Process plant alarm diagnosis using synthesised fault tree knowledge

This item was submitted to Loughborough University’s Institutional Repository by the/author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfillment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/7258

Publisher: © Andrew John Trenchard

Please cite the published version.
This item is held in Loughborough University’s Institutional Repository (https://dspace.lboro.ac.uk/) and was harvested from the British Library’s EThOS service (http://www.ethos.bl.uk/). It is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
Process Plant Alarm Diagnosis
Using Synthesised Fault Tree Knowledge

by

Andrew John Trenchard

A Doctoral Thesis
Submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy of the Loughborough University of Technology

1990

© by Andrew John Trenchard 1990
Acknowledgements

The author wishes to thank the Science and Engineering Research Council, in co-operation with British Petroleum Oil International, for supporting this work, Dr. P.K. Andow, Professor F.P. Less and Professor B.A. Buffam for their advice and encouragement and Mr. J.S. Mullhi and Mr. A. Hunt for their technical assistance. Many thanks are also expressed to Mr. E.J. Horne, Mr. J. Matthews, Mr. A. Farrow, Mr. Vijay Vedwana and Mr. J. Haberfield of British Petroleum, the staff of the Chemical Engineering Department, KBC Process Automation for the use of their facilities and not least my wife Sue for her patience.
Abstract

The development of computer based tools, to assist process plant operators in their task of fault/alarm diagnosis, has received much attention over the last twenty five years. More recently, with the emergence of Artificial Intelligence (AI) technology, the research activity in this subject area has heightened. As a result, there are a great variety of fault diagnosis methodologies, using many different approaches to represent the fault propagation behaviour of process plant. These range in complexity from steady state quantitative models to more abstract definitions of the relationships between process alarms.

Unfortunately, very few of the techniques have been tried and tested on process plant and even fewer have been judged to be commercial successes. One of the outstanding problems still remains the time and effort required to understand and model the fault propagation behaviour of each considered process.

This thesis describes the development of an experimental knowledge based system (KBS) to diagnose process plant faults, as indicated by process variable alarms. In an attempt to minimise the modelling effort, the KBS has been designed to infer diagnoses using a fault tree representation of the process behaviour, generated using an existing fault tree synthesis package (FAULTFINDER). The process is described to FAULTFINDER as a configuration of unit models, derived from a standard model library or by tailoring existing models.

The resultant alarm diagnosis methodology appears to work well for hard (non-rectifying) faults, but is likely to be less robust when attempting to diagnose intermittent faults and transient behaviour.

The synthesised fault trees were found to contain the bulk of the information required for the diagnostic task, however, this needed to be augmented with extra information in certain circumstances.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>i</td>
</tr>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2 LITERATURE SURVEY</td>
<td>7</td>
</tr>
<tr>
<td>3 THE ALARM DIAGNOSIS METHODOLOGY</td>
<td>42</td>
</tr>
<tr>
<td>3.1 The Fault Modelling Technique</td>
<td>42</td>
</tr>
<tr>
<td>3.2 Representing The Fault Trees Within The KBS</td>
<td>44</td>
</tr>
<tr>
<td>3.2.1 The Structure Of The If-Then Rules Within PROLOG</td>
<td>48</td>
</tr>
<tr>
<td>3.2.2 Direct Versus Indirect Alarm Fault Trees</td>
<td>51</td>
</tr>
<tr>
<td>3.2.3 Defining Additional Boundary Conditions</td>
<td>54</td>
</tr>
<tr>
<td>3.2.4 Representing Process Sub-systems</td>
<td>54</td>
</tr>
<tr>
<td>3.3 An Overview Of The Alarm Diagnosis Strategy</td>
<td>57</td>
</tr>
<tr>
<td>4 CONJUGATING THE INDIVIDUAL ALARM DIAGNOSES</td>
<td>62</td>
</tr>
<tr>
<td>4.1 The Detection Of The Causal Couplings Between Alarms</td>
<td>62</td>
</tr>
<tr>
<td>4.2 Deciding How The Related Alarms Should Be Combined</td>
<td>69</td>
</tr>
</tbody>
</table>
4.3.1 Taking Into Consideration Latent Alarm Failures

4.3 Combining The Alarm Causes

4.3.1 Combining The Lists Of Alarm Causes
4.3.2 The Problem Of Combinatorial Explosion
4.3.3 Conjugating The Alarm Causes

4.4 Checking the Cutsets for Logical Consistency

4.4.1 Conflicts With Process Variable Deviation States
4.4.2 Conflicts With Device Failure States
4.4.3 Conflicts With Device Working Conditions
4.4.4 Conflicts With Subsystem Failure Or Working States
4.4.5 Conflicts with Spurious and Passive Alarm Failures
4.4.6 Conflicts Between Boundary Conditions
4.4.7 Conflicts Between Assumed States Of Alarmed Variables

4.5 Simplifying the Cutsets

4.6 Checking the Cutsets for Minimality

5 RANKING THE INDIVIDUAL ALARM CAUSES

5.1 Reasoning With Uncertainty
5.2 Frequencies Of Single Faults
5.3 Frequencies Of Fault Combinations

5.3.1 Probabilities Of Enabling Faults
5.3.2 Probabilities Of Initiating Faults
5.4 Frequencies Of Variable Deviations

5.4.1 Calculation Of The Base Frequency
5.4.2 Interpreting The Monitored Variable Indications
5.4.3 Continuous Variable Discretisation
5.4.4 Modelling Instrumentation Faults
5.4.5 Multiple Indications Of The Same Variable
5.4.6 Common Instrumentation Devices

5.5 Frequencies Of Grouped Failure Modes
5.6 Frequencies Of Spurious Alarm Failures
5.7 Probabilities Of Passive Alarm Failures
5.8 Failure Rate Data Sources

6 THE IMPLEMENTATION OF THE FAULT DIAGNOSIS SYSTEM

6.1 The Process Simulation Software

6.1.1 Process Variable Database
6.1.2 The Database Initialisation Program
6.1.3 The Alarm Detection Task
6.1.4 The Supervisor's Console Task
6.1.5 The Reactor Charging System Dynamic Simulation
6.1.6 The Plant Monitoring Task
6.1.7 The Process Log Playback Task
6.1.8 Installing And Starting The Process Simulation

6.2 The Development KBS

6.2.1 The Languages Used Within The Development KBS
6.2.2 The KBS Control Program
6.2.3 The Implementation Of The Inference Mechanism
6.2.4 The Rulebase 149
6.2.5 The Interface Between The KBS And The Process Variable Database

6.2.5.1 Calculation Of The Fuzzy Logic Membership Functions 150

6.2.6 The Operator Interface 151
6.2.7 The Rulebase Compiler 152

7 THE APPLICATION OF FAULTFINDER TO THE SYSTEMATIC GENERATION OF THE ALARM DIAGNOSIS INFORMATION 153

7.1 The Reactor Charging System 153

7.1.1 Modelling Process Plant Using FAULTFINDER 159
7.1.2 Generating The System Specific Unit Models 163
7.1.3 The System Configuration 170
7.1.4 Generating The Alarm Fault Trees And Cutsets 172
7.1.5 Converting The Fault Trees Into The Fault And Indication Rules 179

7.2 Modelling The Batch Distillation Process 183

7.2.1 System Description 184
7.2.2 The Modelling Simplifications 191
7.2.3 Generating The System Specific Models 194

7.2.3.1 Modelling the Condenser Unit 194
7.2.3.2 Modelling The Distillation Column 199
7.2.3.3 The Reboiler Model 203
7.2.3.4 Creating The Remaining Models 206

7.3 The Modelling Problems Encountered 207
8 WORKED EXAMPLES OF THE FAULT DIAGNOSIS METHODOLOGY 211

8.1 The Reactor Charging System 211

8.1.1 Example 1: Loss Of Fluid From The Buffer Tank T2 211

8.1.1.1 Diagnosing The High Tank Inlet Flow Alarm 213
8.1.1.2 Diagnosing The Low Buffer Tank Level Alarm 215
8.1.1.3 Diagnosing The Low Tank Discharge Flow Alarm 221
8.1.1.4 Diagnosing The Very Low Buffer Tank Level Alarm 226

8.1.2 Example 2: Control Valve Cv_1 Fails Closed And Flow Transducer Trans 1 Fails Invariant 230

8.1.2.1 Diagnosing The Causes Of The Very Low Flow 232
8.1.2.2 Diagnosing The Low Buffer Tank Level Alarm 238

8.2 The Batch Distillation Pilot Plant 243

8.2.1 Example 3: Loss Of Power To The Reboiler Heater 243

8.2.1.1 Diagnosing The Low Reboiler Vapour Temperature 246
8.2.1.2 Diagnosing The Low Column Pressure And Reboiler Temperature Alarms 247
8.2.1.3 Diagnosing The Low Condenser Vapour Temperature Alarms 249
8.2.1.4 Diagnosing The Low Column Wall Temperature Alarms 253
8.2.2 Example 4: Nitrogen Leak Into The Column And Pressure Control Loop PIC5 Fails Inactive

8.2.2.1 Diagnosing The high Column Pressure Alarm
8.2.2.2 Diagnosing The High Reboiler Temperature Alarm
8.2.2.3 Diagnosing The Low Temperature Alarms of Indications Ti3, Ti8, Ti12 And Ti14
8.2.2.4 Diagnosing The High Column Wall Temperature Alarms

9 DISCUSSION AND CONCLUSIONS

9.1 The Diagnosis Methodology

9.1.1 The Single Fault Assumption

9.1.1.1 Considering Multiple Fault Alarm Causes
9.1.1.2 Common Mode Failures
9.1.1.3 Design Philosophy
9.1.1.4 The Drawbacks Of The Multiple Fault Assumption

9.1.2 Reasoning With Time
9.1.3 The Level Of Fault Discrimination
9.1.4 The Use Of Failure Frequencies
9.1.5 The Indication Interpretation Technique

9.2 The User Interface

9.2.1 The Explanation Facility
9.2.2 The Operator Driven Diagnosis System
9.3 Synthesising The Diagnostic Rules Using FAULTFINDER

9.4 System Architecture And Implementation Tools

9.4.1 The POPLOG Language Environment

9.5 Representing The Behaviour Of Process Plant

9.6 Future Developments

9.6.1 The Design Of The Process Model

9.7 Conclusions

9.7.1 The Strengths Of The Methodology

9.7.2 The Weaknesses Of The Methodology

9.8 Future Work

9.8.1 Applications Of The Diagnosis System

9.8.2 The Use Of Quantitative Diagnostic Models

9.8.3 Target Applications Of Diagnosis Systems

References

APPENDIX (Volume II)

A THE DEFINITIONS OF THE PROLOG OPERATORS

B SIMULATING THE REACTOR CHARGING SYSTEM

C THE KBS CONTROL PROGRAM

C.1 Reading Input From The Users Terminal

C.2 Checking The Process Variable Database For New Alarms
C.3 Updating The Current Time And Date
C.4 Retrieving Process Variable Information From The Database

D THE INFORMATION SPECIFIED TO THE RULEBASE COMPILER
D.1 The Process Variable Description
D.2 Causal And Indication Rules
D.3 Basic Event Definitions

E THE USER GRAPHIC SCREENS SUPPORTED BY THE KBS
E.1 The Unaccepted Alarm Message Display
E.2 The Accepted Alarm Message Display
E.3 The Facia Panel Mimic Screen
E.4 The Alarm Diagnosis Display

