the node *Comp 1* is causing the failure. Our state of knowledge is changed and the marginalised probabilities for *Comp 1* and *Comp 2* change, which implies that changes in probability can travel in both directions of the edges.

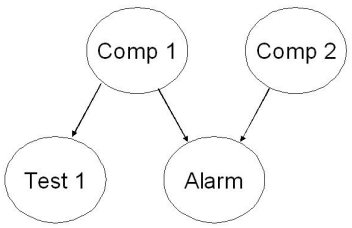

Figure 2.1. A basic Bayesian network consisting of two cause nodes, one problem defining node and one test node.

2.4 Probabilities in the Network

When using a Bayesian network the objective is to draw conclusions about the probabilities that variables take on certain values, given the current state of knowledge, ϵ. The state of knowledge will be discussed further in Section 2.5. Assuming that there are K discrete random variables $X_i, i = 1, \ldots, K$ this probability can be written as:

$$P(X_1, X_2, \ldots, X_K|\epsilon) = \prod_{i=1}^{K} P\left(X_i|Pa(X_i)\right) \tag{2.2}$$

To be able to compute (2.2) all dependencies and underlying probabilities need to be specified. As stated above, the parents of a node i, denoted $Pa(X_i)$, are the nodes that X_i is dependent on. For each node the probability that the variable takes on different values, given the values of its parents and the current state of knowledge, must be given.

Let us introduce some notation that will be used in the diagnosis problem. With the assumption that there are K components in the system, we define $\mathcal{K}$ as the set containing all components. We can then define $\mathcal{F}$ as a subset of $\mathcal{K}$, consisting of the F components that might be faulty. The remaining $N = K - F$ components are assumed to be normally functioning, and are said to belong to the set $\mathcal{N}$. They will not be examined during the troubleshooting, because they are known to function correctly. We represent this by setting the probability of these components being faulty to zero:

$$P(X_i \text{ faulty}) = 0 \qquad \forall X_i \in \mathcal{N}$$

2.5 The State of Knowledge

In the context of probabilistic inference in Bayesian networks, our *state of knowledge*, denoted ϵ, is important. The variable ϵ contains all the information gathered so far during the inference process. The state of knowledge can contain any kind of background information available to us about the system, but also information collected by performing different kinds of information gathering actions. In this thesis, these information gathering actions are observing components, performing tests and repairing components.

When the troubleshooting process starts, we have retrieved information about the state of the truck from the diagnosis system. From the Bayesian network we get information about the components that may have caused the faulty behaviour, namely the components belonging to the set $\mathcal{F}$. This means that we also know which components belong to $\mathcal{N}$. Into the variable ϵ we insert the knowledge about the N non-faulty components. As we proceed with the troubleshooting, during which tests are performed and components are found to be functioning properly, the value of ϵ changes as more information is obtained.

2.6 Conditional Probability Tables

When dealing with discrete stochastic variables, the conditional probabilities of different variables are presented in a *Conditional Probability Table (CPT)*.

To understand how CPTs are used we will present an example where probabilistic reasoning is used in an everyday situation. This example is found in [Pernestål 2007]. The Bayesian network used in the example is shown in Figure 2.2.

The network consists of three nodes, *Sun (S)*, *Holiday(H)* and *Biking (B)*. They are defined as:

S = 'The sun is shining'
H = 'Jane is on holidays'
B = 'Jane goes biking'

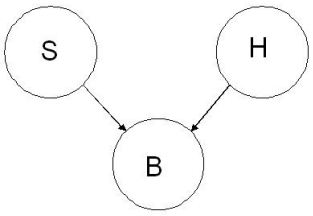

Figure 2.2. A simple Bayesian network, describing the dependencies between the weather and the facts that Jane is on holidays and that she goes biking.

The CPTs for S and H are shown in Tables 2.1, 2.2:

S	$P(S)$
True	0.5
False	0.5

Table 2.1. The CPT for the node S.

H	$P(H)$
True	0.2
False	0.8

Table 2.2. The CPT for the node H.

Tables 2.1, 2.2, 2.3 and the Bayesian network in Figure 2.2 represent our knowledge of the situation.

S	H	B	$P(B\|S,H)$
True	True	True	0.8
False	True	True	0.4
True	False	True	0.7
False	False	True	0.1
True	True	False	0.2
False	True	False	0.6
True	False	False	0.3
False	False	False	0.9

Table 2.3. The CPT for the node B. The states of B are dependent on the states of its parents S and H. The probabilities in the rightmost column are the conditional probabilities of the states of B given the states of S and H.

We see that it is less likely that Jane is on holidays (P=0.2) than the opposite (P=0.8), and that it is as likely that the sun is shining as that it is not (P=0.5).

If we use the convention that X indicates true and $\bar{X}$ indicates false for an event X, we can then use marginalisation (c.f. Appendix B) to compute the probability that the sun *is shining*, after learning that Jane did go biking today:

$$P(S|B) = \frac{P(B|S)P(S)}{P(B)} = \frac{P(B|S,H)P(H)P(S)}{P(B)} + \frac{P(B|S,\bar{H})P(\bar{H})P(S)}{P(B)} = \\ = \frac{(0.8 \cdot 0.2 + 0.7 \cdot 0.8)0.5}{P(B)} = \frac{0.36}{P(B)} \quad (2.3)$$

In this we use Bayes' Theorem (Equation (2.1)) and marginalisation (B.8) over the values of H. In the same way we compute the probabilitity that the sun *is not shining* today:

$$P(S|B) = \frac{P(B|\bar{S})P(\bar{S})}{P(B)} = \frac{P(B|\bar{S},H)P(H)P(\bar{S})}{P(B)} + \frac{P(B|\bar{S},\bar{H})P(\bar{H})P(\bar{S})}{P(B)} = \\ = \frac{(0.4 \cdot 0.2 + 0.1 \cdot 0.8)0.5}{P(B)} = \frac{0.08}{P(B)} \quad (2.4)$$

From the two expressions (2.3) and (2.4) we can now compute the denominator $P(B)$, because these two expressions must sum to one:

$$1 = P(S|B) + P(\bar{S}|B) = \\ = \frac{0.36}{P(B)} + \frac{0.08}{P(B)} = \frac{0.44}{P(B)} \quad (2.5)$$

This gives that $P(B) = 0.44$. Thus, learning that Jane did go biking today has changed the probability that the sun is shining from $P(S) = 0.5$ to

$$P(S|B) = \frac{0.36}{0.44} \approx 0.82. \quad (2.6)$$

When the Bayesian network is known there are efficient methods for computing any conditional probabilities in the network [Pernestål 2007].

Chapter 3

Decision Theory

A repair procedure can be a complex task since the cause of a failure can be hard to find. When repairing a device a number of choices have to be made, e.g. where to start and if to perform some kind of test that can indicate what is causing the problem. The way these choices are made can hasten or delay the repair procedure and in the search for an efficient repair procedure it might be interesting to consider these choices theoretically.

Decision theory divides decision making into two parts; decision making *with* and *without experimentation*. The former includes the possibility of performing tests to reduce the level of uncertainty in a choice. Hillier and Lieberman
[Hillier & Lieberman 2005] discuss, among others, two different decision criteria, the Maximum Payoff criterion and Bayes' decision rule. Bayes' decision rule is the most commonly used, its advantage being its ability to incorporate all the available information [Hillier & Lieberman 2005].

3.1 The Decision Maker and the State of Nature

In a decision making situation there is always a *decision maker*, who must choose an alternative from a set of possible decision alternatives. Because of uncertainty, the decision maker cannot fully predict or control the outcome of the decision chosen. The outcome is instead affected by random factors, and these random factors determine the situation resulting when the decision is executed. Each of these possible situations is referred to as a possible *state of nature* [Hillier & Lieberman 2005]. If we consider the state of nature a random variable, we can translate the information into a probability distribution, assigning each state of nature a probability. When making decisions, the decision maker takes into account the information about the relative possibility of the possible states of nature.

3.2 Maximising Payoff

The decision maker knows the resulting payoff for each combination of a decision alternative and state of nature. The payoff is used to quantitatively measure the value of the consequences of the outcome. The payoff is often represented by net monetary gain thereby creating a goal of maximising the profit.

3.3 Bayes' Decision Rule

Bayes' decision rule is one way to make a decision based on the expected costs of the different outcomes.

The probabilities for the different states of nature are computed and the expected values for all possible decision alternatives are calculated. The alternative that renders the maximum expected payoff is the one to choose.
[Hillier & Lieberman 2005].

3.4 The Introduction of Experimentation

In many cases decision making without experimentation is not good enough, since the estimates of the probabilities are too rough and unspecific. To achieve more information in order to improve and refine the the probabilities we introduce experimentation to the decision making process.

Experimentation gives the decision maker more alternatives to choose from since he, she or it can choose to either perform an experiment or not. In general, the outcome of an experimentation is not certain but instead based on probabilities, and can thus be considered a random variable. Given the outcome of the experiment we can determine new estimates of the probabilities of the different states of nature, using Bayes' theorem (Section 2.2). Thus experimentation is used to provide more information to the decision maker.

3.5 Decision Making Applied to this Thesis

In the context of this thesis, the decision maker is the prototype designed, and each state of nature is a state where exactly one component is out of order. For instance, for a certain broken truck two states of nature could be *Fuel tank faulty* and *High pressure pump faulty*, and two decision alternatives could be *Repair fuel tank* and *Repair high pressure pump*. Now assume that the repair costs for the fuel tank and the high pressure pump are equal and the probability that the fuel tank is broken is twice the probability of the high pressure pump. Then the decision maker would

prefer to repair the fuel tank first, because that alternative would be the most profitable.

In the example above, we might want to perform a test before we repair any of the components. Given the state of nature, this test has the ability to point out the high pressure pump as the faulty component with some probability. By the nature of probabilistic reasoning there is always a risk that the test will not provide us with the information that we want since most tests are non-perfect. The issues of non-perfect tests and the uncertainty of deciding which component to repair can be handled by an application of decision theory.

Chapter 4

The Troubleshooting Problem

Finding a faulty component in a non-working device can be both difficult and time-consuming. Such time-consuming tasks are expensive and it is desirable to find a sequence of actions that will repair the device in an efficient way. Finding this plan of action, which includes replacing faulty components as well as making observations and tests, is the process that is called *troubleshooting (TS)* [Heckerman et al. 1995].

4.1 Decision Theoretic Troubleshooting

In some cases, the cost of the troubleshooting process might be significant. For this reason, it is preferable to consider the costs of repair when looking for the preferred order for the troubleshooting steps. This type of troubleshooting, balancing costs and probabilities to find the best next repairing step, is called *decision theoretic troubleshooting* [Heckerman et al. 1995]. As the name decision theoretic implies, the troubleshooting theory is based on decision theory and the basics of probability.

Langseth and Jensen [Langseth & Jensen 2003] present the following definition of a troubleshooting system:

A device consists of K components $X = X_1, \ldots, X_K$ that can be either non-faulty (NF) or faulty (F). The status of the components is unknown as the troubleshooting is initialised, hence X is considered a set of random variables. As the search for the faulty component begins, the goal is to determine the states of these random variables, following a predefined order of *actions*, discussed further below. This order is called the troubleshooting *strategy*, S.

4.2 Actions

When determining the troubleshooting strategy one can choose from N_A different actions, each being either an *observation*, a *repair* or a *test*. An *observation* is

an action through which we can determine with certainty whether a component is faulty or not.

A *repair* is either a replacement or a repair of a possibly broken component. A repair will guarantee that the component is non-faulty when the repair is made. If the component is the faulty one, the repair action will repair the device.

A *test* will give us additional information about the system when it is performed. Tests in general do not indicate correctly in all cases. Only *perfect tests* will always have correct outcomes. The difference between an observation and a test is therefore that an observation will with certainty point out the component as being faulty or non-faulty, while a test will do this with some probability.

Since we do not know the state of an action before it is performed, the actions are considered random variables. These actions are modelled by the set $\mathcal{A} = A_1, \ldots, A_{N_A}$. For example, let us consider the action A_i: observe X_i. Before the observation is made we do not know the state of X_i. Thus A_i can take on two different values; either X_i is faulty or X_i is non-faulty. As the action is performed, the state of the random variable is determined. Then we know the value of A_i, namely $A_i = a_i$. This adds information to the current state of knowledge.

Chapter 5

Theory Of Expected Cost Of Repair

Chapter 4 presents the troubleshooting process and the idea of ordering components that might be faulty into a strategy. There are many different ways to order the components, and in this section we present the sorting criterion used in the troubleshooter prototype. We will present the theory behind a way of minimising troubleshooting costs, namely the theory of *Expected Cost of Repair*. We begin by presenting some assumptions under which the theory is applied.

5.1 General Assumptions

In this thesis the formulation of the troubleshooting problem follows the general assumptions presented below. These assumptions, apart from number 4, are stated by Heckerman [Heckerman et al. 1995].

1. At the start of troubleshooting, the device is faulty.

2. There exists one single fault, meaning that exactly one component is abnormal and responsible for the failure of the device. All the other components are functioning properly.

3. Each component is either observable or unobservable. An observable component can be unambiguously tested or inspected to determine if it is functioning properly. If it is found to be faulty, it is repaired immediately. Unobservable components can not be directly observed, but are instantly repaired.

4. There are no false alarms for the trouble codes, alarms, from the diagnosis system.

5. Immediately following any component repair, the function of the system is controlled with cost C^s.

6. The costs of observation and repair of any component do not depend on previous repair or observation actions.

It is worthwhile to comment on these assumptions. Assumption 1 is quite natural; if the device is functioning properly and there is no sign of a fault, we do not start a troubleshooting process. Troubleshooting the device becomes important only when a fault is present.

Assumption 2 states that there is exactly one single fault. This assumption is very important throughout this thesis, because it simplifies much of the computations. If we find a faulty component and repair it, we can by assumption 2 be sure that the device is working properly again.

The single fault assumption also implies that all cases with more than one faulty component can be interpreted as impossible. Assuming that there is only one fault clearly is a simplification, but Heckerman argues that it is a reasonable assumption. Heckerman comments that it is possible but not very likely that two components break at fairly the same time [Heckerman et al. 1995].

Assumption 3 handles the property of observability. Depending on the parameters in the model, an observable component may be viewed as unobservable. For example, if a component is observable, but at a very high cost, it might be more cost-efficient to regard it as being unobservable. This can be done without any difficulties.

Assumption 4 concerns the trouble codes generated by the diagnosis system. By this assumption we state that when an alarm is generated, there is an underlying fault in a component. An alarm cannot be generated if the device is functioning properly.

Assumptions 5 and 6 concern the cost of troubleshooting. To use the Expected Cost of Repair as it is described in Section 5.2, these assumptions are essential. Used in an application to a diesel engine, these assumptions are somewhat hard to fulfil and hence we will also discuss a way to relax the two assumptions in Chapter 7.

5.2 Expected Cost of Repair

Let, as before, the system consist of K different components, $c_1, c_2, \ldots, c_K$. Define the probability of component c_i being faulty given our current state of knowledge as $p_i = P(c_i = \text{faulty}|\epsilon)$. At this stage we only allow observations and repairs, and tests are not included in the model. Let $C^o_{c_i}$ and $C^r_{c_i}$ denote the costs of observing and repairing component c_i and let C^s denote the cost of observing whether the system is functioning normally or not.

Let the sequence $S = (x_1, x_2, \ldots, x_K)$ denote a specific ordering of the components $c_1, c_2, \ldots, c_K$ in which we want to make observations and possibly repairs. Then we

can define the *Expected Cost of Repair* for the sequence S, as

$$\begin{aligned} ECR(S) &= ECR(x_1, x_2, \ldots, x_K) = \\ &= \left(C^o_{x_1} + p_{x_1}\left(C^r_{x_1} + C^s\right)\right) + \left(1 - p_{x_1}\right)\left(C^o_{x_2} + \frac{p_{x_2}}{1 - p_{x_1}}\left(C^r_{x_2} + C^s\right)\right) + \\ &\quad + \left(1 - p_{x_1} - p_{x_2}\right)\left(C^o_{x_3} + \frac{p_{x_3}}{1 - p_{x_1} - p_{x_2}}\left(C^r_{x_3} + C^s\right)\right) + \ldots = \\ &= \sum_{i=1}^{k}\left[\left(1 - \sum_{j=1}^{i-1} p_{x_j}\right) C^o_{x_i} + p_{x_i}\left(C^r_{x_i} + C^s\right)\right] \end{aligned} \quad (5.1)$$

The expression above describes the expected cost incurred during a troubleshooting and repair process, and is easiest understood through the following example. Suppose we have a malfunctioning system with three components c_1, c_2, c_3, and we order them in $S = (c_2, c_3, c_1)$. This means that $x_1 = c_2$, $x_2 = c_3$ and $x_3 = c_1$. We first observe c_2 at the cost $C^o_{c_2}$, and with probability p_2, c_2 is found to be faulty. If c_2 is faulty, we repair it and thereby the whole system, resulting in an additional cost of $C^r_{c_2} + C^s$. With probability $1 - p_2$, c_2 is functioning properly, and we then move on to observe the next component in S, c_3. With probability $\frac{p_3}{1-p_2}$ we find that c_3 is faulty and we repair it, and in this way we continue the repair process until the component causing the faulty behaviour is found and repaired.

5.3 The Optimal Sequence

If the objective is to minimise the expected cost of repair in a long term perspective, there exists an optimal way to order (and repair) the components in the system. This is shown by Heckerman [Heckerman et al. 1995] and the proof is presented here. Heckerman uses the expression for $ECR(S)$ and considers a case where the order of components x_k and x_{k+1} is transposed, creating a new sequence called S_N. Except for terms $i = k$ and $i = k + 1$, all terms in $ECR(S)$ and $ECR(S_N)$ are the same. When subtracting $ECR(S_N)$ from $ECR(S)$, these terms will cancel out, which results in:

$$\begin{aligned} ECR(S) - ECR(S_N) &= \\ &= ECR(x_1, x_2, \ldots, x_K) - ECR(x_1, \ldots, x_{k+1}, x_k, \ldots x_K) = \\ &= p_{x_k+1} C^o_{x_k} - p_{x_k} C^o_{x_{k+1}} \end{aligned} \quad (5.2)$$

From this expression it is clear that $ECR(S)$ is lower than $ECR(S_N)$ if and only if

$$p_{x_k} C^o_{x_{k+1}} \quad > \quad p_{x_k+1} C^o_{x_k} \tag{5.3}$$

$$\Longleftrightarrow$$

$$\frac{p_{x_k}}{C^o_{x_k}} \quad > \quad \frac{p_{x_k+1}}{C^o_{x_{k+1}}} \tag{5.4}$$

Heckerman calls the ratio $p_{x_k}/C^o_{x_k}$ the *efficiency* of the kth component in a sequence [Heckerman et al. 1995]. In this thesis, the efficiency of the kth component is denoted η_k. If we construct the sequence S by ordering the components with respect to descending efficiency, we are guaranteed to have the optimal sequence, as long as the sequence only consists of observations and repairs. If tests are included, this is no longer guaranteed to be true [Heckerman et al. 1995].

5.4 Expected Cost of Repair Including Tests

To the model consisting of K components, we add N_T tests, $T_1, T_2, \cdots, T_{N_T}$. The addition of tests changes the properties of the troubleshooting process radically. Most importantly, by the addition of tests as possible troubleshooting actions we can no longer be sure that an optimal repair strategy will be obtained even though the efficiency sorting criterion is used.

To all tests we assign probabilities and test costs. For each test T_i we denote the specific cost corresponding to performing the test C^T_i. We need the probability that the test does indicate (I) a fault given the current state of information, $P(T_i = I|\epsilon)$. We also need the probability that the test does not indicate (NI) given the same state of information, $P(T_i = NI|\epsilon)$.

Consider the small system consisting of three components c_1, c_2, c_3. In Section 5.2 a sorted repair strategy $S = (c_2, c_3, c_1)$ was chosen. Assume that we now have the possibility of performing the test T before deciding on the strategy. This might change the preferred order since the test outcome gives us additional information about the system.

Suppose that we perform the test and the outcome is an indication of a fault in component c_3 or c_1. The probability distribution has changed and from this change we conclude that the best repair strategy is $S_{T_I} = (c_3, c_1, c_2)$. If we instead suppose that the test does not indicate, the probabilities will change in another way. The best strategy might then become $S_{T_{NI}} = (c_2, c_1, c_3)$. As different strategies, S_{T_I} and $S_{T_{NI}}$ also correspond to different values of ECR.

Using this information and the test probabilities we calculate the *Expected Cost of Repair Including Test (ECRT(T))* as:

$$ECRT(T, \epsilon) = C^T + P(T = I|\epsilon) \cdot ECR(S_{T_I}) + P(T = NI|\epsilon) \cdot ECR(S_{T_{NI}}) \quad (5.5)$$

5.5 The Value of Information

Optimal troubleshooting is the task of combining tests, repairs and observations in a way that minimises ECR. To be able to compare tests to other actions, we need a method for evaluating the information the test adds.

Let us assume that we are troubleshooting a system and are to decide whether to perform a test or to repair a component. It is reasonable that we choose the action that, at this moment, will give the minimal expected cost. If we exclude the test from the possible actions, the remaining actions could be ordered into a strategy S. From this we obtain the ECR(S). If the test is considered we can compute the $ECRT(T)$ as the expected cost after performing the test, following Section 5.4.

To determine the most cost-efficient action, we introduce *the Value of Information (VOI)* [Hillier & Lieberman 2005] of test T as:

$$VOI(T) = ECR(S) - ECRT(T) \quad (5.6)$$

If the difference between ECR and ECRT is positive, it can be considered to be the gain we make by performing the test. If the value of VOI(T) is negative, there is no gain of performing the test. Instead, performing the test will only give rise to a higher repair cost.

Chapter 6

Algorithms for Troubleshooting

In [Gustavsson 2006] four different algorithms for cost-efficient troubleshooting are studied. Considering the algorithm characteristics, like complexity and performance, discussed by Gustavsson, two algorithms are chosen for the continued study of the ECR performance. As a development of this previous study, the computations of the ECR are different (c.f. Chapter 7) due to an adjustment to the specific process of troubleshooting a truck. The algorithms are presented in the following sections.

6.1 The Greedy Algorithm

The greedy approach is a naive way to troubleshoot. The components can only be sorted by a certain criterion, decided upon in advance, and no tests can be integrated into the strategy. As the name indicates, the Greedy algorithm aims to choose each step to maximise the utility. Which, from the ECR point of view, means an ordering of components by descending efficiency. The Greedy algorithm can be described by three steps which also are illustrated in the flowchart in figure 6.1.

1. Calculate the probabilities $P(c_i = F|\epsilon)$ for the components being faulty.
2. Calculate the efficiency, η_i, for each component.
3. Sort all components by descending efficiency to obtain the Greedy strategy.

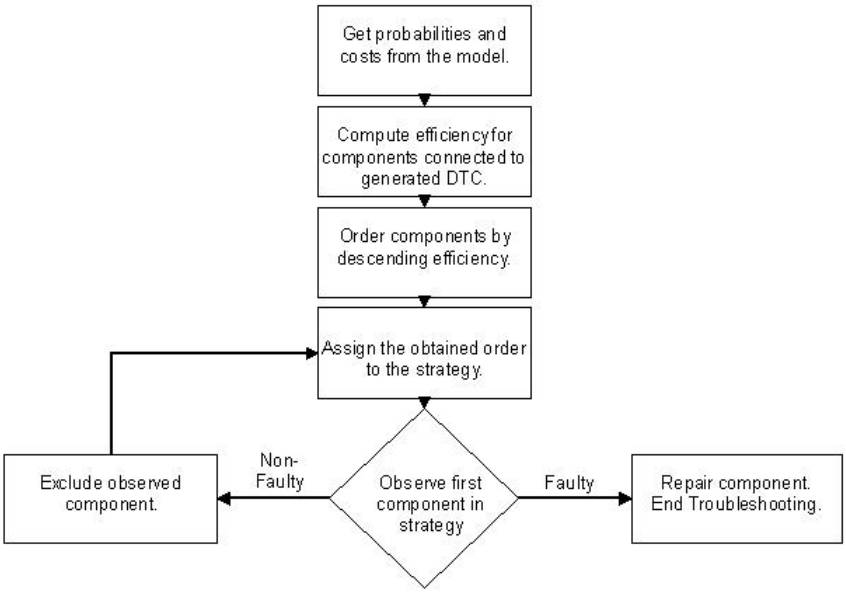

Figure 6.1. Flowchart of the Greedy algorithm

6.2 The One Step Look Ahead Algorithm

The One Step Look Ahead algorithm (One Step) uses the theory of ECR and ECRT to include the possibility of adding information gathering steps to the troubleshooting process. To handle the possibility of tests as future actions One Step uses the myopic approach, described in the following section.

6.2.1 A Myopic Approach

Troubleshooting a system where it is possible to perform one or several tests is a difficult problem. When tests are introduced a predefined strategy cannot be uniquely determined because the outcome of the test is not certain until the test is performed. The test outcome affects the probability distribution for the components to be faulty, and thus it might change the preferred order of the components in the strategy.

One way of handling this problem is to use a *myopic approach* when troubleshooting. The name *myopic* refers to the algorithm's ability to look one step ahead and evaluate the benefit from performing different actions, before an action is chosen. When this evaluation is made, only observations and repair actions are allowed, which is a simplyfying assumption.

Closely following Jensen in [Jensen 2001], the myopic approach is described as follows. Assume that we are at any stage of troubleshooting. At this stage we allow tests to be performed, but in the future we allow only repair and observation actions. In this case, the task reduces to calculating expected costs given the various outcomes of the possible tests. This conveniently enables us to use the theory of Expected Cost of Repair, even in the case where tests are allowed [Jensen 2001].

6.2.2 General Description of the One Step Algorithm

In each step of One Step, two different calculations are performed. Firstly, the greedy approach is used to calculate an order of component repair that minimises ECR. Secondly, ECRT is calculated for all related tests. One Step always chooses one of the actions; perform a test, make an observation or make a repair that, in this step, generates the lowest expected cost. A graphic representation of One Step is shown in Figure 6.2 and can be described by the following steps:

1. Decide the strategy for the components and calculate its ECR.
   The repair strategy is found by using the Greedy algorithm.

2. Calculate the ECRT for all tests related to possibly faulty components.

3. Compare the ECR to all ECRTs.
   Choose the strategy that generates the lowest expected cost.

4. - If ECR is the best choice: Repair the component that comes first in the strategy.
   - If the best choice is one of the ECRTs: Perform the corresponding test.