F THE FUNCTIONALITY OF THE RULEBASE COMPILER
F.1 Parsing The Rulebase
F.2 Compiling The Rulebase Information

G THE FAULTFINDER UNIT MODELS

H THE TYPICAL OUTPUT FROM THE PLOT PROGRAM

I THE UNCOMPILED RULEBASE FOR THE REACTOR CHARGING SYSTEM

J THE UNCOMPILED RULEBASE FOR THE BATCH DISTILLATION COLUMN

K THE COMPILED RULEBASE FOR THE REACTOR CHARGING SYSTEM
TABLE

4.1 The Logical Consistency Checks 89
7.1 The Manifest of Units in Figure 7.1 155
7.2 The Process Variable And Indication Names 157
7.3 The Failure Mode Mnemonics For Figure 7.7 179
7.4 The Manifest Of Equipment In Figure 7.8 186
7.5 The Definitions Of The Variables Used Within The KBS 189
7.6 The Permissible Variable Deviation States 209

H.1 The Failure Mode Mnemonics For Figure H.1 390

FIGURE

2.1 Simple Alarm Tree 11
2.2 Alarm Checklist 15
2.3 Example Cause Consequence Diagram 17
2.4 Time Delay Cause Consequence Diagram 21
2.5 The Closed Tank Model 23
2.6 The Process Variable Alarm Tree 24
2.7 Mini Fault Tree For A Low Flow Deviation 27
2.8 Simplified Consequence Diagram 32
2.9 The Architecture Of The ESCORT System 40

3.1 The Buffer Tank System 45
3.2 The Reduced Alarm Fault Tree 46
3.3 The Hierarchy Of The Prolog Operators 49
3.4 The Direct Alarm Fault Trees 52
3.5 The Indirect Alarm Fault Trees 53
3.6a The Causes of Alarm A 58
3.6b The Causes of Alarm B 59
3.6c The Causes of Alarm C 59
3.7 The Causal Relationships Between The Three Alarms 60
4.1 The Buffer Tank System  
4.2 The Continuously Stirred Mixer Tank  
4.3 The Simple Event Tree  
4.4 The Second Latent Alarm Failure Scenario  
4.5 The Third Latent Alarm Failure Scenario  
4.6 The Causal Searching Strategy  
4.7 The Number Of Minimum Cutsets VS. The Number Of Alarms  
4.8 The Example Alarm Fault Trees  
4.9 The level controlled buffer tank  
4.10 Steam Heat Exchanger  
4.11a The Mini-Fault Tree for a Low Steam Flow Alarm  
4.11b The Mini-Fault Tree for a Low Outlet  
4.12 The Modified Heat Exchanger Example  
4.13a The Causes of a Low Steam Pressure Alarm  
4.13b The Causes of a Low Temperature Alarm  

5.1 The Failure History Of Events A, B and C  
5.2 Fuzzy Logic Membership Functions  
5.3 The Pressure Sensing System  
5.4 Duplicate Indications Of A Process Variable  
5.5 The Spread Of Indications Of Variable R  
5.6 Common Instrumentation Paths  

6.1 The Software Organisation  
6.2 The KBS Program Modules  
6.3 The Definition Of The Fuzzy Membership Functions  

7.1 The Reactor Charging System  
7.2 The Fuzzy Discretisation Graph for Pressure Pl  
7.3 The Open Tank Unit  
7.4 The Causes Of A Low Tank Level  
7.5a The Causes Of A High Output Signal  
7.5b The Causes Of A Low Output Signal  
7.5c The Causes Of No Output Signal  
7.6 Block Diagram For The Reactor Charging System  
7.7 The Low Flow Alarm Fault Tree  
7.8 The Schematic Diagram Of The Batch Distillation Column  

xii
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.9</td>
<td>The Block Diagram For The Batch Distillation Column</td>
<td>190</td>
</tr>
<tr>
<td>7.10</td>
<td>The Condenser Model Port Definitions</td>
<td>195</td>
</tr>
<tr>
<td>7.11</td>
<td>The Distillation Column Port Definition</td>
<td>199</td>
</tr>
<tr>
<td>7.12</td>
<td>The Reboiler Model Port Definitions</td>
<td>203</td>
</tr>
<tr>
<td>8.1</td>
<td>The Reactor Charging System</td>
<td>212</td>
</tr>
<tr>
<td>8.2</td>
<td>Flow Fl Indication Instrumentation</td>
<td>237</td>
</tr>
<tr>
<td>8.3</td>
<td>The Batch Distillation Process</td>
<td>244</td>
</tr>
<tr>
<td>8.4</td>
<td>The Reboiler Liquid Temperature Histories</td>
<td>245</td>
</tr>
<tr>
<td>9.1</td>
<td>The Parallel Pumping System</td>
<td>263</td>
</tr>
<tr>
<td>9.2</td>
<td>The Upstream Section Of The Buffer Tank</td>
<td>267</td>
</tr>
<tr>
<td>9.3</td>
<td>The LPG Distillation Column</td>
<td>270</td>
</tr>
<tr>
<td>B.1</td>
<td>The Modelled Process Plant</td>
<td>311</td>
</tr>
<tr>
<td>B.2</td>
<td>Modelling Pipe Leakages</td>
<td>313</td>
</tr>
<tr>
<td>E.1</td>
<td>The Alarm Diagnosis Graphic</td>
<td>325</td>
</tr>
<tr>
<td>H.1</td>
<td>Fault Tree For A Low Output Signal From Transducer</td>
<td>385</td>
</tr>
</tbody>
</table>

Transl
Most modern large scale process plants are operated from centralised control rooms, by relatively few process personnel. In the majority of cases, process control computers are used to perform the regulatory control functions, and some of the more advanced control tasks. However, the overall day to day supervision of the plant is still the responsibility of the process operators.

When the plant is running normally, the process can be managed very efficiently by a small number of personnel. Consequently each operator will usually be responsible for a large section of plant. When a process fault occurs, it is the operator's task to diagnose, and if possible correct for the fault as quickly as possible. If the process is equipped with automatic trip systems, then economic savings will usually result if the operator can prevent a plant shutdown. Where such equipment is lacking, the operator has the dual responsibility of either restoring the plant to its normal state or guiding it to a safe shutdown.

Conventionally, the operator's attention is directed towards a fault by the alarm system. These alarms are usually associated with specific items of equipment or the key process variables. For example, if an important pump stops running, when it should not, then this should generate an alarm. Similarly, if a key process variable deviates too far from its normal value, then this should also cause an alarm.
The alarms are usually presented to the operator in one of two ways. The more traditional method, which is often retained for hard-wired alarms, uses small facia panels inscribed with the alarm messages. In recent years these have been superseded, to some extent, by the computer VDU. When an alarm becomes active, a siren or klaxon is usually sounded, and the relevant alarm message is flashed. After accepting the message, the audible warning ceases and the message is illuminated / displayed constantly.

In practice, the operator's task of real-time fault diagnosis is usually further complicated for the following reasons:

1 Many alarm systems are defined without a clear design philosophy. This often results in a lack of distinction between the genuine alarms and the process status information. Furthermore, unless the alarm limit values have been carefully selected, some alarms might be prone to oscillating with the process noise.

2 There are problems with the methods used to display the alarm information. It is physically difficult to mount a large number of alarm facias in a given area, without making the surveillance task impossible for the operator. Alternatively, the advantages of using computer VDU's are easily lost, unless the screen is organised very carefully.

3 The process operators often receive very little formal training. In most cases their diagnostic skills are based on experiential knowledge rather than an appreciation of the fundamental theory involved. As a consequence the operators are very adept at solving commonly occurring problems, but they encounter more difficulty when attempting to diagnose rarer faults.

4 The incidence of instrument failures is relatively high on process plant. The individual instruments may be quite reliable, but because they are present in such large numbers, the probability of
at least one being in a failed state is often quite high. Since the diagnostic task relies so heavily on the process indications and alarms, the operator must attempt to corroborate as much information as possible with other indications.

5 When the monitored process variables are highly interrelated, a single fault will often trigger numerous alarms in rapid succession. This may lead to a information overload of the process operator, especially since the control room environment of flashing alarm signals and audible warnings does not aid clear thinking. Given the above problems, and that the penalty for plant maloperation can be high, it is surprising that more accidents are not caused by human error.

The difficulty of the operator's diagnostic task has been recognised for a long time. Almost as soon as computers were being used to detect and display alarms, there was considerable interest in their application to the fault diagnosis problem. The main attraction of the computer was the relative ease with which logical operations and hence pattern recognition functions could be performed. It was hoped that at least some of the operator's diagnostic skills could be emulated, with greater speed and consistency.

Unfortunately the high cost of developing the computer software, and the process plant models, restricted most of the early applications to within the nuclear industry. The first reported alarm analysis systems were implemented in the UK, in the mid to late 1960's, at the Wylfa and Oldbury-on-Severn nuclear power stations. In both cases a large number of alarms were involved, therefore the need for some form of alarm analysis was identified, to avoid the information overload problem.

There was renewed interest in the development of computer based operator aids in the mid 1970's. Two major studies into the application of disturbance analysis to nuclear power plant were initiated in Germany/Norway and in the USA. Whilst the projects were in progress the Three Mile Island accident occurred. The subsequent
investigation identified a clear fault diagnosis problem, which naturally gave new impetus to the work and focused more attention on safety aspects.

However, during the same period there was considerably less interest in computer based operator aids within the chemical process industry. An experimental alarm analysis system was developed for a high vacuum distillation column, at the Pernis refinery in Holland in 1967, but the work was apparently not followed up.

The situation has, however, changed quite dramatically in recent years, mainly because of the growth in artificial intelligence (AI) technology. The fault diagnosis problem has received much attention, and many expert and knowledge based systems have been developed for this purpose. Despite this, very few real time fault diagnosis systems are reported as being implemented on commercial chemical process plant, except for evaluation purposes. In many cases the limiting factor still appears to be the cost of modelling the process plant.

The necessity to improve the speed and efficiency of the modelling process was recognised by Andow and Lees in the early 1970's. Consequently much of their earlier work was directed towards the task of systematically generating data structures, suitable for alarm analysis and diagnosis. The research effort eventually resulted in the development of a formal methodology for the systematic generation of fault trees, which has been implemented in the FAULTFINDER suite of programs.

The currently described research follows on from the previous work of Andow and Lees. The project was initiated to investigate the application of computer-based methods of representing fault propagation in process plants, to the creation of data structure for alarm analysis using an expert system. Given the advanced state of the fault tree synthesis methodology and the availability of the FAULTFINDER programs, the fault tree method of representing fault propagation was selected as the basis of the approach.
The intention of the research work has therefore been to evaluate the suitability of the fault tree method of modelling fault propagation, for use in a real time alarm analysis/diagnosis system, and to investigate the problems involved with the automatic synthesis of this information. To this end, an experimental knowledge based system (KBS) has been developed, based on the previous work of Andow.

In order to develop and test the KBS, two small sections of plant have been considered in detail. The first system is a hypothetical example taken from the literature. This has been dynamically simulated. The second system is a pilot plant based at the BP Research Centre, Sunbury-on-Thames. In each case the processes have been modelled using the FAULTFINDER suite of programs and the resulting structures used as the basis of the diagnostic information.

This thesis describes the development of the experimental KBS and its application to two example sections of process plant. Chapter 2 is principally a review of the literature relating to tree based fault diagnosis methodologies. The great variety of alternative fault diagnosis strategies have only been briefly reviewed, because of their number and limited applicability to the current work.

Chapter 3 introduces the alarm diagnosis methodology in more detail. The choice of fault modelling technique is discussed in greater depth, followed by a description of how this information is represented within the KBS rulebase. The remainder of the chapter presents an overview of the fault diagnosis strategy.

Chapters 4 and 5 describe the inference mechanism of the KBS and the underlying theory involved. Chapter 4 is concerned with the combination of the individual alarm explanations and the simplification techniques employed. Chapter 5 concentrates on probabilistic assessment of the alarm cause explanations and the methods used to interpret the process variable indications.

The actual implementation of the KBS within the POPLOG language environment is discussed in Chapter 6.
Chapter 7 is concerned with the modelling of the two sections of process plant using the FAULTFINDER fault tree synthesis package. The application of the resulting knowledge bases to a number of fault scenarios is then described in Chapter 8.

Finally, the assessment of the suitability of systematically generated fault tree structures, for use in real time fault diagnosis, is presented in Chapter 9. The limitations of the KBS are reviewed in conjunction with the outstanding problems.
Chapter 2

LITERATURE SURVEY

This chapter contains a review of the publications relating to the diagnosis of process plant alarms. The literature survey has revealed a large number of papers pertaining to this subject area, therefore this review is principally concerned with those references relating to tree based fault diagnosis methodologies. However, for completeness a brief overview of some of the other fault diagnosis methodologies is included. References to publications dealing with the more general aspects of alarm analysis and fault diagnosis are denoted separately in the text.

In the middle of the 1960's, two Magnox nuclear power stations were being constructed for the Central Electricity Generating Board (CEGB), at Wylfa and Oldbury-on-Severn. In the initial specifications for both power stations the CEGB asked for alarm analysis to be incorporated into the computer monitoring systems. The papers by Welbourne [1,4], Kay [2], Kay and Heywood [3] and Paterson [5] discussed the development of the approach at the two sites.

Welbourne [1] first outlined the features of the data processing and control system that was to be installed at Wylfa, on completion of the construction phase. The paper briefly referred to alarm analysis, and described how cathode ray tubes (CRT's) were to be employed to display alarm information, but no details of the method of alarm analysis were included.
Just over a year later, in 1966, Kay [2] published a paper discussing on-line alarm analysis at the Oldbury-on-Severn nuclear power station. Kay argued that displaying the 3000 alarms in the central control room, using small inscribed glass facia panels, would make the task of alarm surveillance difficult for a single operator. Furthermore, because the plant was highly instrumented, and the alarmed variables interrelated to a large extent, one failure could cause numerous alarms to trigger in rapid succession. This would inevitably overload the operator with fault information. Given the high cost of erroneous actions at the Oldbury-on-Severn nuclear power station, Kay concluded that the existing alarm handling methods were inadequate.

To overcome at least some of these problems a computer based alarm analysis system was developed. The system was designed to analyse patterns of alarms, using knowledge of the cause and effect relationships between the alarms.

The objectives of the analysis were fourfold:

1. To determine the basic cause of the appearance of a group of related alarms;

2. To identify which alarm indications are giving unnecessary information so that they can be suppressed;

3. To select appropriate messages of advice or warning to be transmitted to the operator;

4. To infer from the appearance of monitored faults, those faults which are not monitored.

The resulting analysed alarms were to be displayed on four 21 inch CRT’s, one on each of the two reactor control desks and two on the supervisor’s desk.
At a conference later in the same year, Kay and Heywood [3] presented a more detailed account of the alarm analysis system being developed for the Oldbury nuclear power station. In the paper, the authors emphasised the importance of making a distinction between the alarm analysis control program software, and the plant specific data. This enabled the software development to be initiated early in the project, before the cause and effect relationships between the alarms were fully understood. Furthermore, the resulting software was plant independent and hence could be utilized in similar applications elsewhere.

Three techniques for analysing alarms were discussed, these being the probability method, the logical statement method and the tree analysis system. These three methods are outlined below:

1. The probability method is based on assigning a group of alarm inputs to an alarm message. Within each group the individual inputs are associated with a weighting factor. When the sum of the weighting factors of the active alarm inputs, exceeds the threshold value for that group, then the appropriate alarm message is sent.

   The authors stated that the technique was simple, flexible, and could cater for conditions that were not quite understood. However, they also noted that it was difficult to decide with confidence on the number of alarm groupings and values for the weighting factors. For these reasons, and because the method used large amounts of computer memory, the technique was not employed at Oldbury.

2. The logical statement method again involves assigning a group of alarm inputs to an alarm message. However, instead of employing arbitrary weighting factors, logical equations are used to decide if the alarm message should be sent. The example given in the paper illustrates the technique in more detail.
Consider the six binary state alarm inputs, A, B, C, D, E, and F, which are used to decide if the two alarm messages M1 and M2 should be displayed. The logical equations for each message are shown below:

IF (A or B) and D and not E \[ (2.1) \]
THEN display message M1

IF (A and B and D) or (F and not E) \[ (2.2) \]
THEN display message M2

Table 2.1 illustrates the possible outcomes, given certain combinations of active alarm inputs:

<table>
<thead>
<tr>
<th>Active Alarms</th>
<th>Message M1</th>
<th>Message M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>B, C, D</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>A, B, C, D</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>A, B, C, E</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>A, B, C, D, E, F</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

Kay and Heywood argued that it was easier to decide upon the logical relationships connecting alarm inputs to alarm messages, than it was to assign weighting factors. However, the method required more computer memory than the probability method, and the control program was more difficult to write.

The tree analysis system does not group together alarm inputs, but associates a message with each alarm signal. The relationships between the various alarms are then specified in terms of OR
logic. By using these simple building blocks, the complex interrelationships between the alarms can be represented using the minimum amount of computer memory. For example consider the alarm tree shown in Figure 2.1.

**Figure 2.1** Simple Alarm Tree

![Simple Alarm Tree Diagram]

The tree specifies that the fault indicated by the root alarm D will also cause alarms B and C, and eventually alarm A, to become active. The alternative root alarm E will also cause alarms C and A to become active, without affecting alarm B.

In this case the alarm analysis control program considers each new active alarm individually. The alarm tree information is searched to determine if any of the causes or consequences of the new alarm are active. If none of its causes are active, the new alarm is considered to be representing a new process fault, termed a prime cause alarm. Otherwise the alarm is simply treated as an additional consequence of another prime cause alarm.

The individual alarm messages are grouped together on the operator's display, based on the causal relationships that are identified by the alarm analysis system. However, in order to reduce the number of alarm messages that are displayed, only the prime cause of a group of alarms and the most recent consequences, are presented. The method of suppressing the intermediate alarms in a causal chain is termed 'alarm darkening'.

11
Kay and Heywood reported that the majority of the Oldbury alarms were handled using the tree analysis system. In a few cases the logical statement method was retained to deduce certain non-instrumented alarm conditions associated with process unit failures.

In November 1968 Welbourne [4] published a second paper, describing in more detail the alarm analysis approach used at the Wylfa nuclear power station. The three alarm analysis methods, previously discussed by Kay and Heywood, were also described, and the tree analysis system was again selected as the most suitable technique.

However, Welbourne considered the objectives of alarm analysis to be slightly different from those adopted at Oldbury. He suggested that the most important task was to identify whether fresh alarms were significant or not, rather than attempting to interpret a static jumble of alarms in order to find the most important alarm. Consequently a different method of displaying the alarm information was employed at Wylfa. Instead of grouping alarms on the basis of their causal relationships, a separate VDU was reserved to display all the prime cause alarms. All the remaining alarms were displayed on a second VDU, which could be cleared once the alarms had been accepted.

One month later Patterson [5] published an account of the alarm analysis system at Oldbury, which was by then in operational use. The paper described the system hardware and then went on to discuss the alarm trees in more detail, by referring to a number of realistic examples.

Paterson also noted that the Oldbury trees were draw-up by a committee, consisting of alarm-tree experts and the design engineers responsible for the section of plant being considered. A member of the station operational staff was also present at each meeting, and they were mainly responsible for the content of the alarm messages that contained operator instructions. Frequent meetings of the committee were required over a two year period, before the trees were considered acceptable to all concerned.
In the concluding remarks of the paper, the author stated that in the six months since the reactor had been operational, the alarm analysis system had performed fairly well. The alarm trees were considered to be basically satisfactory, although in some cases they were deemed to be over-complicated.

However, some ten years later, the assessment of the system was somewhat different. Long [6] reported that at a meeting in 1978, representatives from the CEGB had stated that the performance of the Oldbury-on-Severn and the Wylfa alarm analysis systems had been less than satisfactory. Specifically, the alarm trees had been costly to develop, subject to error and difficult to modify.

The first reported application of an alarm analysis system within the chemical process industry, was described by Barth and Maarleveld [7] in 1967. The problem of alarm analysis was considered as part of a research project into the operational aspects of a direct digital control (DDC) system, implemented at the Pernis refinery in Holland.

The objectives of the alarm analysis system were similar to those proposed for the Oldbury system, namely to sort the process alarms into groups of causally related disturbances. However, a different approach was adopted to tackle the problem, as outlined below:

1 The modelled plant is first split into small sections. These sections have to be selected on the basis that they are small enough to virtually exclude the risk of two independent mishaps occurring in the same section. On the other hand, they have to contain enough instruments to permit the detection of defective ones.

2 During the modelling phase the plant is examined section by section. For each section, the consequences of every considered fault are related to the alarms both within and external to the section. The time sequencing of the alarms is also noted.
When complete, the information is then rearranged so that for each alarm a classification tree type structure (termed a checklist) is generated, as shown in Figure 2.2.

When the causes of an alarm are being investigated, the relevant checklist is interrogated. If the new alarm cannot be related to another active alarm, then it is considered to be a prime cause alarm. Otherwise the alarm is appended to the list of consequences of another prime cause alarm. The alarm message derived from the checklist and the related alarms are then logged on a line printer.

Unlike approaches taken at Oldbury and Wylfa, the authors emphasised the importance of taking into account the time ordering of the alarms during the analysis. In addition, they also drew attention to the fact that the checks depended on the plant layout and location of the instruments. Since these would often change during the lifetime of a plant, and since the understanding of the existing plant usually improved with time, a simple method of updating the checklists was considered to be necessary.

Andow [8] commented that the technique was apparently successful for the experimental project, but it would be unsuitable for a large plant, due to the large number of interactions and correspondingly enormous volume of storage space required by the data. The work was apparently not followed up.

Following the pioneering work on alarm analysis systems, at the Oldbury and Wylfa nuclear power stations, two major studies into the application of disturbance analysis to nuclear power plant, were initiated in Europe and the USA.

The European project was a collaborative venture between the Institutt for Atomenergi (IFA) in Norway and the Gesellschaft fur Reactorsicherheit (GRS) at Garching, West Germany, together with the Kraftwerk Union and the Bayernwerk Utility Company.
The feasibility of the disturbance analysis technique was demonstrated through its application to the Halden Boiling Water Reactor in Norway. Dahll et al. [9,10] reported that the initial objectives of the disturbance analysis system (DAS) went beyond those of alarm analysis. The system was developed to analyse a disturbance, in order to find its causes, to determine the potential consequences of the fault and to suggest to the operator a corrective strategy. Following the success of the initial study, the disturbance analysis system, called Stvrunganalyserechner (STAR) in German, was applied to the Grafenrheinfeld nuclear power station, in Germany.