5. Return to 1 if the device does not work properly.

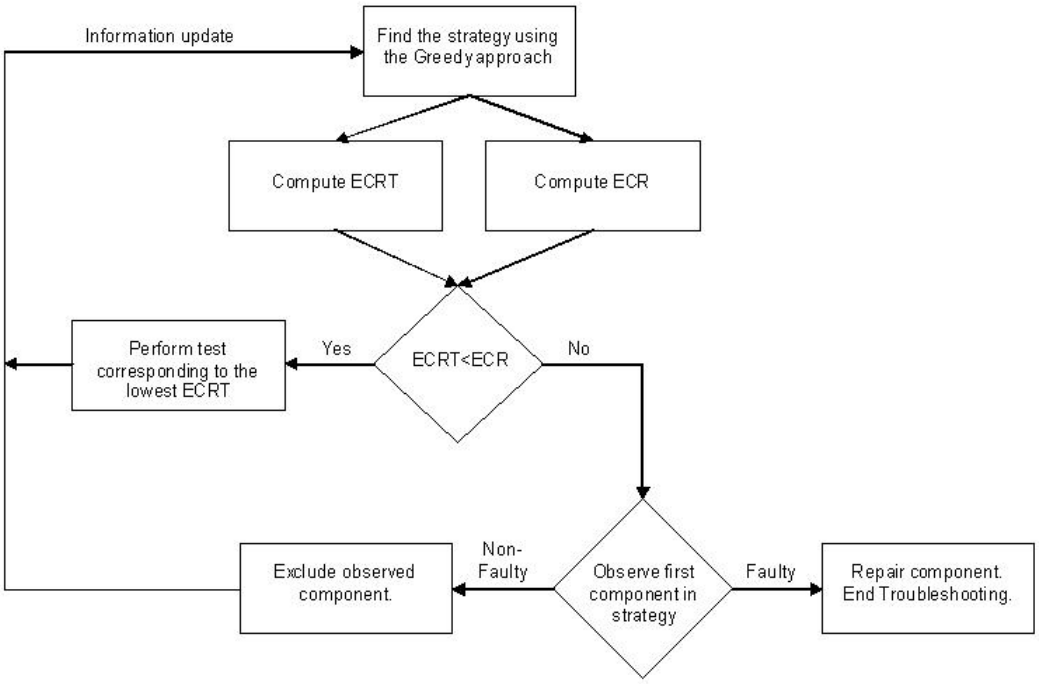

Figure 6.2. Flowchart of the One Step Look Ahead algorithm

Chapter 7

Costs

Heckerman assumes that the cost of repairing a component is independent of previous actions [Heckerman et al. 1995]. Since the costs, by that assumption, do not depend on the previous actions we will refer to them as *independent costs*.

This section presents one possible way of relaxing the assumption of independent costs developed during this thesis work. The relaxation is made to adapt the theory to the process of troubleshooting trucks. A process where the cost of repair depends as much on how much we have to dismantle the truck to reach a specific component as on the spare part cost for that component.

7.1 Independent and Dependent Costs

The assumption that the cost of an action is independent of the previous actions leads to a case with constant costs. When assigning the cost of an action, we do not consider the current state of the device; for example if any parts are disassembled or not. When using independent costs we assume that when an action is to be performed, no changes have been done to the device beforehand.

Assuming independent costs clearly is a way to simplify the problem and in order to adjust the troubleshooter to a realistic case, this simplification is abandoned. The cost of repair will no longer be considered as constant but will be a function of the current state of the device. In this thesis we introduce the *level of disassembly* to describe these states.

7.2 Level of Disassembly

For a device, we identify different *levels of disassembly*, corresponding to different amounts of dismantlement. We denote the ith level of disassembly λ_i. All levels of

disassembly belong to the set Λ such that

$$\Lambda = \{\lambda_1, \lambda_2, \lambda_3, \ldots\} \tag{7.1}$$

Figure 7.1 shows how a general device can be divided into different levels of disassembly. The graph consists of levels of disassembly and groups of components. A group can contain one or many components. Every level must contain at least one group. Groups on the same level are named $a, b, c, \ldots$.

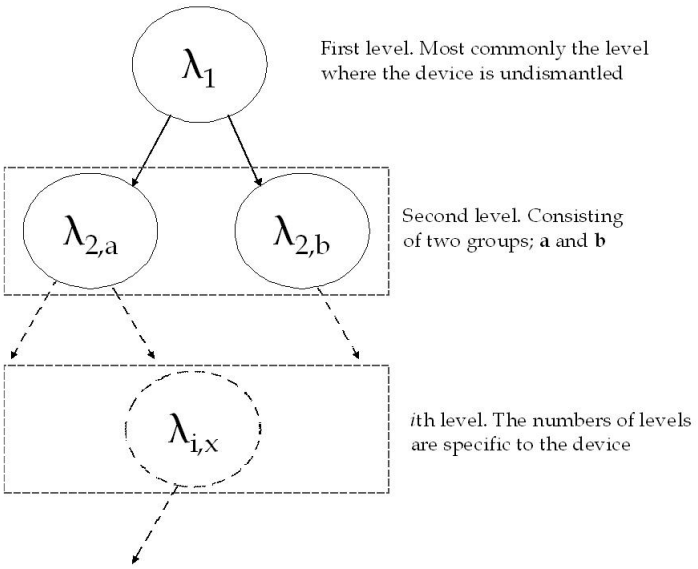

Figure 7.1. Levels of disassembly in a general case.

Typical levels of disassembly are "complete device", which corresponds to the case when no components have been removed from the device, and "free component level: component X" which corresponds to the situation when the device is dismantled until component X is free. We can define as many levels of disassembly as we want. With many levels, the costs can be more specifically defined, but this will also increase the complexity of the model.

It is reasonable to assume that different components can be observed and repaired at different levels of disassembly. To each component we assign the level of disassembly at which the component can be repaired. If the component is observable we must also specify the level of disassembly at which we can observe the component. For these levels the repair and observation costs are specified. For each test we must specify the level of disassembly at which the test can be performed, and the cost of performing the test.

7.3 Costs of Transition

The dismounting and mounting of parts on the device corresponds to transitions between different levels of disassembly. To these transitions there are corresponding costs, called *costs of transition*. The cost of transition between level λ_i and λ_j is denoted $C_{\lambda_i \to \lambda_j}$. These costs of transition are added to each cost of action to obtain a cost of repair dependent on the different levels of disassembly.

There are some rules regarding the calculations of the transition costs, which we will explain below.

- A *simple transition* is a transition from one level to an adjacent level.
- Transitions between different groups on the same level must go via an adjacent level, connected to both groups.
- A transition between level λ_i and λ_j is made via a *transition path*. A transition path from λ_i to λ_j is the shortest possible sequence of simple transitions from λ_i to λ_j.
- The cost of a transition path is the sum of the cost of all simple transitions in the transition path.
- All paths are reversible, meaning that the cost of transition from λ_i to λ_j equals the cost of transition from λ_j to λ_i.

7.4 Functional Control

As it is necessary to define a level of disassembly where all repairs, observations and tests can be performed, it is also necessary to define the level where it is possible to perform the functional control. For a complex system it is possible that these functional controls can be performed at different levels depending on the previous action performed.

The cost for the functional control, C^s, will not only be the cost for performing the actual control, it will also depend on which level of disassembly the previous repair was made i.e. the transition cost from the level of last repair to the level where the functional control can be performed.

7.5 Determining a Strategy using Dependent Costs

The changes in costs due to transitions between different levels of disassembly cause changes in the determination of the troubleshooting strategy. The efficiency for each component is no longer constant but needs to be determined by considering

the previous action performed. This means that after each chosen action, the efficency needs to be calculated to determine the next component s_i in the strategy. An attempt to formulate this problem in mathematical terms is presented below.

The troubleshooting case consists of a specific set of possibly faulty components, $\mathcal{F}$, (c.f. Section 2.4) which we want to order into a strategy $S = (s_1, ..., s_n)$. We want to use the efficiency criterion to order the components and therefore it is of great importance to obtain the correct observation cost at each level of disassembly. We introduce the variable l_i as the level of disassembly at step i, which can be either of the levels of disassembly, λ_j. This variable l_i is linked to the ith component in the repair strategy and we consider all costs as a function of this variable.

The troubleshooting starts on level l_0. The first component in the strategy, s_1, is determined by considering the component with the highest efficiency based on this level, usually the complete device level.

$$\max\left(\frac{p_i}{C_i^o(l_0)}\right) \text{ for } i \text{ such that} \quad c_i \in \mathcal{F} \tag{7.2}$$

When the first component, s_1, has been determined, the level of assembly is changed from $l_0 \rightarrow l_1$. The level l_1 is where we can observe the component corresponding to s_1. The efficiency of the remaining components is then calculated using the cost of observation for the current level, l_1.

$$\max\left(\frac{p_j}{C_j^o(l_1)}\right) \text{ for } j \text{ such that} \quad c_j \in \mathcal{F}\backslash c_i \tag{7.3}$$

Each time a component is ordered into the strategy it is removed from the set $\mathcal{F}$. In each step, the efficiency is consequently calculated for components not yet ordered into the strategy. This follows Equation 7.3 where more and more components are excluded from the set $\mathcal{F}$.

7.6 The Expected Cost of Repair using Dependent Costs

As the independent cost approach is abandoned we obtain a more flexible representation of the repair procedure. This approach means that all costs are varying with the different levels of disassembly. Thus, the calculation of the ECR has to be changed.

The variable l is significant in the computation of the ECR since all costs are said to be a function of this variable. The probabilities for the components to be faulty are unaffected by l and are left unchanged:

For a strategy $S = (s_1, s_2, ...,)$ the ECR for dependent costs is:

$$ECR(S,l) = \left(C^o_{s_1}(l_0) + p_{s_1}\left(C^r_{s_1}(l^o_1) + C^s(l^r_1)\right)\right) + \\ + (1-p_{s_1})\left(C^o_{s_2}(l_1) + \frac{p_{s_2}}{1-p_{s_1}}\left(C^r_{s_2}(l^o_2) + C^s(l^r_2)\right)\right) + \cdots \tag{7.4}$$

When we study the expression above it is important to consider the dependencies on l. This is done in the example below.

A small device consists of two different components, c_1 and c_2. The components are faulty with the probabilities p_1 and p_2. For this device there are three different levels of disassembly, $\lambda_a, \lambda_b, \lambda_c$. The levels for performing repairs and observations are presented in Table 7.1. The functional control can only be performed at λ_a.

Action	Level of Disassembly
Observe c_1	λ_a
Observe c_2	λ_b
Repair c_1	λ_b
Repair c_2	λ_c

Table 7.1. Level of disassembly for the components in Example 2

The strategy for this troubleshooting procedure is $S' = (c_2, c_1)$ and from this we can calculate the $ECR(S', l)$.

$$ECR(S',l) = \left(C^o_{c_2}(\lambda_a) + p_1\left(C^r_{c_2}(\lambda_b) + C^s(\lambda_c)\right)\right) + \\ + (1-p_1)\left(C^o_{c_1}(\lambda_b) + \frac{p_2}{1-p_1}\left(C^r_{c_1}(\lambda_a) + C^s(\lambda_a)\right)\right) \tag{7.5}$$

We start the troubleshooting at λ_a, the level corresponding to the complete device. The cost of observing c_2 is obtained from the expression $C^o_{c_2}(\lambda_a)$.

- $C^o_{c_2}(\lambda_a)$
  In this cost we include the cost of observation of c_2 and the transition cost from level λ_a to the level where c_2 can be observed, λ_b . For this we use the following notation: $C^o_{c_2}(\lambda_a) = C^o_{c_2} + C_{\lambda_a \to \lambda_b}$.

With probability p_2 we find the component c_2 to be faulty and want to repair it. We denote the cost of repair as $C^r_{c_2}(\lambda_b)$.

- $C^r_{c_2}(\lambda_b)$
  We include the repair cost for c_2 as well as the transition cost from the current level λ_b to the level where c_2 can be repaired, λ_c. $C^r_{c_2}(\lambda_b) = C^r_{c_2} + C_{\lambda_b \to \lambda_c}$.

After a repair is made we perform a functional control corresponding to the cost $C^S_{c_2}(\lambda_c)$.

- $C^S_{c_2}(\lambda_c)$
  The cost of the functional control adds the cost of the functional control, C^S, and the transition cost from $\lambda_c \rightarrow \lambda_a$ where the functional control is performed. $C^S_{c_2}(\lambda_c) = C^S_{c_2} + C_{\lambda_c \rightarrow \lambda_a}$

With the probability $(1 - p_2)$ is c_2 non-faulty and we get the remaining terms for c_1. To determine the cost of observation of c_1, $C^o_{c_1}(\lambda)$, we need to consider whether c_2 is observable or not.

- c_2 is observable

If c_2 is observable, the level we are to continue from is considered to be the level of observation for c_2. We get the observation cost for c_1 as $C^o_{c_1}(\lambda_b) = C^o_{c_1} + C_{\lambda_b \rightarrow \lambda_a}$.

- c_2 is unobservable

If c_2 is unobservable we know that a functional control has been performed before continuing to c_1 and therefore we get the observation cost for c_1 as $C^o_{c_1}(\lambda_a) = C^o_{c_1} + C_{\lambda_a \rightarrow \lambda_a}$. In this case the cost of transition will not correspond to a cost since it is the same level and no groups of components exists on this level.