Felkel and Zapp [11] reported that a number of modelling strategies had been considered for use within the STAR system. A full unsteady state model of the plant was deemed to be the most desirable
in principle, but the practical difficulties of creating and solving such a model were considered to outweigh its usefulness. A tree or graph based modelling technique was therefore sought, resulting in the eventual selection of the cause consequence diagram (CCD) method.

Owre and Felkel \cite{12,13} discussed the plant modelling technique in more detail. An outline of the method is described below. The plant is first divided into subsystems such as the:

1. Electric Power Supply Subsystem;
2. Turbine Subsystem;
3. Generator Subsystem.

The next stage involves identifying the observable events (process faults) within each subsystem. The logical relationships between the events, within a subsystem and between subsystems are determined, and these are then described in terms of CCD's. An example of a CCD, taken from Reference 12, is shown overleaf in Figure 2.3.

The diagram illustrates the effect of a clogged oil filter on the lubrication and cooling system of a main condensate pump (MCP) in a pressurised water reactor. In addition to the usual cause and effect information, the tree also includes a number of messages, for each significant plant state that is reached.

The method of diagnosing the causes of a disturbance and predicting its future consequences involves a high degree of interaction between the process operators, and the STAR system. For example, consider the "High Pressure Drop Over Oil Filter" alarm, shown in Figure 2.3. If the alarm occurs, then the DAS facility searches its library of CCD's to find a tree which includes the active alarm state. In this case the CCD shown in Figure 2.3 will be selected. The message associated with the active alarm state will then be sent to the operator's VDU console. If the operator requests a diagnosis of the alarm, then the prime cause fault "Clogged Oil Filter" will be displayed.
If no corrective action is taken and the "High Temperature Reading Downstream Of Oil Cooler" alarm is triggered, then the operator is sent a further message of warning. The propagation of the disturbance in the pump cooling and lubrication system is tracked with the remainder of the CCD. At each stage, if the preceding alarm becomes active, then the relevant operator message is sent.
A colour graphic VDU was utilized to present the results of the disturbance analyses to the operators. In addition to the textual warning and advice messages, the operators could also recall the CCD's. Different colours were used to differentiate between active and inactive branches.

Felkel at al. [14] acknowledged that the most difficult and costly part of developing the STAR disturbance analysis system, was the generation of the CCD's. They attributed this to three main problems:

1. The CCD's cannot readily be used as a computer database;
2. The construction of a CCD requires a thorough knowledge of the process;
3. The analysts, who have the detailed knowledge of the process, are not computer specialists in most cases.

In an attempt to overcome this limitation, the MOGEN computer language and compiler was developed. This enabled the CCD's to be described in engineering terms by the system analyst. The resulting source code could then be compiled into a suitable format for the STAR database.

The desirability of an automatic CCD synthesis package was recognised by Felkel and Zapp [11]. They argued that it would reduce the probability of human error and oversight. However, such a system was apparently not developed.

Unfortunately, the literature published in English does not appear to include any assessments of the eventual performance of the STAR system.

The second major investigation of disturbance analysis systems was conducted by the Electric Power Research Institute (EPRI) in the USA. Accounts of the project were reported by Frogner and Meijer [15,16] and Long [6,17].
The development of a disturbance analysis facility was initially motivated by a desire to improve the availability of nuclear power plant, through the early diagnosis of disturbances that could lead to a plant shutdown. However, during the development of the project, the Three Mile Island (TMI) accident occurred. As Long suggested, if the DAS facility had been present at the TMI nuclear power station, the accident might have been averted. As a consequence, the potential safety benefits of the method were recognised and the DAS system was extended to provide certain safety-related surveillance functions.

The initial objectives of the EPRI-DAS facility were broadly similar to those of the STAR system. More specifically, the DAS was designed to assist the operator in the real-time assessment of plant disturbances by performing the following tasks:

1. Determine cause, plant status and corrective action;

2. Recognise the significance of events and alarms with respect to the current plant conditions and preceding events;

3. Recognise the significance of events and alarms with respect to the plant mode of operation;

4. Enhance the information content of the alarms during disturbances and reduce the number of secondary and extraneous alarms;

5. Predict the future propagation of disturbances;

6. After a plant trip, contribute to the analysis of the events that preceded the trip.

Although the two systems shared many of the same objectives, the methods used to achieve them in each case, were significantly different. In addition, the EPRI-DAS facility was not designed to be as interactive as the STAR system and less use was made of probabilistic information. It was felt that the value of such features was marginal relative to the effort needed to include them.
The EPRI-DAS facility utilised the following three level methodology for analysing the disturbances:

1. The first level is based on lookup tables, relating sensor signals to operator messages. When a disturbance is detected, the lookup tables are scanned for a match between the currently active process alarms and an operator message. If a match is found, then the corresponding message is displayed at the operator's console.

   Frogner and Meijer [15] argue that the simple lookup table approach is an efficient and simple means of analysis, for a significant fraction of the commonly occurring plant disturbances. Furthermore, the analysis module functions as an information filter, which can prevent a large number of the disturbances from being subjected to the considerably more time consuming and sophisticated analysis schemes, of the higher levels.

2. The second level of analysis is used to analyse disturbances that are characterised by complex logical relationships and/or sequences of events. The approach appears to be quite similar to that used within the STAR disturbance analysis system. Cause consequence diagrams are used to track the propagation of the disturbance, and messages are again sent to the operator when the relevant branches of the tree become active.

   However, more emphasis is placed on the time ordering of the process disturbances. For example, consider the section of CCD shown overleaf in Figure 2.4.

   The diagram specifies that in this causal path, disturbances A and B occur with a minimum time delay of 10 seconds between them. If in reality the two disturbances are observed to occur within a 10 second time interval, then the CCD shown in Figure 2.4 is rejected as an explanation of the fault propagation.
Figure 2.4 Time Delay Cause Consequence Diagram

Disturbance A
/|\
| Time Delay
of 10 s
| |
Disturbance B
/|\
|

3 The third level of analysis is used when the disturbances being investigated cannot be adequately described in terms of CCD's. In this case, quantitative engineering type calculations are used by the DAS system.

Despite the development of a three tier methodology, the majority of the research effort appeared to be concentrated on the CCD technique. In similarity with the STAR system, the CCD's were again manually constructed. However, the method of processing them into a suitable form for the computer database, was not as advanced as the MOGEN language and compiler. Lees [18] also comments that although the EPRI-DAS cause consequence diagrams were used to predict future events, they resembled fault trees. This was because they converged to a single top event, rather than diverging to a set of outcomes.

The EPRI-DAS facility was developed and tested through its application to two nuclear power plant subsystems. The subsystems were selected using data of their outages, from existing nuclear power stations. The selection criteria were that the subsystem has caused significant operational problems and plant downtime, that these be recoverable by operator action, that the system be representative, that it be adequately simulated on the available simulators and that
application of disturbance analysis to the system constitute a suitable test for the methodology. As a consequence, the Feed Water and the Component Cooling Water Subsystems were eventually selected.

In the final report, describing the experimental DAS system, the results of the performance evaluation were presented. Frogner and Meijer [16] concluded that the DAS provided the operator with an earlier and more precise diagnosis of disturbances than the existing alarms and instrumentation of the plant simulator. As a result, it was observed that the operator was consistently able to take the proper corrective action, with less challenge to the protective system.

Research into alarm systems started within the Department of Chemical Engineering at Loughborough University in the early seventies. Andow [8] and later Andow and Lees [19], investigated the application of alarm analysis to process surveillance and control.

The earlier work of Andow discussed the broad range of problems associated with alarm analysis. However, the two areas that were investigated in the greatest detail were those of alarm system definition and the systematic generation of alarm data structure. Andow argued that the effort required to set up an alarm analysis scheme from an intuitive model of a plant was excessive. It was therefore reasonable to assume that alarm analysis schemes would only be used when an alternative method requiring considerably less effort was used.

In order to address the problem of alarm system definition, Andow developed a suite of programs which enabled the modelled system to be dynamically simulated. The process model was represented in terms of a configuration of units models, linked by their common process variables. The interrelationships between the process variables, within each unit model, were then defined using unsteady state mass and energy balances, and equilibrium and rate relationships. By simulating the propagation of process disturbances through the plant, the magnitudes of the dependencies between the process variables were determined, thus enabling sensible alarm limits to be derived.
The method of systematically creating alarm data structure was based on a similar technique. The system was again represented as a configuration of unit models, connected in accordance with the flow diagram. However, the behaviour of the unit models could be specified using either full differential equations or signed functional equations.

For example, a unit model for a simple mixing tank is shown in Figure 2.5. The tank unit has eight process variables associated with it, namely the pressure, flowrate and temperature at the inlet and outlet, and the level and pressure within the closed tank.

Figure 2.5 The Closed Tank Model

From the network of process models, a network of process variables was constructed. This was then simplified to eliminate the unobservable variables. Finally, the reduced process model was interrogated to determine the causal relationships between each monitored variable, and this information was then used as the basis of the alarm data structure.

The section of process variable alarm tree that would be derived from the unit model specified in Figure 2.5, is shown in Figure 2.6. The arrows indicate the direction of search from the undesired alarm
to its prime causes, and the signs associated with each edge denote the sign of the gain. For example, the process variable alarm tree specifies that a high discharge pressure ($P_c$) can cause a low discharge flow ($F_c$), a high tank level ($L_b$) and a high inlet pressure ($P_a$).

2.6 The Process Variable Alarm Tree

A linked list structure was found by Andow to be the most convenient method of storing the alarm trees within a computer. The lists related a given alarm to its immediate cause alarms, which were then related to their immediate cause alarms, until the roots of the alarm trees were reached.

In the paper by Andow and Lees [19] the authors stated that they had subsequently abandoned using differential equations to model the interrelationships between the process variables, in favour of signed functional equations. They argued that for the purpose of identifying causal relationships, functional equations were almost as good as full differential equations, yet they required much less time and effort to derive.
The alarm trees were principally derived to represent the propagation of disturbances between the process variables. Deduced alarms, similar to those employed at Wylfa and Oldbury, were to be used to identify mechanical faults. Andow and Lees briefly discussed the synthesis of the deduced alarm information, but they acknowledged that this aspect of the work was not fully developed. The authors also noted that no use had been made of probabilistic information, but that it was likely that this would be introduced as the treatment of deduced alarms was developed.

The real-time aspects of analysing process alarms were later discussed by Andow [20]. The method employed alarm trees, described previously, to sort the process alarms into groups of causally related symptoms, and to identify the prime cause alarms. Deduced alarms were also used to resolve mechanical faults.

Probabilistic information was not explicitly used in the analysis procedure. However, the causes of each alarm were stored in a list structure, in the order of their probability of occurrence. In this way the higher probability causes of an alarm were considered first, hence maximising the efficiency of the searching strategy.

The number of resolvable mechanical failures was limited to the number of unique patterns and sequences of alarms. Where faults exhibited the same symptoms, the time interval between the alarms and their chronological order was considered. However, Andow noted that the method of deducing process unit mechanical failures, implicitly assumed that the pattern of symptoms was caused by a single fault.

The experimental alarm analysis scheme was implemented on a PDP 11/20 computer using the RTL/2 real-time language. The language was selected on the basis that it was efficient in terms of execution speed, the code was compact, its error handling facilities were well developed and it was flexible enough to interface to the required operating system routines. In addition the list processing facilities could be implemented relatively easily.
Soon afterwards, an alternative method of analysing process alarms, based on a fault tree approach, was reported by Martin-Solis, Andow and Lees [21]. The technique essentially involved constructing fault trees in real-time to identify the causes of an alarm, in terms of other monitored process variable deviations and failures of the process equipment.

The plant was again described in terms of a configuration of process unit models, connected by their common variables in accordance with the flow diagram. The unit models were also based on functional equations, however, they were written in the form of mini fault trees. An illustrative example is given below:

The functional equations relating the pressure and flow through a pipe are as follows:

\[
\begin{align*}
\frac{d P_{in}}{dt} &= f(Q_{in} , Q_{out}) \\
Q_{out} &= f(P_{in} , P_{out})
\end{align*}
\]  

(2.3)  

For example, Equation 2.4 specifies that the pipe outlet flowrate is proportional to the inlet pressure and inversely proportional to the outlet pressure.

The mini fault trees are derived by considering in turn, the deviation states of the variables on the left hand side of the functional equations. By using the signs of proportionality, the corresponding deviation states of the right hand side variables can be determined. The mini fault tree for a low flow deviation out of a pipe is shown in Figure 2.7.
Additional information relating process unit faults to variable deviations can also be specified within the models. In the given example, a partial blockage of the pipe or a leakage of the unit is expected to cause a low discharge flow. For each process variable deviation that is considered at the outlet port of a model, a corresponding mini fault tree is therefore required to specify its causes.

The process alarm fault trees were synthesised by selecting a top event, usually specified in terms of a process variable deviation at a given point in the system configuration, and developing each branch of the tree using appropriate mini fault trees. In addition to simply joining the individual mini fault trees together, the synthesis algorithm also performed a number of consistency checks to prevent the development of erroneous branches.

The fault tree synthesis algorithm was also employed to generate fault trees off-line, for design purposes. The major difference between the alarm fault trees and their design counterparts, was the extent to which the fault trees were developed. The design trees were synthesised to include all the possible causes of a given top event.
However, the alarm fault trees only included those branches which were consistent with the observed state of the plant, when the top event alarm was active.

The authors discussed in some detail the outstanding problems of applying the fault tree synthesis technique to alarm analysis. They commented that the usefulness of the method was a strong function of the level of plant instrumentation. In the case of a highly monitored process the alarm fault trees would be small, because many of the alternative failure paths would be eliminated. Conversely, on a poorly instrumented plant there would be little difference between the design and alarm fault trees, and hence the benefits of constructing the fault trees in real-time would be minimal.

The weakness of the fault tree technique for representing the time ordering of events was also noted.

The development of the fault tree synthesis methodology has been further described by Shafagi, Andow and Lees [22], and more recently by Kelly and Lees [23] and Mullhi, Ang, Lees and Andrews [24]. However, the later references are principally concerned with the synthesis of fault trees for design applications.

In 1985, Andow [25] discussed the application of intelligent knowledge based systems (IKBS) to the problem of fault diagnosis. The author argued that in general there was a good match between the requirements of the diagnostic task and the characteristics of IKBS, such as pattern recognition, problem solving and rule based knowledge representation. However, the application of such techniques to real-time process plant fault diagnosis was not straightforward, for the reasons discussed below:

1 Not all plant states are observable;

2 Instrument failures are common and often cause confusion;

3 The analysis carried out must allow for the dynamic behaviour of the plant - the rules are not static;
The monitoring and display systems used by the operator may be difficult to use for fault diagnosis - although they may be well-suited to normal operation.

An experimental IKBS was therefore developed by Andow to explore the problems in greater depth. The software was written in the PROLOG artificial intelligence language.

The method of representing the initiation and propagation of process disturbances was similar to that previously described by Martin-Solis et al. [21]. However, the techniques used to piece together the mini fault trees were significantly different in the following respects:

1. The causes of the investigated fault could be developed to include process variable deviation states, at variance with the observed state of the plant. The discrepancy was explained in terms of an instrument failure. As a consequence the likelihood of the explanation was penalised to take account of the likelihood of the instrument being in a failed state.

2. The method of searching for the causes of a process disturbance was based on a 'best first' approach. Andow suggested that the following strategy was a suitable candidate for real-time use:

   I. Carry out a breadth-first search along all causal paths until either the path terminates or a measured variable is encountered.

   II. All the solutions found in step 1 are compared and only the 'best ones' retained in an ordered list.

   III. The best solutions are then searched to successively greater depth (using the pruning mechanism described in step 2) at each level of search.
The 'best' solutions were those which matched the observed state of the plant, with the least number of instrument failures having to be taken into account.

When the causal search was complete, the operator was presented with a ranked list of explanations for the causes of the investigated disturbance. A facility was provided to allow the operator to volunteer additional information, not available to the diagnosis system. In addition, the user could reject solutions offered by the IKBS and focus on others. The intention was to help avoid the 'mind set' problem, yet still have the operator in overall control.

An alternative method of representing the initiation and propagation of process disturbances is the directed graph (digraph) model. This technique has been used, both for the synthesis of fault trees, as described by Lapp and Powers [26], and for fault diagnosis, as reported by Iri et al. [27] and Kramer and Palowitch [28].

A digraph is a network of nodes and edges, representing the pathways of causality in the modelled system. The nodes describe state variables, alarm conditions or failure origins, whilst the edges represent the direction of causal influences between the nodes. A refinement of the method is the signed digraph (SDG), in which the signs associated with each edge specify whether the cause and effect variables tend to change in the same or opposite directions.

Iri et al. [27] proposed an algorithm for the diagnosis of chemical process system failures, based on the SDG model. The method first involved mapping the observed symptoms of the fault, onto the pre-prepared digraph of the modelled system. The high, low and normal monitored variable states were denoted using the '+', '-' and '0' symbols respectively. However, controlled variables which had been maintained in their normal states, through the action of their control loops, were treated as special cases.

The second stage of the procedure involved reducing the general system digraph, to a cause and effect digraph explaining the pattern of observed symptoms. Two main criterion were used to reject possible
causal pathways. Firstly, those edges which converged on or diverged from a steady state process variable node, were deleted. The notable exception in this case being controlled variable nodes. Secondly, if the sign of an edge was inconsistent with the states of its cause and effect nodes, then it also was deleted. The remaining valid edges were then used to relate the observed pattern of symptoms to the root causes of the fault.

The technique had three main limitations:

1. The digraph reduction process was time consuming and hence was unsuitable for real-time use;

2. The method assumed that a single fault was the source of all the disturbances;

3. The possibility of indication failures was not taken into consideration.

The problems of interrogating digraph structures in real-time were recognised by Kramer and Palowitch [28]. Consequently, the authors proposed a method of synthesising boolean rules from the signed digraph information, which could then be used to diagnose process faults in real-time. A simplified account of the synthesis procedure is given below.

The diagnosis technique again assumes that a pattern of observed symptoms is caused by a single process unit failure. The consequences of each considered fault are first developed through the SDG, until all the monitored variable nodes have been reached. The resulting consequence diagram is then simplified to eliminate the unobserved process variable nodes.

Each consequence diagram therefore details how the monitored process variables will be affected by the original fault, and the order in which the variables will deviate. This information is then specified in the boolean rules using logical equalities. For example, consider the simplified consequence diagram shown in Figure 2.8.
The diagram specifies that fault A causes a positive deviation of variable B, which then causes a positive deviation of variable C, which in turn causes a negative deviation in variable D.

The relationship between variables B and C can be defined using the logical Equation 2.5

\[ p(BC) \iff (B = C) \text{ or } (|B| > |C|) \]  

Equation 2.5

The equality \( p(BC) \) will therefore be true, if the state of variable B is the same as variable C or if variable C is less deviated than variable B. The later inequality is to take account of the fault propagation time delay. Similarly, the logical equality defined in Equation 2.6 can be used to test if the disturbance has propagated from variable C to D.

\[ p(CD) \iff (C = -D) \text{ or } (|C| > |D|) \]  

Equation 2.6

The resulting boolean rules therefore specify the expected pattern of symptoms that will be generated by each considered fault. For the above example, the following rule would identify fault A:

If
\[ p(BC) \]
and
\[ p(CD) \]
Then
fault A
In real-time the causes of a pattern of symptoms are determined by searching through a rulebase of similar (and in reality much more complex) boolean rules, until a match is found. The authors also comment that because of the general If-Then format of the rules, the knowledge generated from the digraph can be easily integrated with information obtained from other sources.

Two different fault diagnosis methodologies, based on more fundamental representations of the process behaviour, have been reported by Dhurjati, Lamb and Chester [29,30] and Herbert and Williams [31].

Dhurjati et al. described the development of the FALCON (Fault AnaLyser CONsultant) expert system, and its application to a commercial scale chemical process plant. The authors stated that they had first considered a causal modelling approach to the fault diagnosis problem, as reported by Chester et al. [30]. However, they subsequently found that this abstraction of the physical behaviour, was inadequate to describe the complex dynamic interdependencies between the process variables in the modelled system. A mixture of both quantitative and qualitative information was therefore used in the FALCON system.

The quantitative information was derived from the following sources:

1. Material balance equations (written in terms of measured variables) around defined control volumes;
2. Energy balance equations (also written in terms of measured variables) on various control volumes;
3. Empirical equations relating measured variables;
4. Equations for PI controllers;
Valve curve correlations relating flow measurements to controller outputs;

Equations for calculation of heat transfer coefficients.

These equations were then simplified, and their underlying assumptions examined, to determine if they could be used to detect process unit failures. For example, the mass balance equation would be expected to hold true, within a certain error margin, if there was a continuity of flow within the control volume. However, a pipe leakage would violate the continuity assumption and hence would be detectable from the mass balance equation. For those equations which were affected by process unit failures, a propositional link was defined between the validity of the equation and the existence of the fault. The approach therefore has many similarities with data reconciliation techniques.

The qualitative knowledge was mainly in the form of shortcut heuristics, used by the process operators in the diagnostic task. The knowledge was elicited by simulating a number of faults, using a mathematical model of the process, and plotting the values of the observable variables, as functions of time. This information was then presented to the operators, who were encouraged to "think out loud".

The inference strategy involved using the observed state of the process to determine the validity of the quantitative equations. The propositions associated with each piece of quantitative information, in conjunction with the heuristic knowledge, were then used in a predominantly forward chaining process, to identify the root cause of the process disturbance. The eventual conclusions were finally presented to the operator using a colour graphic, touch sensitive screen.

The initial evaluation of the FALCON system was performed by introducing faults into a dynamic simulation of the process. Of the 100 faults that were tested, the FALCON software managed to identify them all. The final evaluation of the system was conducted using actual plant data, and of the 10 faults that occurred during the trial period, the FALCON system again managed to correctly identify them.
The FALCON system was therefore very adept at diagnosing alarms on the modelled process. However, the project cost was in the order of 1 million US dollars, and no indication of the system modelling effort was given.

Herbert and Williams [31] described an experimental fault diagnosis system, which had been developed to assess the ability of a technique called Incremental Qualitative Analysis (IQA), to discriminate between faults on nuclear power plant.

The IQA method uses sign algebra and relates variable changes with time (trends) to deviations in variables from their steady state values. The behaviour of a modelled system is described using 'confluences' which are effectively qualitative differential equations.

The authors noted that the technique had been previously applied to the problem of diagnosing faults in electronic computer hardware. However, the earlier work had not considered the problems of feedback, two way fault propagation and sensor failures, therefore these features had been developed as part of the reported work.