Chapter 8

The Extreme High Pressure Injection System

In this section the reader is provided with a short description of the XPI (Extreme High Pressure Injection) system, the diesel engine and the on-board diagnosis system.

8.1 The Diesel Engine

This description of the diesel engine is based on an article in [Wikipedia 2007]. The diesel engine is an internal combustion engine that uses compression ignition. There are two classes of diesel engines, one with a two-stroke cycle and one, more commonly used, using a four-stroke cycle. The four-stroke engine is the type used in Scania trucks.

Internal combustion engines require a combustion chamber, where pressure is built up. This chamber is the cylinder, in which the piston moves. Attached to the cylinder head are two valves for the inlet of air and two valves for the outlet of exhaust gases. An injector valve, which injects the fuel into the cylinder, is also attached to each cylinder head. The parts are shown in the schematic picture of a cylinder in Figure 8.1. Most truck engines have five, six or eight cylinders.

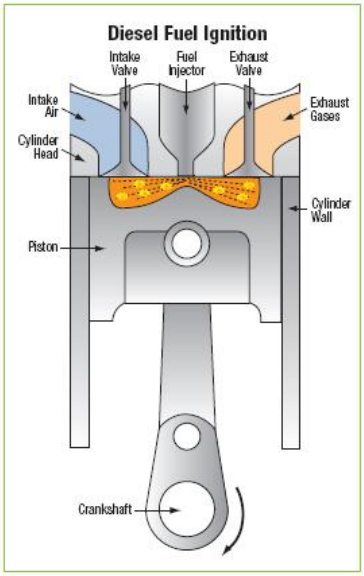

Figure 8.1. A schematic picture of a cylinder and the important parts.

Each cycle consists of four phases, called *strokes*, which are shown in Figure 8.2.

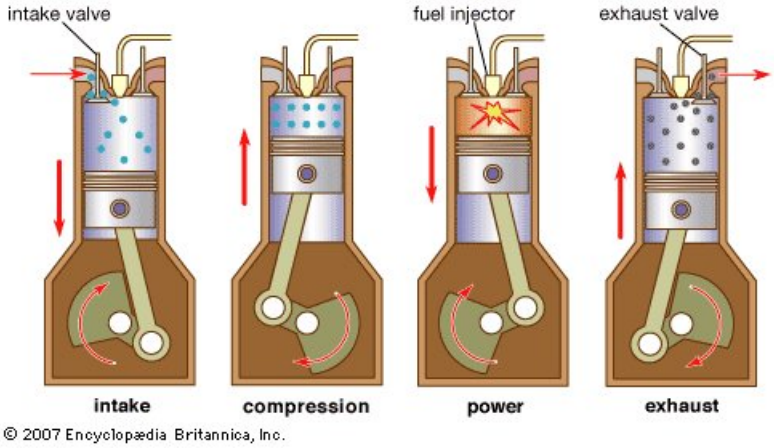

Figure 8.2. A schematic picture of the four strokes of a diesel engine.

The first stroke is called the *intake stroke*. During this phase the piston moves down, reducing the pressure in the cylinder. Thereby air is drawn into the cylinder through the inlet valves. The intake valves then close and the *compression stroke* starts.

In this phase the air is compressed towards the top of the cylinder by the piston, which moves upwards. The compression leads to a higher pressure and the temper-

ature rises. Just as the pressure reaches the point at which the fuel spontaneously combusts, fuel of very high pressure is injected. The mixture of fuel and air then ignites. The combustion leads to an expansion forcing the piston back down, and the *power stroke* begins.

During the power stroke, all valves are still kept closed, and the piston moves downwards. The kinetic energy caused by this movement is transferred to the crankshaft. The last phase is called the *exhaust stroke*. At this stage the outlet valves are opened and the piston moves upwards again, forcing the exhaust gases out through the outlet valves. The outlet valves are then closed, the inlet valves opened and the cycle is repeated.

8.2 Operating Principle of the XPI System

The XPI system is a fuel injection system where all fuel injectors are connected to a high pressure tube, called the *common rail*. This rail distributes the fuel at a steady high pressure to the injectors and can maintain a constant high pressure in all injectors simultaneously.

High pressure fuel injection improves the combustion in the cylinders. This results in more efficient fuel consumption and a decreasing amount of particles, i.e. soot, in the exhaust gases. A drawback of the common rail system is its sensitivity due to very small tolerances. Bad fuel quality and unwanted particles can cause leakage [Telborn 2007].

8.2.1 Fuel Transport in the XPI System

The fuel is pressed through the suction side filter, marked by 1 in Figure 8.3, before passing the Low Pressure Pump, LPP, (4) on its way to the Inlet Metering Valve, IMV (6). The fuel then passes the pressure side filter (5). The IMV is controlled by the ECU (Electric Control Unit, described in the following section) and regulates the amount of fuel transported to the High Pressure Pump, HPP (7). The fuel pressure is heavily increased in the HPP, at maximum reaching 2500 bar. The fuel is then accumulated in the rail (8). On the rail, the Rail Pressure Sensor, RPS (9), is mounted. As the name suggests, this sensor measures the pressure in the rail.

From the rail the fuel is transported to the injectors via High Pressure Lines (11) and High Pressure Connectors, HPC (12). The fuel is injected into the cylinders by the injectors (13). The fuel injections are electrically controlled by magnet valves which enable high precision regarding fuel amount and injection time. As a precaution the Metering Dump Valve, MDV (10), is connected to the rail. If the rail pressure gets above 3000 bar, this valve will open, thereby lowering the pressure to 1000 bar [Scania 2006].

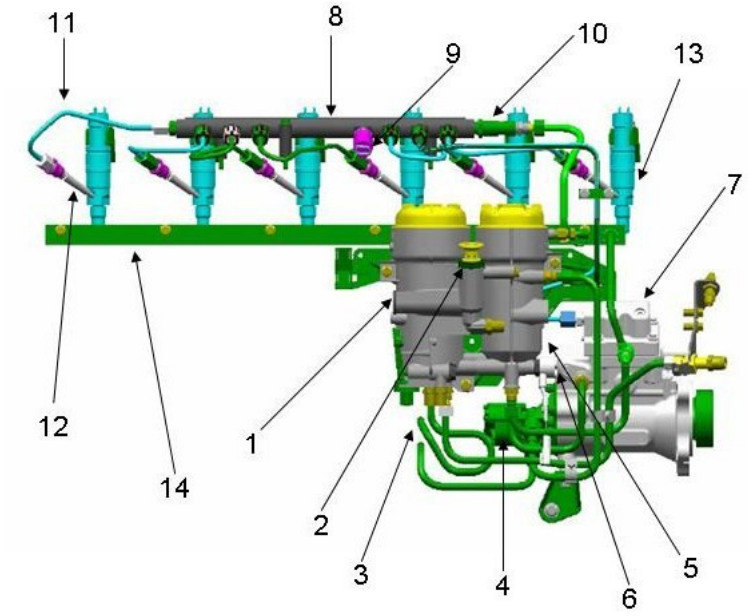

Figure 8.3. A schematic picture of the XPI system. The picture can be found in [Scania 2006]

8.2.2 On-Board Diagnosis and Diagnostic Trouble Codes

In modern trucks, most operating systems, such as the engine and the brake system, are controlled by *Electric Control Units* (ECU). In these systems, there are sensors that measure temperature, fuel amount, RPM (Revolutions per Minute) etc. These signals are registered by the ECU. In the ECU there is also a diagnosis system. The principle of the diagnosis system is shown in Figure 8.4.

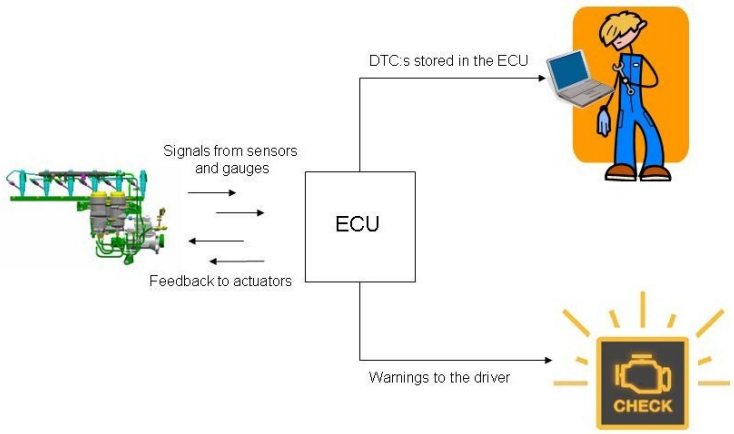

Figure 8.4. A sketch of the on-board diagnosis.

Continuously during operation, diagnostic tests are performed by the ECU. These tests compare measured signals to each other or with reference values. If an abnormality is detected a *Diagnostic Trouble Code (DTC)* is generated and stored in the ECU.

Some DTCs may attract the driver's attention by lighting an indicator on the dashboard, but that is not always the case. Others are used only as information to the workshop mechanic. It is possible that no DTC is generated, even though there is some fault present. In this thesis we assume that the opposite is impossible, i.e. a DTC cannot be generated unless some component is faulty.

At the workshop, a computer is connected to the ECU and the generated DTCs are retrieved. Thereafter, the troubleshooting process can begin.

Chapter 9

A Model of the XPI System

Real world systems are often very complex, which makes them hard to survey. To be able to analyse the behaviour and performance of such systems we use models. When creating a model, the main goal is to reduce the complexity without losing too much of the characteristics of the real system. In [Mossberg 2007] a Bayesian model of the XPI system is designed. In this section we will discuss some of the issues concerning the creation of the Bayesian model of the XPI system.

One of the difficulties in designing the model is to determine which parts to include and which to exclude. In general, the number of DTCs and components must be strictly limited to avoid storage and computational problems. Including too many nodes in the Bayesian network makes it too complex and does not simplify the real world system enough. On the other hand, having very few nodes will make the model too simplified and have it differing too much from the real system.

9.1 Components

One must take different aspects into account when determining which components to use in the model. Important issues can be formulated as the following questions.

- What is the failure rate of the component?

Since the model is built to be a part of a troubleshooter, we are interested in the problematic parts of the system, i.e. components with high failure rates. Including components that break very infrequently will only increase the model complexity without capturing any significant behaviour.

- How vital is the component in terms of system performance?

We want to capture the characteristic behaviour of the real world system as efficiently as possible. Therefore we concentrate on the components that are the most important parts of the system.

- Is the component available as a spare part?

In the XPI system, most of the components are very complex and constructed with extremely small tolerances. Therefore, it is not possible to repair such parts at the workshop. If such a component is found to be faulty it must be replaced with a new one, since the workshops do not have the equipment to repair it. An example of this is the injectors where the tolerance at the injection needle is only a few μm. Small particles in the diesel might cause leakage in the injector which will lead to decreasing system pressure and a "Low pressure DTC" will be generated. The extremely small tolerance in the injector is not possible to maintain in an ordinary workshop. With this in mind, only components available as spare parts are included in the model.

In total, 17 components are identified as important and included in the model. A more detailed description of how this is done can be found in [Mossberg 2007].

9.2 Observability

For some components it is possible to observe the component to determine whether it is faulty or not. Such components are called *observable*. An observable component will never be repaired unless it is first observed to be faulty. If it is found to be faulty, it must be repaired immediately.

Unobservable components, on the other hand, cannot be observed. When we choose to repair an unobservable component we therefore accept the risk that we might "repair" a non-faulty component.

To *repair* a component, observable or not, means to *replace* it.

When we determine the troubleshooting strategy we use the efficiency, η. We present the definition again:

$$\eta_i = \frac{p_i}{C^o_{c_i}}$$

We see that we use the cost of observation, $C^o_{c_i}$, when we determine the efficiency. Observation costs cannot be assigned to the unobservable components. However, it is straightforward to include the unobservable components.

Let us suppose that an unobservable component c_i is repaired with a cost R_i. We can then view the unobservable component as an observable component that is observed with cost R_i, always found to be faulty and repaired with cost zero. That is, we can include unobservable components, provided we set $C^o_i = R_i$ and $C^r_i = 0$ [Heckerman et al. 1995].

9.3 The Single Fault Constraint Node

One of the most important simplifications is the single fault assumption. This assumption means that exactly one component is faulty and repairing this component leads to a complete repair of the device. As previously stated, the single fault assumption is often a good approximation, because it is not likely that two components will fail at roughly the same time [Heckerman et al. 1995]. Furthermore, without this assumption, filling in the CPTs would be much more cumbersome. This assumption simplifies the problem, but it also implies strong dependencies between the components [Mossberg 2007].

In the Bayesian model this assumption is met by the *SFC* node, short for *Single Fault Constraint* node. This node does not represent a physical component. Instead, its only purpose is to guarantee that only one component is faulty at a time.

For a model with K components, the SFC node has $K+1$ states. The first K states handle the cases where one of the K components is faulty in a way that the ith state corresponds to the ith component being faulty. The $(K+1)$th state handles the case where none of the components are faulty. In the CPT for this node we specify the prior probability for each component to be faulty.

9.4 Diagnostic Trouble Codes

There are different kinds of DTCs (Diagnostic Trouble Codes). Some DTCs are very specific and can point out a faulty component with very high precision. These DTCs usually handle electrical faults. There are also DTCs that handle mechanical faults, and these are often not as precise. They do not point out a specific component, but rather a troublesome symptom. In the XPI system, with pressure being the most important issue, a typical symptomatic DTC is the 'Low pressure' DTC.

In this thesis, the symptomatic DTCs are of greatest interest. Because of their inability to specifically point out one component as most probable, they are difficult to use on their own, without any other information, when troubleshooting. This makes them suitable for this troubleshooter application. The troubleshooter puts the DTCs in a context where other information is available, and thereby enables more efficient troubleshooting.

For the XPI system, nine symptomatic DTCs are identified. Most of these concern pressure and leakage problems. The DTCs are listed and presented in Appendix C.

9.5 The Bayesian Model of the XPI System

The resulting model consists of 40 nodes. The nine DTCs, seventeen components and thirteen tests sum to 39 nodes. Finally, the 40th node is the Single Fault Constraint Node, described in detail in Section 9.3. Between these nodes a huge number of edges are drawn, connecting the nodes in such a way that the physical dependencies are captured [Mossberg 2007]. The model is shown in Figure 9.1.

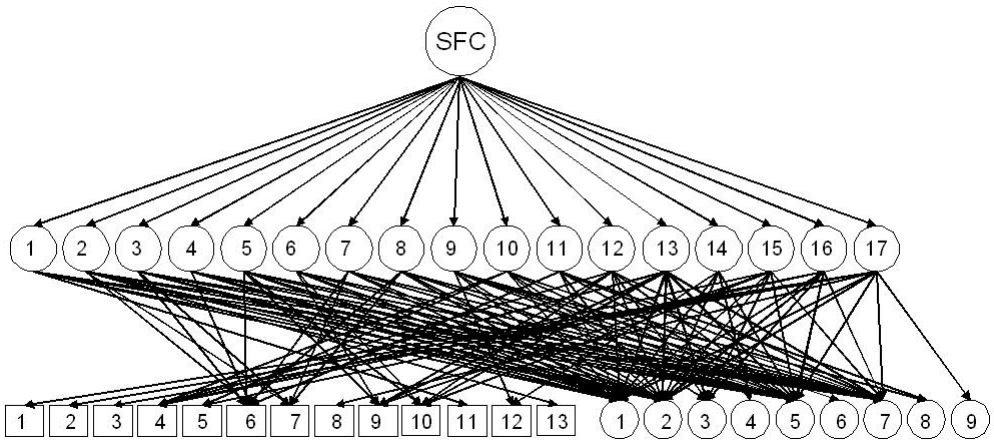

Figure 9.1. The XPI-model as a Bayesian network. The SFC (Single Fault Constraint) node is connected to all 17 components. Each of the 17 components is then connected to a specific set of tests and DTCs. The whole model has 13 tests, here represented as the squares to the left. The 9 circular nodes to the right models the DTCs (Diagnostic Trouble Codes) connected to the XPI system.

9.6 Determining the Levels of Disassembly

The procedures of repairing and observing components in the XPI system are different depending on the component in question. To model these differences we use the approach presented in Section 7. To do this we need to define our approach to the levels of disassembly for a truck.

9.6.1 Level of Disassembly

For a truck in a workshop three different levels of disassembly are taken into account; the *complete truck level*, the *tilted cab level* and the *free component level*. The free component level has six groups of components.

The complete truck level corresponds to a complete truck, where the cab is not tilted and no components are removed. It is assumed that a truck arriving at the

workshop is on the complete truck level. It is also assumed that when the functional control of the truck is performed, the truck is on this level. During much of the troubleshooting in a workshop, the cab is tilted. This is the level on which many of the tests are performed.

The free component level corresponds to the situation when the truck is dismantled until a specific component (or group of components) is free. It is not necessary that the component is removed from its original position in the truck, but in some cases this is needed. Some tests and observations require that the component is removed and mounted in some special test equipment.

9.6.2 Transition Costs

Figure 9.2 shows the different levels of disassembly and the costs corresponding to simple transitions.

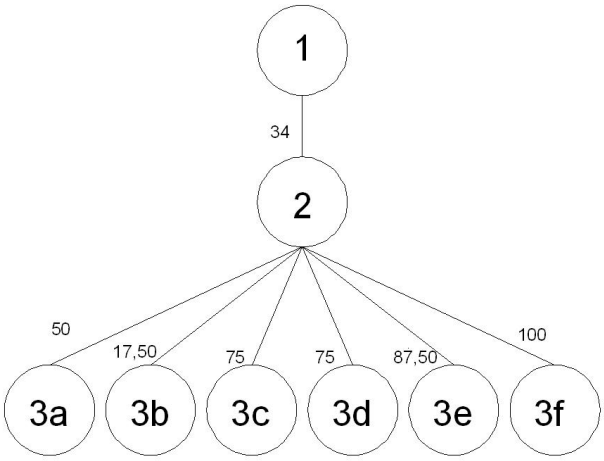

Figure 9.2. Transition costs for simple transitions between different levels of disassembly of a truck. Costs are given in SEK.

The numbers correspond to the following levels of disassembly.

λ_1 Complete truck level.

λ_2 Tilted cab level.

λ_3 Free component level:

- $\lambda_{3,a}$ The components on the suction side are free.
- $\lambda_{3,b}$ The filter housing is open.
- $\lambda_{3,c}$ The components on the pressure side are free.

$\lambda_{3,d}$ The pipe between the HPP and the rail is free.

$\lambda_{3,e}$ One injector and its HPL and HPC are free.

$\lambda_{3,f}$ The rail is free.

Using information from time measurements and experts, the costs of transitions between level λ_1 and λ_2, and between λ_2 and the groups on level λ_3 are defined. These costs are presented in Figure 9.2. From these costs, the cost of any transition can be calculated, following the rules presented in Section 7.3.

When dealing with dependent costs, we must also specify the levels of disassembly at which the components can be repaired. This data is presented in Table 9.1. There we also find the levels of disassembly at which the observable components can be observed. Unobservable components cannot be observed at any level and are marked with ' - '.

Component	Level of repair	Level of observation
Fuel tank	λ_1	λ_1
Pipes, suction side	$\lambda_{3,a}$	-
Hand pump	λ_2	-
Filter, suction side	$\lambda_{3,b}$	-
Low pressure pump, LPP	$\lambda_{3,a}$	λ_2
Pipes, pressure side	$\lambda_{3,c}$	λ_2
Main filter	$\lambda_{3,b}$	-
Pipes, filter housing	$\lambda_{3,b}$	-
Inlet metering valve, IMV	$\lambda_{3,c}$	λ_2
High pressure pump, HPP	$\lambda_{3,c}$	-
Air bleed venturi	$\lambda_{3,c}$	-
Pipes, high pressure side	$\lambda_{3,d}$	λ_2
Rail	$\lambda_{3,f}$	λ_2
Rail pressure sensor, RPS	λ_2	-
Mechanical dump valve, MDV	λ_2	λ_3
High pressure connector, HPC	$\lambda_{3,e}$	-
Injector	$\lambda_{3,e}$	-

Table 9.1. Levels of disassembly at which repairs and observations can be made.

Chapter 10

Simulating the Repair Process

To evaluate the choice of using the efficiency as the sorting parameter, the program prototype is used to find the faulty component in simulated cases. The simulations models the behaviour of the troubleshooter prototype in invented workshop cases.

10.1 Simulation Objectives

The main goal for the simulations is to conclude whether the ECR theory handles the component sorting in a cost-efficient way or not. The objective of these simulations is to rate the efficiency sorting to other sorting criteria, to see how the ECR approach to troubleshooting performs.

10.2 Component Sorting Criteria

After the sorting by efficiency is used, we change to a Greedy sorting by probability. By using this sorting we keep the probabilistic reasoning but do not add the information about the observation cost. This will affect the component ordering and it is interesting to see whether or not this makes a difference in the total repair cost. Second, we change the sorting criterion to be a completely random order, this to model a repair procedure where no information is taken into account. It is reasonable to assume that a random ordering of the components will give a total repair cost higher than a cost-efficient sorting. The random ordering can therefore be seen as somewhat of a worst-case repair scenario.

The different sorting criteria used are stated in Table 10.1.

a)	Efficiency, η	Components sorted by descending efficiency.
b)	Probability	$P(c_i \text{ faulty}\|\epsilon)$ sorted in descending order.
c)	Random order	All components randomised.

Table 10.1. The sorting criteria used for the different simulations

10.3 Carrying Out the Simulations

The simulation algorithm is constructed as a virtual mechanic that can provide the right information to the troubleshooter prototype. For one simulation cycle we obtain 100 fictious workshop cases where the same component is modelled as faulty. The modelling of component failure is described more in detail in Section 10.3.1.

The simulated repair is initialised by giving a DTC, thereby simulated as generated, as input to the program. This corresponds to the information the mechanic gets from the diagnosis system in the truck. The generated DTC becomes the current state of knowledge. The task for the troubleshooting program is to present the strategy by which the actions should be performed. The simulated repair process follows this strategy until the faulty component is encountered. The order of tests and repairs made during the search for the faulty component is obtained in a repair sequence.

To each of these repair sequences a specific cost of repair, CR, is assigned. The cost of repair is calculated using the dependent cost approach presented in Section 7. This fictious repair cost is used as a comparison tool in the evaluations of the different component sorting criteria.

10.3.1 Simulating Component Failure

For each simulation cycle one component in the XPI-model is modelled to be faulty. The simulation algorithm handles the modelling of the faulty component by a global parameter, set to be the component number of the faulty component. It is important to mention that the XPI-model and the troubleshooting prototype are unaffected by this parameter. All probability calculations and the presentation of the next action will be exactly like it would be in a real workshop troubleshooting case. The global parameter is used to be able to determine whether a certain repair action has repaired the truck or not.

Each time a repair action has been performed, the troubleshooter prototype demands a functional control. In a real workshop scenario this means that the mechanic controls if the truck is behaving normally after the specific repair. In the simulations this is handled by comparing the number of the newly repaired component to the number assigned to the global parameter. If they are identical, the troubleshooting is terminated.

In total, five components from the XPI-model are chosen to, in turn, be modelled as faulty. These components constitute simulation cycles named A to E in Table 10.2 below, which also shows the given combinations of faulty components and generated DTCs.

Cycle	Component	Generated DTC
A	MDV, c_{15}	DTC3
B	HPC, c_{16}	DTC1, DTC3
C	Injectors, c_{17}	DTC1
D	Filter, suction side, c_4	DTC1
E	IMV, c_9	DTC8

Table 10.2. Components and corresponding DTCs used in the simulations.

10.3.2 Simulating Test Outcome

When a test is performed during program execution, the user is asked to provide information about the test outcome. In a computerised simulation this feedback needs to be provided in another way.

To obtain a somewhat realistic test outcome, the marginalised test probability, $p_{T_i} = P(T_i = I|\epsilon)$, from the Bayesian network needs to be considered. The test outcome is therefore generated as follows: A random number, r, is generated from the uniform distribution on the unit interval, $U(0, 1)$. The random number, r, is compared to p_{T_i} which constitutes the limit value for the outcome to be indication or no indication. If r is less than this value, placed on the left side of the dashed line in figure 10.1, the test outcome is modelled to be indicating (I). Naturally, non-indication (NI) is modeled by the random number being greater than p_{T_i}.

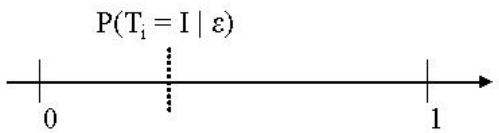

Figure 10.1. The unit interval in relation to the marginalized test probability

Random number	Test outcome
$r < p_{T_i}$	I
$r > p_{T_i}$	NI

Table 10.3. Generated test outcome by a random variable, r.

10.4 Simulation Results

To be able to understand the results from the simulations it is important to consider the model and the connections between the faulty component and other nodes in the model. For each simulation cycle the corresponding model, here called the *active model*, is presented and discussed in terms of specific DTCs and available tests.

In these simulations the cost distribution corresponds to the number of different costs that appear. These simulations are graphicly presented by plotting the frequency of a certain repair sequence vs the corresponding cost of repair.

10.4.1 Simulation Cycle A - Component 15 Faulty

The active model for cycle A

For component 15, DTC3 was predefined to be generated. DTC3 is a rather specific DTC, meaning that only 5 out of 17 components are connected to this DTC which can be seen in Figure 10.2. As a consequence, the generation of DTC3 makes a troubleshooting case easier since the number of possible actions decrease. At the beginning of troubleshooting there are 9 different actions to chose from, 4 tests and 5 components. The small number of possibly faulty components decreases the number of repair strategies.

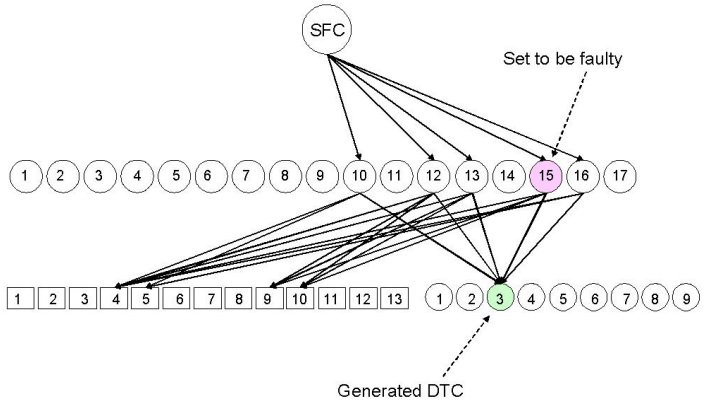

Figure 10.2. The active model for simulation A. We see that DTC3 is the generated DTC out of the 9 possible DTCs modelled by the nine nodes down to the right. From the 17 nodes representing the components there are 5 that are connected to DTC3, e.g. component 10, 12, 13, 15 and 16. Component 15 is modelled to be faulty. The square nodes to the left are the 13 tests in the XPI-model, out of which tests 4, 5, 9 and 10 are available for this combination of DTC and components.