An implementation of IQA, in PROLOG, was used to model the behaviour of a pressurised water reactor (PWR) pressuriser, and its associated control systems. The confluences described the thermodynamic behaviour of the fluid in the pressuriser vessel and the correct behaviour of the sensors, control systems and relief valves. This required the definition of 161 confluences, 134 logical connectives and 72 qualitative variables.

The process variable information was simulated using a quantitative dynamic, non-equilibrium reference computer model of the PWR pressuriser. An input processor, written in FORTRAN 77, captured data from this simulation and calculated the variable trend values. In addition the program also discretized the continuous process variable values into the three states of high, normal and low.
The IQA model of the system was continually updated with qualitative trend and variable values from the input processor. The model was then solved in real-time to determine if all of the constraints, describing the correct behaviour of the system, could be satisfied simultaneously. If all the constraints could be satisfied, then the process was fault free, otherwise at least one fault was assumed to be present. When a fault was detected, the diagnosis procedure involved relaxing some of the constraints until the remaining constraints could be satisfied simultaneously.

In theory the diagnosis technique could identify any number of simultaneous faults, but there was a heavy time penalty if more than one fault was considered. For example, the authors quoted that the experimental system took approximately 10 seconds to detect a fault and five minutes to diagnose a single fault. However, if two faults were considered, then the system took about one hour to search for the faults.

The strengths and weaknesses of qualitative simulation and automatic fault tree synthesis techniques have been discussed by Waters and Ponton [32]. The authors were principally concerned with the problem of simulating the response of process plant to input disturbances, as a means of providing computer support for the HAZOP task. However, many of their comments are also relevant to the field of fault diagnosis.

Waters and Ponton noted that in general terms the bottom-up (qualitative simulation) and top-down (fault tree synthesis) approaches were broadly similar in the following respects:

1. There is a need to reason from first principles;
2. It is sometimes necessary to use numbers or estimates of scale;
3. Heuristics are required to deal with sequential or time dependent behaviour in a reliable manner.
However, they also highlighted some striking differences due to the relative speeds and efficiencies of each technique. The authors stated that a top down method only follows the pathways that lead to the top event of interest, whereas a qualitative simulation explores all behaviours that are consistent with the initial state of the system and the input deviations. As a result, qualitative simulation is potentially more combinatorial than fault tree analysis because many of these explored behaviours are "uninteresting" from a safety point of view.

Conversely, a bottom-up simulation uses a clearer notion of system state than the top-down approach, which means there is more information available to resolve ambiguities. As a consequence, the authors suggested that the inferencing power of a bottom-up approach is much greater than the top-down one, since it is possible to introduce extra information to "prove" the connection between the initiating events and the hazard (or the alarm in the case of alarm diagnosis).

The remainder of the paper described how the authors applied and extended the qualitative simulation method, used by De Kleer and Brown [33], to model a set of pre-defined flowsheet situations, consisting of the following components:

1. Relief valves;
2. Resistance elements;
3. Binary junctions;
4. Binary splits;
5. Total blockage points.

They reported that the enhanced qualitative simulation technique could solve all but one of the flowsheet configurations. Unfortunately, they also suggested that the cost of the improvements (in terms of computational efficiency) were inordinately high. In addition, some
serious problems were identified relating to the solution procedure. These were felt to severely limit the applicability of qualitative simulation tools in their current form.

Rich and Venkatasubramanian [34] discussed the trade-off between the speed of interrogating compiled knowledge, such as that derived by Kramer and Palowitch [28], and the reliability of more fundamental models of process causality. They suggested that an expert system which inferred conclusions based solely on compiled knowledge would suffer from two main disadvantages:

1. The diagnosis system would have no means of verifying the intermediate causal relationships skipped by the (compiled) heuristics;

2. The expert system would tend to be "brittle", because it would fail abruptly when asked to locate system malfunctions for which no available heuristic applied.

The authors further argued that these problems could be overcome by relying upon more fundamental models of system behaviour, comprised of device models and connectivity information. The major disadvantage of the approach was identified as the time and effort required to methodically chain through the device models, whilst attempting to relate the process symptoms to the origins of the fault.

Given the strengths and weaknesses of the two general fault diagnosis methodologies, the authors proposed a hybrid strategy based on a two tier knowledge base, consisting of heuristics supported by a lower tier of deep-level knowledge. The approach aimed to improve the diagnostic efficiency by performing the bulk of any diagnosis using the compiled knowledge. Any gaps in the compiled knowledge could be overcome by reasoning with the deeper knowledge, and then updating the heuristic knowledge base for future use. A brief synopsis of the methodology is outlined overleaf:
The first stage in the diagnostic procedure involves interrogating the compiled heuristics, in the form of production rules, until a match is found between the observed symptoms and a process fault. The result is then displayed to the operator. If the user requires additional causal information relating to the diagnosis, the expert system can then interrogate the deeper-level knowledge base. Similarly, if the diagnosis system fails to determine a cause for the pattern of symptoms from the heuristic knowledge, or if the diagnosis proves to be incorrect, then the expert system is forced to reason from the deep-level model of system behaviour.

Following the diagnosis, the expert system updates the heuristic knowledge base if it failed to correctly identify the process fault. This is achieved by either adding a new rule or modifying the existing rules through rule specialisation or rule generalisation. A heuristic is specialised by including additional premises to the existing heuristic in order to limit its applicability. For rule generalisation, heuristics are combined to reduce redundancy in the knowledge base.

In the concluding remarks of the paper, the authors suggested that for each new process application, the deep-level knowledge base could be assembled from an existing library of device models. The diagnosis system could then be driven by a simulator that simulated the process response to a number of common fault conditions. In response the diagnosis system would recognise the important patterns and compile them into heuristics.

An example of a frame based fault diagnosis expert system has been described by Paterson, Sachs and Turner [35,36] and later by Paterson and Sachs [37]. The ESCORT (Expert System for Complex Operations in Real-Time) system was designed to help users of information systems which generate large volumes of dynamic data. In the papers, the authors discussed its application to the problem of process plant alarm analysis/fault diagnosis.
The architecture of the ESCORT system is shown in Figure 2.9. The system firstly recognises events in the process plant which may indicate a problem. These primary events are then crudely prioritised so that they can be diagnosed in a sensible order.

Figure 2.9 The Architecture Of The ESCORT System

The main diagnosis system attempts to relate events to operator errors and/or instrument failures. Process unit failures such as pipe blockages or leakages are not considered because of their rarity. The resulting root cause problems are then prioritised and presented to the process operator.

The definition of the process plant is held in a "class inheritance lattice" which is essentially a frame-based representation with class/subclass links, and an inheritance of attribute values from higher classes. For example, a pressure control loop would be represented as a subclass of the more general "plant items" class. The "pressure control loop" class would then have defined a number of subclasses including "pressure sensor", "control loop" and "control valve". Within each frame the attributes represent the connectivity of the process units, and rulesets to determine if a particular assertion concerning the component is valid and its possible causes.
The diagnosis system works by maintaining a dynamic hypothesis network, which represents the system's understanding of the state of the process at that point in time. The hypotheses define the truth value of some assertion, concerning the state of a process variable or unit, for a given time period. For each new event that is recognised as important, the ESCORT system defines a new hypothesis to describe the fault condition.

When time permits, the causal knowledge stored in the class inheritance lattice is used to relate the event to its possible causes and consequences in the hypothesis network. When the event is no longer observed to be true, the inference mechanism attempts to delete the hypothesis from the network, and any other related cause hypotheses.

The authors noted that probabilistic information was not used in the ESCORT system at the time of reporting, but that it was being considered as a future development.

Escort has been applied to the problem of diagnosing alarms on a commercial scale process plant at the BP Grangemouth site. However, no independent assessments of the systems performance have been published as yet.
Chapter 3

THE ALARM DIAGNOSIS METHODOLOGY

The purpose of this chapter is to introduce the alarm diagnosis methodology that has been used within the development KBS. Because the fault propagation modelling technique is the foundation of the methodology, the first section of the chapter is devoted to the topic. The following section describes how this information is represented within the rulebase.

Finally, Section 3.3 provides an overview of the fault diagnosis strategy and a brief outline of the techniques involved. These are then discussed in more detail in Chapters 4, 5 and 6.

3.1 The Fault Modelling Technique

The alarm diagnosis methodology described in this thesis is based on the fault tree method of representing the initiation and propagation of faults within process plant. This fault modelling technique was chosen for three main reasons:

1. The fault tree methodology is an established and widely used tool for the detailed and quantitative assessment of risk. As such, the formal theory relating to the manipulation and evaluation of the structures is well documented.
The fault tree technique is well suited to describing the causes of process alarms, as discussed by Andow [25], Lees [38] and Powers and Tompkins [39]. The fault tree structure enables any number of causes to be related to a single top event, in this application a process alarm. Furthermore, the trees are not simply restricted to OR logic, therefore more complex fault scenarios can be represented.

One of the most time consuming tasks in the application of any fault diagnosis methodology, is the modelling of the process plant. As described in the previous chapter, the techniques used to derive the fault models vary considerably. Some methodologies use well defined procedures to synthesise the fault models from the process equations, whereas others rely on less systematic approaches, such as interviewing the process experts.

The usefulness of any fault diagnosis system depends upon its credibility. If the system frequently provides incorrect diagnoses, or fails to identify the most obvious causes of an alarm, then the process personnel will fail to use it. A rigorous, and if possible, an automated method of generating the fault propagation models is therefore desirable, if the speed and reliability of the modelling process is to be improved.

In recent years there has been a considerable interest in the automatic synthesis of fault propagation models, for use in alarm and disturbance analysis systems. Many different fault tree synthesis methodologies have been developed as reported by Fussel [40], Powers and Lapp [41], Apostolakis et al. [42] and Taylor [43].

Within the Plant Engineering Group at Loughborough University of Technology (LUT), there has been considerable research activity in the area of systematically creating alarm structures and fault trees. This has been reported by Andow [8], Andow and Lees [19] and Martin-Solis, Andow and Lees [21]. One of the results of this research work has been the development of the FAULTFINDER fault tree synthesis package. Although the FAULTFINDER programs have been used to model and generate fault trees for process plant, as reported by Kelly and Lees [23] and
Mullhi, Ang, Lees and Andrews [24], the system had not been applied in the area of alarm diagnosis. Therefore, one of the aims of this research project has been to assess the suitability of the FAULTFINDER synthesis package, and its resulting trees, for use within an alarm diagnosis system.

3.2 Representing The Fault Trees Within The KBS

As discussed earlier, the fault tree is an efficient and concise method of describing the causes of a process alarm. However, they can often contain considerable amounts of information. For example, if the causes of one alarm are developed through a commercial scale plant to its boundaries, the resulting fault tree will usually be very large and detailed. Given that one fault tree would be required for each alarm, the quantity of diagnostic information would be excessive. In addition to difficulties of simply storing and manipulating such information in real-time, the knowledge representation technique would also be extremely inefficient, because of the quantity of duplicated information.

To overcome these limitations, a representation similar to that described by Andow [25] is used within the development KBS. Instead of developing the causes of each alarm to the process boundaries (as is the case with design type fault trees) the causes are now only traced to the nearest alarmed process variable deviations, except in a few special cases. For example, consider the buffer tank system shown in Figure 3.1.