The Three Cost Distributions for Cycle A

Figure 10.3 is a representation of the resulting costs from the three sorting criteria. We can observe that there are fewer unique costs for the efficiency sorting (topmost plot) and probability sorting (middle plot) and that these costs have higher frequency than the costs associated with random sorting (last plot). High frequency means that many repair procedures correspond to the same cost and thereby the same repair strategy. Both efficiency and probability sortings have been able to use only a few different repair sequences. The random sorting has used a large number of repair sequences giving many different cost of repair with low frequency. This is an observation that is general throughout these simulation results.

Let us concentrate on the topmost plot. We see that the efficiency sorting has a cost distribution of only three different repair sequences. The composition of this distribution can be understood by studying the repair sequences resulting in the specific costs.

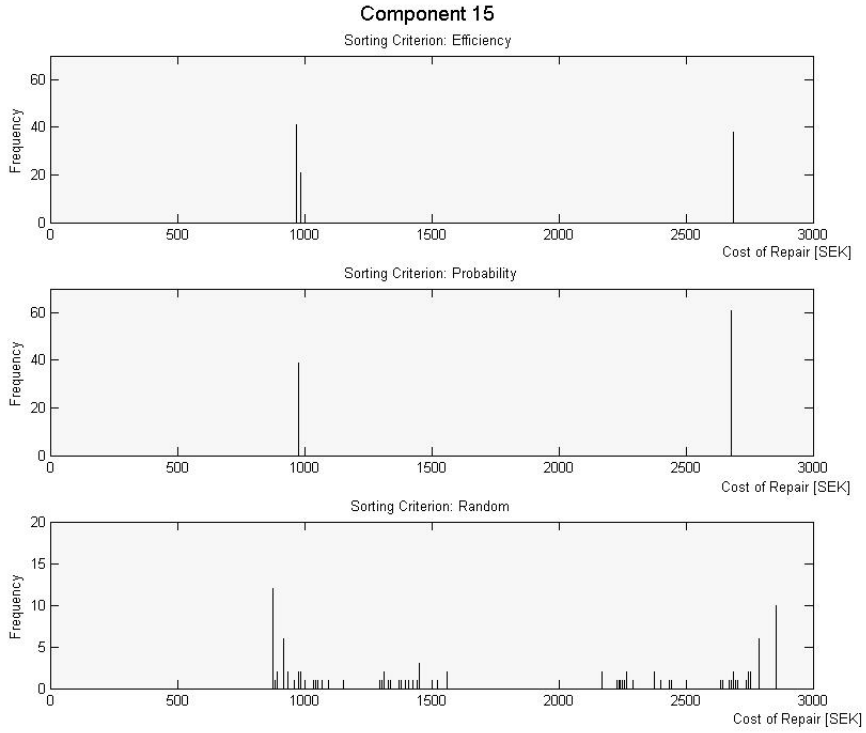

Figure 10.3. The cost distributions for simulation cycle A. The topmost plot shows the resulting cost distribution for the efficiency sorting, the middle plot shows corresponding result for the probability sorting and in the last plot the results for the random sorting is presented. All plots present the frequencies of specific repair sequences on the y-axis and the costs of repair for these sequences on the x-axis.

The three repair sequences for the efficiency sorting are presented in Table 10.4. To each test in the sequence, the test outcome is denoted.

Sequence	CR [SEK]	Repair sequence
A1	968	T9=I - 12 - 13 - 15
A2	985	T9=NI - T10=I - 12 - 13 - 15
A3	2688	T9=NI - T10=NI - 16 - 12 - 13 - 10 - 15

Table 10.4. The obtained repair sequences for the efficiency sorting in simulation cycle A.

We can see that all sequences begin with test number 9. This test checks whether or not there is an unusually strong smell of diesel, which is - for test indication - a sign of an external diesel leakage. The fault of component 15 is leakage and the test indication changes the probability distribution. Sequence A1 corresponds to this case, where component 12 and 13 are proposed to be repaired first since they have a higher efficiency compared to component 15.

Sequence A3, which is much more expensive, corresponds to the opposite case to A1. Test 9 does not indicate diesel smell and test 10 does not indicate a visible diesel leakage. When this information is added to the model, the probability for component 15 to be faulty decreases and the repair sequence gets longer and thereby more expensive.

For the middle sequence, A2, we can conclude that the indication of T10 is a strong evidence to the model and therefore we get the same behaviour as for sequence A1. But in this case it is important to reflect upon the fact that these two tests, both able to indicate external leakage, should most certainly give the same result. For a visible leakage, it is highly probable that there is a distinct smell of diesel as well. This means that this case is somewhat unrealistic.

Studying the middle plot we see that the cost distribution for the probability sorting has the same properties as the ones mentioned above. Since the DTC is rather specific, the troubleshooter only chooses between a few number of sequences where the test outcomes can mean the difference between higher and lower costs.

The random sorting does not show the same properties as the two above, here we can see many bars where the frequency is low. This is due to the fact that this sorting simply chooses one component out of the possible ones. All components are thereby equally probable to be the first one in the repair sequence and the risk for choosing a long repair strategy is higher.

10.4.2 Simulation Cycle B - Component 16 Faulty

The active model for cycle B

For cycle B we have both DTC1 and DTC3 generated. In Figure 10.4 we see that the active model for cycle B looks like the model for cycle A and the reason for this is that the “no false alarm” assumption means that the faulty component must be connected to all generated DTCs.

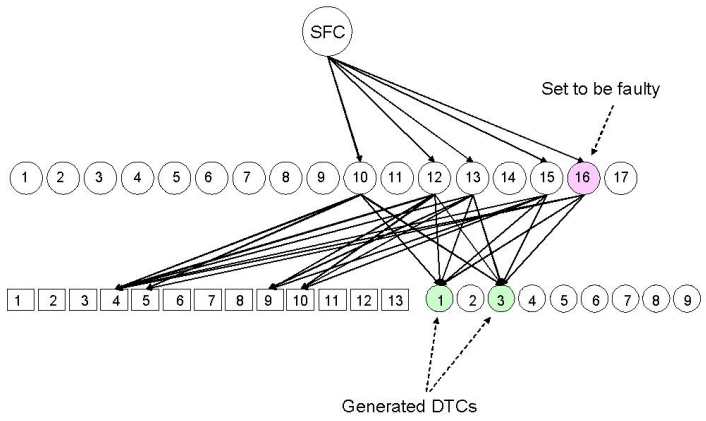

Figure 10.4. The active model for simulation B

The three cost distributions for cycle B

In the same way that the active models for simulation cycles A and B are similar, we can see great resemblance between the resulting plots presenting the three different cost distributions. The results for component 16 being faulty are presented in Figure 10.5.

If we study the sequences for this simulation cycle the results from the previous cycle are repeated. The two topmost plots, showing efficiency and probability sortings, have few sequences with high frequencies whereas the random sorting has a wider range of repair sequences. The sequences for the efficiency sorting in cycle B are presented in Table 10.5.

Sequence	CR [SEK]	Repair sequence
B1	436	T9=NI - T10=NI - 16
B2	744	T9=I - 12 - 13 - 15 - 16
B3	761	T9=NI - T10=I - 12 - 13 - 15 - 16

Table 10.5. The obtained repair sequences for the efficiency sorting in simulation cycle B.

The plots for cycle A and B are similar, but we can detect a difference in the actual sequences, which is that the tests indicate in the opposite way in cycle B. This is explained by the fault of component 16 being internal leakage. Internal leakage will not produce diesel smell and visible leakage and no test indication makes the troubleshooter find the right component quickly.

We can see the indication of a strong dependence between the two tests, T9 and T10, like we could do in cycle A, if we study sequence B2 and B3. The only difference

between these two sequences is that in B2 the test indicating is T9, in B3 it is T10 which indicates. The component order is the same.

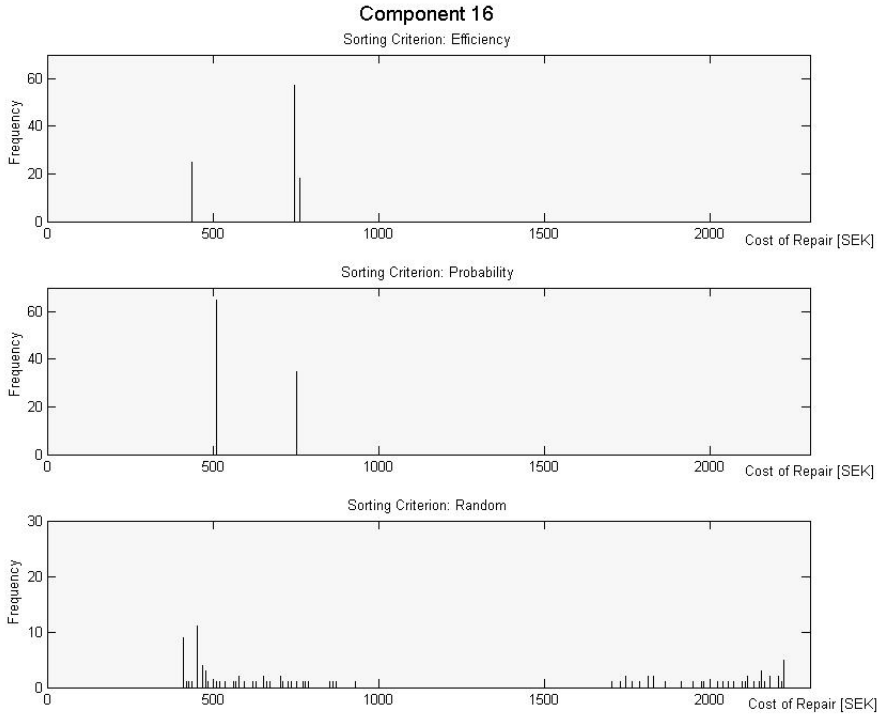

Figure 10.5. The cost distributions for simulated repair of component 16

10.4.3 Simulation Cycle C - Component 17 faulty

The active model for cycle C

Simulation cycle C is more complex since the generated DTC1 is connected to all components as we can see in Figure 10.6 and as a consequence all tests are also available. This naturally makes the troubleshooting more complex.

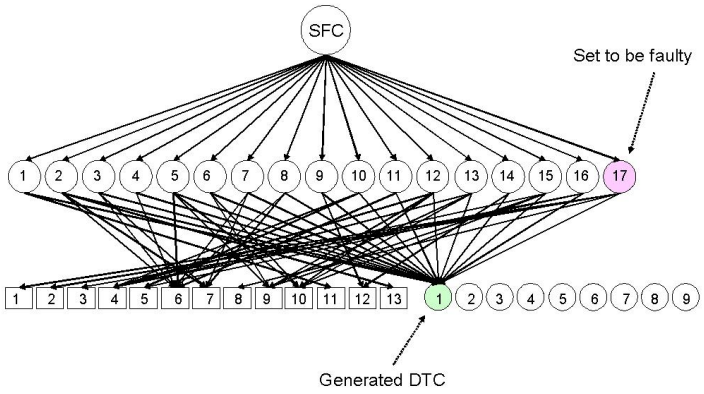

Figure 10.6. The active model for simulation C

The three cost distributions for cycle C

The results from these simulations are not as clear as the two previous ones. The fact that DTC1 is generated, enabling all components to be possibly faulty, makes it hard for the algorithm to find a small number of repair sequences. We see in Figure 10.7 that the cost distribution for all sortings is distributed rather equally on the cost interval. The repair costs are high when compared to costs for other cases. This is partly because component 17 is an expensive spare part, is hard (expensive) to exchange and has quite low probability of being faulty. These are characteristics that render low efficiency and thereby a position at the end of the repair strategy. Despite these facts we can find the same behaviour as in the two previous cases. The two highest bars (2641 and 5631 SEK) in the efficiency (topmost) plot corresponds to sequences which are equal until the 11th action which is the test T1, a test only connected to component 17. The outcome of this test determines the sequence to end (test indication) or to continue with 14 additional actions (no indication).

It is important to note that many of the bars with a frequency of 1 can be explained by being sequences where test outcomes are un-realistic, as they have a strong physical connection but are simulated to indicate differently due to the randomised test outcome.