Fluid from an upstream section of plant, flows through pipe 1, control valve cv_1, flow sensor fs_1 and pipe 2 into the header tank. The liquid is then removed via pipe 3 and flow sensor fs_2, to the downstream section of plant. The tank level is monitored by level sensor ls_1, and the signal output is used by controller cnt_1 to manipulate the inlet flowrate. The three measured variables, flow f1, level 11 and flow f2, are all associated with high and low alarm limits.
The fault trees for the level alarms are now only developed to include equipment failures in the plant bounded by the two flow sensors, and deviations in the two flow variables, \( f_1 \) and \( f_2 \). This contrasts with a design type fault tree where the top event is predominantly developed to basic failures, rather than undeveloped causes. For example, Figure 3.2 illustrates the reduced fault tree for the high level alarm.

Figure 3.1 The Buffer Tank System
Figure 3.2 The Reduced Alarm Fault Tree

HIGH LEVEL L1 ALARM

OR

FLOW F1 HIGH
LEVEL SENSOR LS_1 FAILS HIGH
CONTROLLER CNT_1 FAILS HIGH
CONTROL VALVE CV_1 FAILS OPEN

OR

SET POINT IS HIGH

AND

CONTROLLABLE FAULTS

OR

CONTROL VALVE CV_1 STUCK NORMAL
CONTROLLER CNT_1 STUCK NORMAL
FLOW F2 LOW
FLOW SENSOR FS_2 BLOCKED
PIPE 3 BLOCKED
There are two points of special interest within the tree:

1. Because the control loop has the ability to compensate for a downstream blockage, these faults are combined with the causes of the control loop failing stuck in its normal state. The upstream flowrate deviation is also controllable, however, the control loop failure modes in this case are included in the high flow alarm fault tree.

2. In the high level alarm fault tree, the causes of the control loop failing stuck normal do not include a stuck normal failure of the level sensor. Similarly, a low level sensor failure, which would cause an actual high level in the tank through the action of the controller, is not considered. This is because if a process fault is the cause of the alarm, then the level sensor must be working. Conversely, if the level sensor had failed low, it would not trigger a high level alarm.

Whilst an alarm fault tree is a concise structure, the relationships between the primary equipment failures and the top event (the alarm) are not always clear, especially when there are AND gates in the intervening logic. Furthermore, if the tree structure were to be represented in a suitable format for a computer, the relationships could become even more blurred.

In order to overcome these limitations, a very simple knowledge representation method was chosen to describe the alarm causes, namely If-Then rules. As the name suggests this structure is composed of two parts, a cause or conditional part and a consequence or effect part.

To further simplify the structuring of the knowledge, the alarm fault trees are decomposed into minimum cutsets before being cast in the form of If-Then rules. Each minimum cutset describes one method by which the top event can occur, in terms of the minimum number of faults required.
The conversion from an AND/OR tree structure to a minimum cutset form can be performed manually, or using a cutset evaluation code such as FTAP [44] or PREP [45]. The resulting minimum cutsets then form the rule antecedents.

For example, the minimum cutsets for the high level alarm fault tree, shown in Figure 3.2, are listed below:

1. Flow f1 high;
2. Level sensor ls_l fails high;
3. Controller cnt_l fails high;
4. Control valve cv_l fails open;
5. Set point high;
6. Flow f2 low and control valve cv_l fails normal;
7. Flow f2 low and controller cnt_l fails normal;
8. Pipe 3 fails blocked and control valve cv_l fails normal;
9. Pipe 3 fails blocked and controller cnt_l fails normal;
10. Flow sensor fs_2 fails blocked and control valve cv_l fails normal;
11. Flow sensor fs_2 fails blocked and controller cnt_l fails normal.

3.2.1 The Structure Of The If-Then Rules Within PROLOG

As will be discussed in Chapter 6, the diagnostic component of the KBS is written in PROLOG. This language was developed during artificial intelligence research and is therefore very adept at
representing and manipulating symbolic information. The language also has list processing features in addition to the usual numeric facilities.

Given the symbolic processing abilities of the language the If-Then rules were very easy to represent in PROLOG. The method used to construct the rules is based on the technique described by Bratko [46]. Bratko employed user defined operators, such as 'and', 'or', 'if' and 'then', to create an If-Then rule structure for a simple expert system shell. The adapted rule structure used within the rulebase source code is shown below:

```
rule 'X' :
    if
        'condition(s)' 
    then
        'consequence(s)'.
```

The hierarchy of the operators can be seen more clearly in Figure 3.3.

![Figure 3.3 The Hierarchy Of The Prolog Operators](image)

The colon operator has the highest priority or precedence and divides the rule into a head and a body. The head simply allows the rule to be assigned a unique index, whilst the rule body contains the condition and consequence information. The 'then' operator is infix
and therefore resides in between its two arguments, the 'if' operator and the rule consequences. The 'if' operator simply prefixes the rule conditions to improve legibility.

The rule conditional part can include both AND and OR logic. Alternative sets of conditions are delimited using the 'or' operator, and conjunctions of conditions within each set are defined using the 'and' operator. The two types of conditions, basic unit failures and process variable deviations, are represented using the following syntax:

device DEVICE NAME fails FAILURE MODE

or

variable_type VARIABLE TYPE => DEVIATION STATE

In both cases, the hierarchy of the structures is the same. The 'fails' and '=>' operators are infix and bind the name of the unit or process variable to the fault state. The 'device' operator prefixes the device name and the variable type operator, such as 'level' or 'pressure', prefixes the process variable name. A complete list of the PROLOG operators used within the development KBS, and their definitions, is included in Appendix A.

To illustrate the syntax of the IF-Then rules more clearly, the following two examples are shown below:

rule 1:
if
  flow fl => high
then
  level 11 => high.
rule 2:

if
device cnt_l fails high
then
level 11 => high.

3.2.2 Direct Versus Indirect Alarm Fault Trees

As can be seen from rules 1 and 2, the alarm causes are not directly related to the process alarm they trigger, but rather to the alarmed process variable deviation. This distinction is maintained for two reasons:

1 It is useful to be able to consider process variable states which are at variance with their indicated state. For example, if the level sensor in the buffer tank system (shown in Figure 3.1) fails high, the controller will react by closing the inlet valve. In time the buffer tank level will become low and cause a low flow deviation downstream. Unless it is possible to distinguish between the alarm and the actual state of the level, it will be more difficult to relate the downstream alarms to the low tank level and then to the level sensor fault.

2 If a process variable is monitored by more than one indication, as is often the case in the nuclear industry, the duplicated alarms will have an almost identical set of causes. The only difference will be the causes of the spurious failure of each alarm. In this situation it is grossly inefficient (both in terms of storage space and processing effort) to directly relate the alarms to their primary failures.

For example, consider two alarms A and B which both represent a high deviation of process variable Q. Figure 3.4 illustrates the two alarm fault trees which directly relate the alarms to their root causes, without referencing the high deviation state of variable Q.
As can be seen, the five process causes U, V, W, X and Y are common to both trees. However, each tree also includes a unique fault (Z or R) which represents the alarm specific faults, such as the spurious failure of the alarm instrumentation.

**Figure 3.4 The Direct Alarm Fault Trees**

After reducing both 'direct' alarm fault trees into minimum cutsets, twelve If-Then rules would be necessary to describe all the causes of the two alarms. Five of these rules would contain duplicated rule antecedents.
An alternative organisation of the information (that used within the KBS) is shown in Figure 3.5. In this case, process faults U, V, W, X and Y are first related to the process variable deviation they cause, namely Q -> high. This deviation is then related, along with any instrumentation faults, to the process alarms.

**Figure 3.5 The Indirect Alarm Fault Trees**

By removing the If-Then rules with duplicate antecedents only nine rules are now required to describe how the process and instrument faults relate to their respective alarms. This improves the storage efficiency of the rulebase. However, more importantly, the information contained within the alarm rules has been enhanced to include the assumed state of process variable Q. This knowledge is invaluable when the diagnoses of multiple alarms are checked for logical consistency, as discussed later in the thesis.
3.2.3 Defining Additional Boundary Conditions

Occasionally the consequences of a process fault may be directly observable through another indication. For example, a steam traced pipe may have a temperature sensor attached at some point along its length. If the steam tracing fails, the contents of the pipe may freeze, or become more viscous, and hence trigger a low flow alarm.

The causes of the low flow alarm should therefore include the fault 'device steam_tracing fails off'. However, if this fault is active the pipe temperature will be low. This piece of information can therefore be used to confirm or reject a hypothesised cause of an alarm. The extra information (termed a boundary condition) is represented in the alarm cause rules in the following format:

```
rule 20 :
  if
    device steam_tracing fails off
  then
    flow fl => low
  and
    temperature tl => low.
```

Using the 'and' operator any number of boundary conditions can be associated with a particular rule by binding them to the rule consequence part.

3.2.4 Representing Process Sub-systems

When control or trip loops are present on a process plant, the alarm explanations generally include failure modes of these protective systems, in combination with the basic unit failures or process variable deviations. This can be seen in the cutsets derived from the alarm fault tree shown in Figure 3.2.
Unfortunately, the AND logic associated with these control or protection systems increases both the number and size of the minimum cutsets derived from an alarm fault tree. Both these attributes increase the diagnosis execution time, as discussed later in this thesis.

In order to reduce the number of minimum cutsets, the KBS has been designed to allow protective system failures to be grouped together as one fault, rather than being expanded into the individual unit failures. These compounded faults are then represented in the fault propagation rules, in following form:

control_loop 'NAME' fails 'FAILURE MODE'

or

trip_loop 'NAME' fails 'FAILURE MODE'

The causes of the subsystem or compound unit failure are then described in a separate rule. For example, it can be seen from Figure 3.2 that three of the causes of a high tank level include the 'control loop fails normal' fault. Because of the AND gate in the tree, these three faults are each combined with the two causes of the control loop failing normal which results in the following six minimum cutsets:

6 Flow f2 low and control valve cv_1 fails normal;

7 Flow f2 low and controller cnt_1 fails normal;

8 Pipe 3 fails blocked and control valve cv_1 fails normal;

9 Pipe 3 fails blocked and controller cnt_1 fails normal;

10 Flow sensor fs_2 fails blocked and control valve cv_1 fails normal;

11 Flow sensor fs_2 fails blocked and controller cnt_1 fails normal.
If the control loop failure is modelled as a single fault then the six expanded cutsets can be replaced by the three shown below:

1. Flow $f_2$ low and control_loop $c_{ll}$ fails normal (6 and 7)
2. Pipe 3 fails blocked and control_loop $c_{ll}$ fails normal (8 and 9)
3. Flow sensor $fs_2$ fails blocked and control_loop $c_{ll}$ fails normal (10 and 11)

The control loop failure is modelled in the foregoing using the following rule:

rule 20 :

if

device $cv_1$ fails normal

or

device $cnt_1$ fails normal

then

control_loop $c_{ll}$ fails normal.

The consequence part of this rule then matches the clause in the main alarm If-Then rule.

Unfortunately, the subsystem failure rule shown above is only valid when level deviations are being considered, because both the level alarm and the control loop share the same sensor. If deviations in flow $f_1$ or $f_2$ are being modelled, then level sensor faults must also be considered as a cause of the control loop failure.

In order to make the subsystem failure rule generic to all process variable deviations the level sensor fault is included, but it is ANDED with an extra boundary condition as shown overleaf:
rule 20 :
    if
       device cvl fails normal
    or
       device cntl fails normal
    or
       device lsl fails normal
    and
       signal sl is_in_state normal
    then
       control_loop cll fails normal.