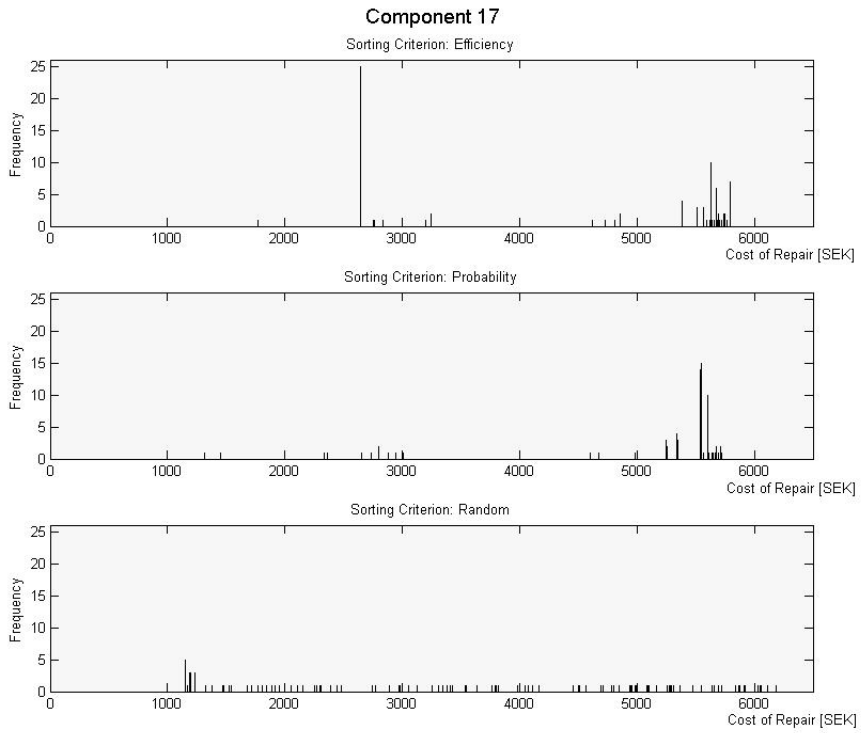

Figure 10.7. The cost distributions for simulated repair of component 17

10.4.4 Simulation Cycle D - Component 4 Faulty

The active model for cycle D

The active model for simulation cycle D, presented below in Figure 10.8, also has a large number of connections because of the generated trouble code being DTC1. The non-specific DTC returns a great number of possibly faulty components.

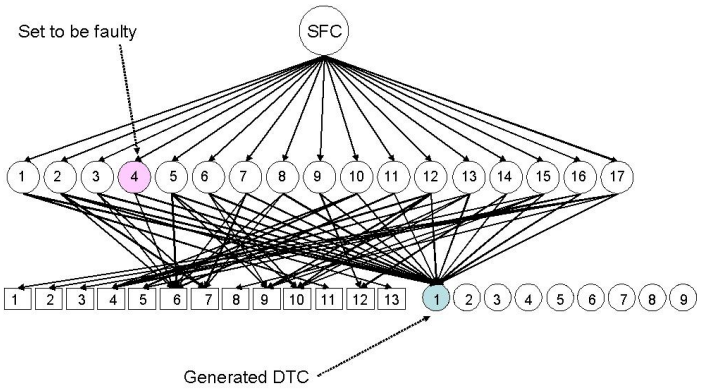

Figure 10.8. The active model for simulation D

The three cost distributions for cycle D

The resemblance to the results from component 17 is due to the same generated DTC. The difference is the cost for the spare part and replacing procedure which is lower for component 4 and as a consequence the range of the cost distributions are smaller, almost half the cost compared to cycle C. We also see that the frequencies are higher and the number of sequences are fewer, which implies a higher probability for component 4 to be faulty.

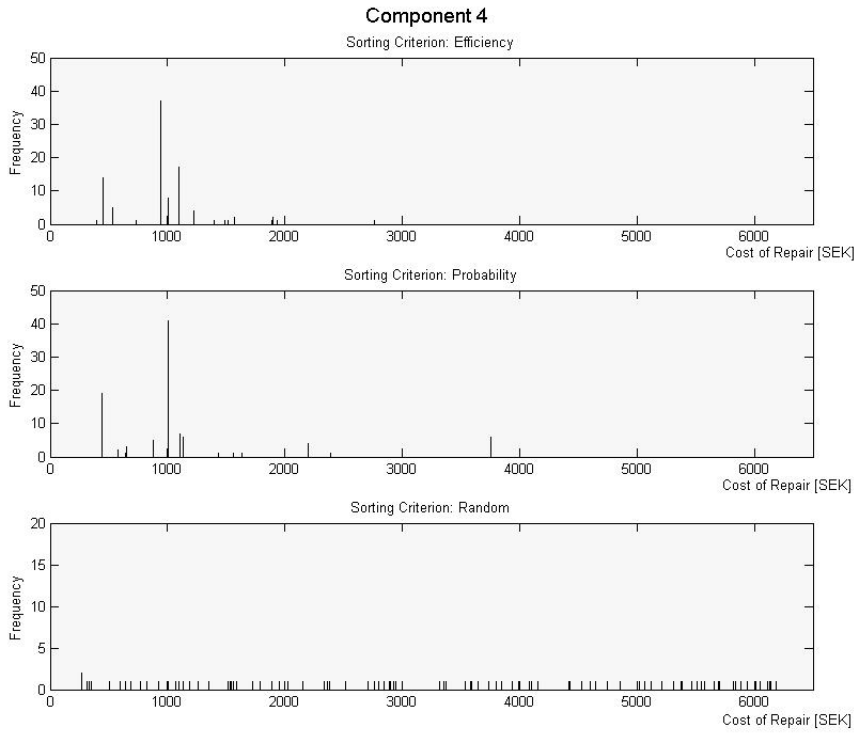

Figure 10.9. The cost distributions for simulated repair of component 4

10.4.5 Simulation Cycle E - Component 9 Faulty

The active model for cycle E

This simulation cycle is special since the generated DTC8 is very specific. As shown in Figure 10.10 the generated DTC8 only points out two components as possibly faulty. With only one test available, the troubleshooter has three different actions to choose from.

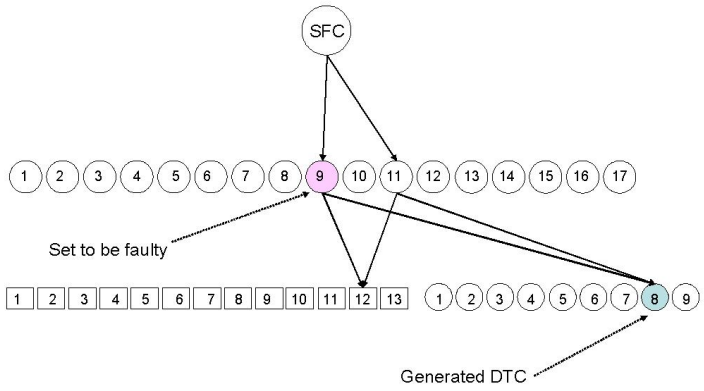

Figure 10.10. The active model for simulation E

The three cost distributions for cycle E

In total there are five different troubleshooting strategies available for a repair procedure in this simulation cycle. We can see that neither efficiency nor probability sorting uses all five repair sequences. The random sorting has generated all five sequences but as the sequences T_{12}, c_{11}, c_9 and c_{11}, T_{12}, c_9 give the same cost of repair, they are represented in the same bar (894 SEK).

If we study the result in the same way as the other simulations we recognise the same behaviour as before. The efficiency sorting chooses from two different sequences where the test outcome is what tells them apart. The probability sorting always chooses the same sequence, not performing any test but to repair the non-faulty component before the one modelled as faulty. By this we can conclude that the test is rather cheap and is performed by the efficiency sorting since it is considered to add valuable information. The probability sorting on the other hand, not considering costs, will not perform the test since the probability of the non-faulty component being faulty is high enough.

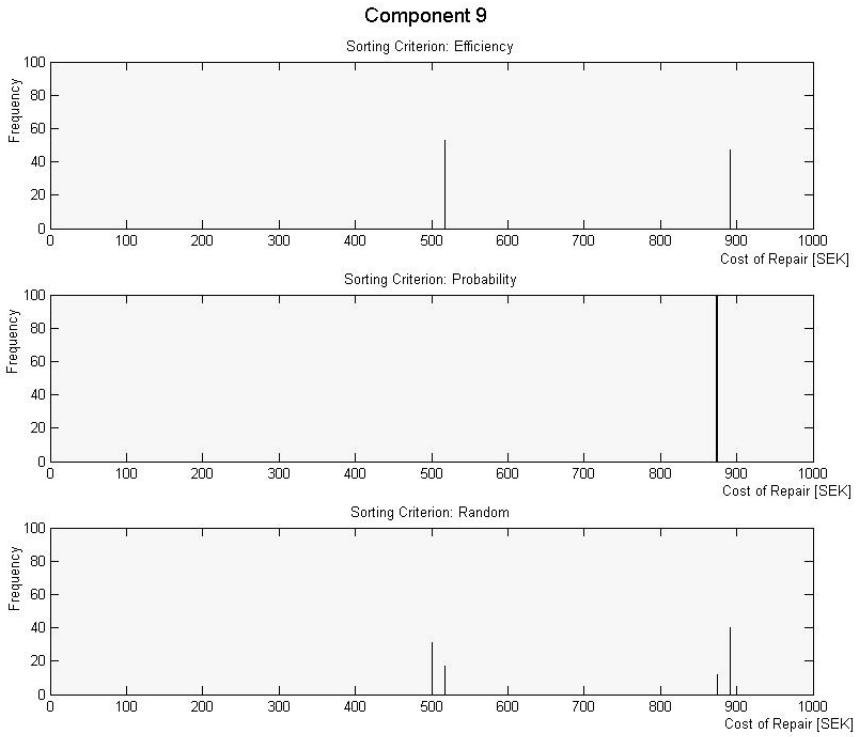

Figure 10.11. The cost distributions for simulated repair of component 9

One thing that becomes important in these simulations is the fact that the random ordering places the faulty component first in 20 % (five possible troubleshooting strategies) of the sequences. This happens in all simulations but is not of great importance since the number of sequences that start with the faulty component is small. In this case, where the faulty component can be repaired immediately with a probability of 1/3, the random sorting performs well in comparison to the other sortings.

10.4.6 The Average Cost of Repair

As a final comparison the average costs for all simulation cycles are presented. The average value is used as a rough estimation, and rating, of the different sortings. The values are presented in Table 10.6.

Simulation	A	B	C	D	E
Component	c_{15}	c_{16}	c_{17}	c_4	c_9
Efficiency	1625	670	4627	967	693
Probability	2015	596	5187	1117	874
Random	1779	1165	3610	3443	705

Table 10.6. The average cost of repair for all simulation cycles

We can see that the efficiency sorting criterion gives the lowest average cost in three out of five simulation cycles. In a small example like this it can be considered rather good, but if one was to use this sorting criterion in a troubleshooter program one would prefer it to produce the lowest average cost in a larger number of cases.

The average costs and the plots of the cost frequency show that this kind of probability-based reasoning and troubleshooting approach is highly affected by how specific the DTC is. For unspecific DTCs we get cycles similar to cycles C and D where all components are possibly faulty. In such a case, the cost of repair etc. is very important for how well the efficiency criterion performs. On the other hand, the advantages gained from using troubleshooting theory are lost when confronted with very specific DTCs. This is due to costs having too large an influence on the sorting, as seen in simulation E.

10.5 Conclusions

From the simulation results presented in this chapter we can conclude that the efficiency sorting criterion performs as well as or better than the other two criteria. Yet it is important to mention that this comparison only is relative to the sortings studied and can not give a measurement of how the efficiency criterion relates to an optimal sorting.

We find two important characteristics that affect the results and they are discussed further below.

10.5.1 Importance of Test Outcome

The most important effect of the results is the importance of the test outcomes. For a correct test indication, the repair sequence is shortened and the cost of repair is significantly decreased. This is interesting, and important to investigate further, since we know that including tests in the possible actions we can perform no longer guarantees optimality for the ECR theory. Does the fact that tests can shorten the repair sequence conflict with an optimal solution for this theory? One could argue that there exists a limit where a large number of reliable tests simplifies the troubleshooting process so much that the need for a troubleshooting prototype will disappear, but considering the complicated systems in a truck today, this is unlikely

to happen. The truck market is as much a computerised business as any market today.

The quality of a test is another interesting part of this project. It would be very profitable to find a way to measure how well a test has to perform to actually contribute to making a troubleshooting program more cost-efficient. Inefficient tests could be removed and further developed which could make tests more valuable to the troubleshooting process. For further discussion of the tests and their characteristics see [Lotz 2007].

10.5.2 Specificity of DTCs

The five different simulation cycles correspond to five different active models where the generated DTCs have varying specificity, i.e. the ability to exclude components as non-faulty. For the cycles with a relatively specific DTC (A and B), the troubleshooting process was more efficient and the number of different sequences were low. The opposite case (cycles C and D), where DTC1 was connected to all components, meant a more cumbersome troubleshooting process and higher repair costs. These results describe the general problems of troubleshooting a truck rather well.

The specificity of a DTC is important when troubleshooting a truck since we can choose to either use a troubleshooter program or do it manually. If every DTC was able to correctly point out exactly one component as faulty, it would be easier for the mechanic to find the faulty component and a program of this kind would be unnecessary. Unfortunately such specific diagnosis is hard to manage, especially for closed systems where components of the same type may be almost impossible to tell apart using only electrical diagnosis.

An advantage we can achieve by using a troubleshooting program is that it might be possible that even small improvements in DTC specificity can be considered as an improvement in the performance of a troubleshooter program. The simulations showed that the best results were achieved for a generated DTC that singled out five components as possibly faulty. If this is generalised it is reasonable to believe that further DTC development, towards DTCs which are good, but not excellent, in collaboration with correct test outcomes would make a program like this valuable when troubleshooting trucks.

Chapter 11

Discussion and Future Work

This chapter concludes the work and discusses some of the issues related to the relaxations of the assumptions in existing ECR theory.

11.1 Performance of the ECR Theory

The results from the simulations presented in Section 10 indicate that the ECR theory using an algorithm based on a sorting by efficiency, η, performs rather well in comparison to algorithms based on sorting by probability or a random order. However, the performance was highly affected by the characteristics of the generated DTCs and the test outcomes.

The importance of the test outcomes is interesting for further studies. It is clear that the "right" test outcome can shorten a repair sequence and thereby the cost of repair. From the algorithm point of view, tests make the construction of the algorithm more complex but may shorten the troubleshooting process significantly. Since we can see several advantages of including tests it would be very interesting to perform a study which aims to find an optimality criterion for a test-including ECR theory.

11.2 Relaxations of General Assumptions

The troubleshooting of a truck is a complex problem and is not easily adapted to existing troubleshooting theory. Using the theory of Expected Cost of Repair there are two specific problems that arise during the design of a truck troubleshooter prototype.

11.2.1 Dependent Costs

Considering the size of a truck and its great number of parts, the assumption that no cost of action is dependent on any previous actions is an important simplifica-

tion. As a consequence of this we have discussed an approach to handle dependent costs by using different levels of disassembly (Section 7.2) and thereby considering the transition costs (Section 7.3) between these levels.

The different levels of disassembly create a more realistic representation of a truck but they also create the need for always knowing the current, or previous, state of the truck. This is important when e.g. determining the troubleshooting strategy. The efficiency criterion demands observation costs for all components, which for dependent costs change as we move between different levels of disassembly. This means that we get an uncertainty in the strategy determination since the previous level of disassembly might be either the repair level or the control level for the previously chosen component. This will affect the observation costs.

The dependent costs are a new approximation and make the ECR theory even more uncertain. After the introduction of tests into the troubleshooting system the guarantee for optimality no longer holds, and the introduction of dependent costs is only an approach to model reality. The question of optimality for cases like this has to be studied throughly.

11.2.2 Performance of a Functional Control

Another assumption not adapted to the troubleshooting of trucks is the performance of a functional control after each repair action. When repairing a truck it is common to replace several parts before performing a functional control because the functional control often means that the cab has to be tilted back into its correct position which amounts to relatively much work.

The idea behind the demand for a functional control after each repair is to ensure whether the faulty component is found or not. Considering a Bayesian network, this information is the evidence that is used by the network to compute a new marginalised probability distribution. If we choose not to perform a functional control we can no longer provide the information obtained from the control and we are left without the tools we need for computing a new troubleshooting strategy. The functional control assumption was never abandoned because of this, but it will be an issue for future truck troubleshooting.

Bibliography

[Andersson 2006] Andersson J. (2006). *Minimizing Troubleshooting Costs - A Model of the HPI Injection System*. Master's Thesis. School of Electrical Engineering, Royal Institute of Technology.

[Gustavsson 2006] Gustavsson T. (2006). *Troubleshooting using cost-effective algorithms and bayesian networks*. Master's Thesis. School of Electrical Engineering, Royal Institute of Technology.

[Gut 1995] Gut A. (1995). *An Intermediate Course in Probability*. Springer Science+Business Media, Inc. ISBN:0-387-94507-5

[Heckerman 1995] Heckerman D. (1995). A Tutorial on Learning With Bayesian Networks. Technical Report MSR-TR-95-06. Microsoft Research.

[Heckerman et al. 1995] Heckerman D., Breese J., Rommelse K. (1995). Decision-Theoretic Troubleshooting. *Communications of the ACM*, 38(3):49-56. ISSN:0001-0782

[Hillier & Lieberman 2005] Hillier F. S., Lieberman G. J. (2005). *Introduction To Operations Research*. Eighth edition. The McGraw-Hill Companies, Inc. ISBN:007-123828-X

[Jaynes 2001] Jaynes E. T. (2001). *Probability Theory - the Logic of Science*. Cambridge University Press. Cambridge. ISBN:052-159271-2

[Jensen 2001] Jensen F. V. (2001). *Bayesian Networks and Decision Graphs*. Springer-Verlag, New York Inc. ISBN:0-387-95259-4.

[Jensen et al. 2000] Jensen F. V., Kjaerulff U., Kristiansen B., Langseth H., Skaanning C., Vomlel J., Vomlelová M. (2000). The SACSO methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 2001*, 15(4):321-333. ISSN:0890-0604

[Langseth & Jensen 2003] Langseth H., Jensen F. V. (2003). Decision Theoretic Troubleshooting of Coherent Systems. *Reliability Engineering and System Safety*, 80(1):49–62.

[Lotz 2007] Lotz K. (2007). *Optimizing Guided Troubleshooting Using Interactive Tests and Bayesian Networks with an Application to Diesel Engine Diagnosis.* Master's Thesis. Department of Mathematics, Royal Institute of Technology.

[Mossberg 2007] Mossberg J. (2007). *Bayesian Modeling of A Diesel Injection System for Optimal Troubleshooting.* Master's Thesis. Department of Mathematics, Royal Institute of Technology.

[Murphy 2001] Murphy K. P. (2001). *A Brief Introduction to Graphical Models and Bayesian Networks.*
URL: `http://www.cs.ubc.ca/~murphyk/Papers/intro_gm.pdf`

[Pernestål 2007] Pernestål A. (2007) *A Bayesian Approach to Fault Isolation with Application to Diesel Engine Diagnosis.* Licentiate Thesis. School of Electrical Engineering, Royal Institute of Technology.

[Scania 2006] (2006). *Funktionsbeskrivning, XPI.* Internal Scania document.

[Wikipedia 2007] Wikipedia. (2007-08-02). *Search term: 'Diesel engine'.* `http://en.wikipedia.org/wiki/Diesel_engine`.

Unprinted References

[Flink 2007] Flink T. (2007). Tomas Flink, employed at Scania CV AB. Interviews during April to June 2007.

[Telborn 2007] Telborn K. (2007). Klas Telborn, employed at Scania CV AB. Interviews during April to June 2007.

Picture References

[Figure 8.1] `http://www1.eere.energy.gov/vehiclesandfuels/pdfs/basics/jtb_diesel_engine.pdf`

[Figure 8.2] `http://www.britannica.com/eb/art-19423/Four-stroke-diesel-engine-The-typical-sequence-of-cycle-events?articleTypeId=1`

Appendix A

List of Notation

XPI	Extreme High Pressure Injection
LPP	Low Pressure Pump
HPP	High Pressure Pump
IMV	Inlet Metering Valve
MDV	Metering Dump Valve
HPC	High Pressure Connector
DTC	Diagnostic Trouble Code
ECU	Electric Control Unit
BNT	Bayesian Network Toolbox for Matlab
TS	Troubleshooting
S	*strategy*, the order of actions in which the troubleshooting process works
VOI	Value Of Information
SFC	Single Fault Constraint
CPT	Conditional Probability Table
c_i	system component numbered i
F	component is faulty
NF	component is not faulty
C_i^o	cost of observation for component c_i
C_i^r	cost of reparation for component c_i
C_i^T	cost of performing test T_i
C^s	cost of observing whether the system is functioning normally
A_i	action in troubleshooting process, i.e. repair, observation or test
$ECR(S)$	Expected Cost of Repair for the strategy S
$ECRT(T)$	Expected Cost of Repair Including Test for the test T
η	*efficiency*, ratio between probability and cost of observation
ϵ	given state of knowledge, evidence
p_i	probability for component i to be faulty, given the evidence $P(c_i\|\epsilon)$
p_T	probability for test outcome

I	test outcome, indicating failure
NI	test outcome, not indicating failure
N_T	number of possible tests belonging to a system
N_A	number of possible actions in a troubleshooting case
l_i	level of disassembly at step i
λ_j	jth level of disassembly

Appendix B

Basics of Probability Theory

In this appendix we go through some fundamental rules of probability theory. Some basic rules in probability theory are necessary in this thesis, and in this section we will briefly discuss the joint probability, the marginal probability and the Chain Rule.

We will assume X, Y and Z to be discrete random variables, the continuous case being analogous. The range of X, Y and Z are denoted S_X, S_Y and S_Z, respectively. To each of these we assign a probability mass function, which for X is written as

$$P(X = x) = p_X(x) \qquad \forall \quad x \in S_X \tag{B.1}$$

B.1 The Basic Rules

Two well-known facts can be stated about probability mass functions.

$$p_X(x) \in [0, 1] \qquad \forall \quad x \in S_X \tag{B.2}$$

and

$$\sum_{x \in S_X} p_X(x) = 1 \tag{B.3}$$

As mentioned above, the continuous case is analogous, except that we use the probability density function instead of the probability mass function, and that the sums are replaced by integrals. The probability density function is defined as the function $f_X(x)$ that satisfies

$$P(a < X < b) = \int_a^b f_X(x)dx \tag{B.4}$$

A more comprehensive disquisition can be found in for instance [Gut 1995].

B.2 The Joint Probability

The joint probability mass function captures dependencies between two or more random variables. For the two random variables X and Y, the joint probability mass function is

$$P(X = x, Y = y) = p_{X,Y}(x, y) \tag{B.5}$$

It is clear that

$$\sum_{\substack{x \in S_X \\ y \in S_Y}} p_{X,Y}(x, y) = 1 \tag{B.6}$$

where $p_{X,Y}(x, y)$ fulfils the facts mentioned above.

$$p_{X,Y}(x, y) \in [0, 1] \qquad \forall \quad x \in S_X, y \in S_Y \tag{B.7}$$

Furthermore, X and Y are said to be independent if $p_{X,Y}(x, y) = p_X(x)p_Y(y)$.

B.3 The Marginal Probability

The marginal distribution of X is calculated by summing the joint probability distribution over all possible values of Y.

$$p_X(x) = \sum_{y \in S_Y} p_{X,Y}(x, y) \tag{B.8}$$

If we are given the value of Y, and $p_Y(y) \neq 0$, the distribution of X given Y is

$$p_{X|Y}(x) = \frac{p_{X,Y}(x, y)}{p_Y(y)} \tag{B.9}$$

If X and Y are independent, we know that $p_{X,Y}(x, y) = p_X(x)p_Y(y)$. If we insert this into the formula above, we see that under these circumstances $p_{X|Y}(x) = p_X(x)$.

B.4 The Chain Rule

If we rearrange the definition of conditional probability we obtain the Chain Rule.

$$p_{X,Y}(x, y) = p_{X|Y}(x)p_Y(y) = p_{Y|X}(y)p_X(x) \tag{B.10}$$

Quite often in this thesis, some variables have known values. At this stage we use the random variable Z. The variables with known values can be described by letting the value of Z be z. Then the chain rule is written as:

$$p_{X,Y|Z=z}(x, y) = p_{X|Y,Z=z}(x)p_{Y|Z=z}(y) = p_{Y|X,Z=z}(y)p_{X|Z=z}(x) \tag{B.11}$$

Appendix C

List of Diagnostic Trouble Codes

DTC_1 Fuel rail pressure is excessively below command is generated if the measured fuel pressure in the rail is excessively below command.

DTC_2 Fuel rail pressure is excessively above command is generated if the measured fuel pressure in the rail is excessively above command.

DTC_3 Fuel rail pressure, volume leak is generated if the rail pressure varies in a way that makes it likely to be a leak somewhere on the high pressure side.

DTC_4 Fuel rail pressure sensor, stuck or electrical faults is generated if the rail pressure sensor measures exactly the same pressure for a longer time or if there is an electrical fault caused by the sensor.

DTC_5 Fuel rail pressure is excessively high is generated if the measured fuel pressure in the rail exceeds a limit value over a certain time.

DTC_6 IMV electrical faults is generated if there is an electrical fault connected to the IMV.

DTC_7 MDV tripped is generated if the MDV is tripped, which it should be only when the rail pressure exceeds a limit value.

DTC_8 Plausable leakage in the IMV is generated if the rail pressure increases when the IMV is commanded to be fully closed.

DTC_9 Cylinder balancing compensation has reached min or max is generated when the cylinder balancing compensation value has reached the minimum or maximum value for any of the cylinders.

Appendix D

List of Tests

T_1: Cylinder balancing test ECU test to test the injectors.

T_2: Fall down test ECU test to test the injectors.

T_3: Cylinder cut off test ECU test to test the injectors.

T_4: Raise in rail pressure test ECU test to analyse the behaviour when the rail pressure is increased by the ECU.

T_5: Pressure measurement in rebound strip Measurement of the pressure in the rebound strip.

T_6: LPP pressure measurement Measurement of the pressure at the LPP.

T_7: Control of air in fuel Measurement to see if there are bubbles of air in the fuel.

T_8: Injector test Measurement. The injector is mounted in a specific test environment.

T_9: Smell of diesel Inspection, observing whether there is an abnormal smell of diesel or not.

T_{10}: Visible diesel leakage Inspection, searching for visible leakage.

T_{11}: Fuel in tank Inspection of fuel amount in fuel tank.

T_{12}: MDV pipe hot Inspection to observe whether the MDV pipe is heated.

T_{13}: Control of connectors pre LPP Inspection of pipes and connectors on the suction side.

TRITA-CSC-E 2008:026
ISRN-KTH/CSC/E--08/026--SE
ISSN-1653-5715

www.kth.se