3.3 An Overview Of The Alarm Diagnosis Strategy

The diagnostic strategy used within the development KBS is summarised in the following text. A request for the diagnosis of an alarm is passed to the KBS by the operator. As Andow [25] discusses, the alternative mode of operation is for the KBS to diagnose the alarms automatically as they are detected. Andow then goes on to say that it is also possible (and perhaps desirable) to design systems that lie between these two extremes. However, during this project the operator driven approach was selected for simplicity, since the operator interface was not the main focus of the research.

When the KBS processes the diagnosis request, it first attempts to causally relate the new alarm to a previously diagnosed alarm. In this way the new alarm can be classified as either another manifestation of an already diagnosed fault, or an indication of a new process disturbance. Welbourne [4] considered this classification process to be an important feature of the alarm analysis system developed for the Wylfa nuclear power station.

Three types of causal relationship are considered. If a previously diagnosed alarm is a cause, a consequence or shares common failure modes with the new alarm, the two are considered to be related. Unfortunately the task of interrogating all the alarm If-Then rules is time consuming because of the method used to structure the
knowledge within the KBS. Therefore all the potential links between each alarm are identified off-line whilst the knowledge base is compiled. The network of alarms can then be quickly interrogated in real time, as discussed in Chapter 6.

The second stage of the diagnosis procedure depends upon the outcome of the first stage. If no causal links can be found, then the new alarm is considered in isolation. However, if the new alarm is potentially related to another diagnosed alarm, then all the symptoms are analysed in conjunction with each other by modifying the previous diagnosis. The intention is to be able to explain each pattern of causally related alarms in terms of a minimum number of process faults. For example, consider alarms A, B and C, whose fault trees are illustrated in Figures 3.6a, 3.6b and 3.6c.

Figure 3.6a  The Causes of Alarm A
Figure 3.6b The Causes of Alarm B

ALARM B

OR

VARIABLE Q IS HIGH

OR

VARIABLE R IS HIGH

FAULT Y

SPURIOUS HIGH FAILURE OF ALARM B

Figure 3.6c The Causes of Alarm C

ALARM C

OR

VARIABLE R IS HIGH

OR

FAULT Z

SPURIOUS HIGH FAILURE OF ALARM C
If all three alarms become active, they will be diagnosed in conjunction with each other because they are causally related. Figure 3.7 illustrates these relationships more clearly in terms of a cause consequence type structure.

**Figure 3.7 The Causal Relationships Between The Three Alarms**
As can be seen from the diagram, all three alarms can be caused by fault Z. However, in addition to the single fault cause, there are three other possibilities which must not be overlooked, namely:

1. Alarm C spuriously fails and fault Y;
2. Alarm C spuriously fails and alarm B spuriously fails and fault X;
3. Alarm C spuriously fails and alarm B spuriously fails and alarm A spuriously fails.

By diagnosing the related alarms in conjunction with each other, the three symptoms can therefore be explained in terms of four possible fault scenarios.

As discussed in Chapter 4, the method of combining the diagnoses involves logically ANDing the individual alarm fault trees together. The resulting fault scenarios are then checked for simplifications and logically consistency.

Finally, the alternative explanations for either a single alarm or a combination of alarms are ranked according to their likelihood. As discussed in Chapter 5, the ranking procedure uses 'a priori' failure rate data to estimate the frequency of each fault scenario. In certain cases, the process indications are also used to help confirm or reject a particular fault scenario, and therefore a rigorous method is used to interpret these indications.

On completion, the ranked list of alarm explanations is presented to the operator. In this way the investigative efforts of the operator can be focused onto the most likely alarm cause first. If the highest ranked fault scenarios are subsequently found to be invalid, the operator's attention can then turn to the less probable explanations. The operator is therefore not required to hypothesise the more obscure fault scenarios, a task which could make heavy demands on the human imagination and memory.
Chapter 4

CONJUGATING THE INDIVIDUAL ALARM DIAGNOSES

The previous chapter broadly outlines the alarm diagnosis methodology. The purpose of this chapter is to explain in more detail how the causal relationships between the individual alarms are detected, and to discuss the method used to conjugate the alarm causes.

4.1 The Detection Of The Causal Couplings Between Alarms

As discussed in Chapters 3 and 6, when the rulebase information is compiled, the causal relationships between each alarm (specified in the alarm diagnosis rules) are summarised, so that they can be interrogated in real-time.

Three types of causal coupling are considered. Firstly, the alarmed process variable deviations which cause each alarm are noted. Secondly, the consequences of the process variable deviation, represented by each alarm, are noted. Finally, those alarms which share common failure modes are noted. Within the compiled rulebase information, each alarm is therefore associated with a list of alarms, which can, in certain circumstances, be related to that alarm.
When an alarm is to be diagnosed, its list of related alarms is used to decide if the new alarm is simply another manifestation of an already diagnosed fault or an indication of a new process disturbance. For example, re-consider the buffer tank system as shown below in Figure 4.1.

Figure 4.1 The Buffer Tank System

When the KBS diagnoses a low level alarm, it will first search the PROLOG database to determine if either the upstream low flow alarm or the downstream high flow alarm has been diagnosed. The latter are selected because they are the only alarms that cause the low level alarm.
In the second phase of the causal search, the KBS checks if either the high flow \( f_1 \), the low flow \( f_1 \) or the low flow \( f_2 \) alarms have been diagnosed. The high flow \( f_1 \) alarm is related to the low level alarm, because they both can be caused by a leakage in pipe 2. Similarly a blockage in pipe 2 will cause a low flow \( f_1 \) alarm and a low level alarm. Thirdly, both the low tank level and the low discharge flow alarms can be caused by a leakage in pipe 3.

Finally, the consequence of a low tank level, namely a low discharge flow alarm, is examined to determine if it has been diagnosed.

Whilst the compiled knowledge of the causal couplings between the process alarms does provide an excellent starting point, the KBS cannot decide if two alarms are actually related, solely on the basis of this 'a priori' information. In practice, the final decision also needs to take account of the time ordering of the alarms. For example, consider the buffer tank system. If both the low tank level and the low discharge flow alarms activate shortly after each other, then there is a strong possibility that the two events are causally related. However, if there is a significant time delay between the two alarms, this tends to indicate that they are unrelated (providing that they can occur independently). In the latter case there is little point in diagnosing the alarms in conjunction with each other.

Unfortunately, as Lees [38] discusses, the fault tree method of describing fault propagation cannot easily represent time delays and the sequencing of events. The cause consequence diagram (CCD) method is a little more flexible, as was demonstrated by the GRS - Halden and EPRI- DASS disturbance analysis projects. In these cases time delay information was incorporated into the CCD's, in order to identify if the process alarms were causally related.
However, regardless of the problems of describing time within either of the two graph based representations, the major problem still remains in estimating reliable values for the time delays between the alarms. This is mainly because of the following four factors:

1. It is important that the fault propagation dynamics between the process alarms should be well understood. This can be achieved either by perturbing the real process, or by simulating disturbances using a mathematical model of the system. Unfortunately, the former alternative is not possible in many situations because of safety or production constraints, whilst the latter option can often be quite time consuming and hence expensive. Furthermore, if the whole system needs to be dynamically simulated, this cancels most of the benefits of using a discrete state method of synthesising the fault propagation rules, such as FAULTFINDER.

2. The time delays between the alarms will often be a strong function of the alarm limit values themselves. As a consequence, if these values are modified during the lifetime of the process, it may be necessary to readjust a number of the fault propagation time delays.

3. In some situations the expected fault propagation time delay between two alarms will depend upon the state of another process variable. For example, consider the continuously stirred mixer tank shown overleaf in Figure 4.2.
Providing that there are no heat losses from the tank, and there is no internal generation of heat, the thermal behaviour of the system can be described by the unsteady state heat balance given in Equation 4.1.

\[ X_1 Q_1 - X_2 Q_2 - L A \frac{dX}{dt} = 0 \]  \hspace{1cm} (4.1)

Where

- \( X \) is the fluid temperature
- \( Q \) is the fluid flowrate
- \( L \) is the tank level
- \( A \) is the cross sectional area of the tank
- \( t \) is the time

As can be seen from Equation 4.1, if both the flowrate through the system and the tank level remain relatively constant, the time delay between an inlet and outlet temperature deviation of the same magnitude, will also remain relatively constant. However, if either the liquid flowrate or level change significantly, the time delay will also change significantly.
In theory, the expected time delay between two temperature alarms could probably be estimated from the values of the level and flow variables. However, this dynamic model based approach could lead to further problems for the following reasons. Firstly, a reasonably detailed model would be required to predict the time delays with any degree of accuracy. This would be expensive, both in terms of the computer resources and the project cost. Secondly, if either the level or flow sensors failed in a misleading state, the time delay prediction would be incorrect, which could in turn lead to an incorrect diagnosis.

4 If a number of alarms are potentially related because they share common failure modes, the important fault propagation time delays are those between the initiation of the faults and the manifestation of their symptoms. For example, consider the simple event tree shown in Figure 4.3.

**Figure 4.3 The Simple Event Tree**

```
Alarm A (ta)  Alarm B (tb)
               |
            /|
           Da Db
          / |
        /   |
      /     |
   /       |
 Fault X
```

Alarms A and B are detected at times \( ta \) and \( tb \) respectively, and are potentially related by fault X. If the propagation time delays \( Da \) and \( Db \) could be estimated, then two approximate fault initiation times could be calculated as \( ta - Da \) and \( tb - Db \). Providing that these two times were in reasonable agreement, this would indicate that fault X was a possible common cause, and consequently the alarms could be diagnosed in conjunction with each other.
Unfortunately, a qualitative method of describing the fault severities does not enable the fault propagation time delays to be estimated with any degree of accuracy. This is principally because of their dependence on the severity of the faults. For example, a small leakage in a tank might take ten minutes to cause a low level alarm, whereas a large leakage will have a much more immediate effect.

As discussed in Chapter 9, the problems of estimating the fault propagation time delays are not considered to be insurmountable. However, in order to explicitly take account of fault propagation time delays it would be necessary to augment the synthesised fault tree information with dynamic information, which would tend to cancel most of the benefits of using FAULTFINDER. Within the KBS a heuristic based method is therefore used to decide if two alarms are causally related.

The heuristic simply states that two related alarms will coexist together for some period of time. The practical lower limit for this time window being the scan interval of the alarm monitoring system. Implicit within this heuristic is the assumption that the time delay between the first alarm (in any pair) detecting a fault and the second alarm detecting the same fault, is less than the duration of the fault.

For example, reconsider the two alarms, A and B, shown in Figure 4.3. Let us assume that alarm A has already been diagnosed and alarm B is new. If both alarms are caused by fault X, then they will coexist providing that fault X is active (unrepaired or in deviation) for longer than the difference between propagation delays Da and Db. If this is the case, alarm B will be diagnosed in conjunction with alarm A. The heuristic implies same assumption if alarm A is suspected to be a direct consequence of alarm B or vice versa.

When one of the related pair of alarms has already been considered in conjunction with other alarms, the criterion is only applied to the two alarms thought to be related, rather than all the alarms. The implications of the assumption are discussed in Chapter 9.
4.2 Deciding How The Related Alarms Should Be Combined

When the causal couplings between a new alarm and any previously diagnosed alarms have been determined, one of three possible courses of action is taken by the diagnosis procedure, depending upon the number of links that are identified:

1. The simplest situation arises if the new alarm cannot be related to any previously diagnosed alarms. In this case the causes of the new alarm are considered in isolation.

2. If only one causal link is detected, the causes of the new alarm are simply combined with those of the second alarm. When the second alarm has already been considered in conjunction with other alarms, then all the alarms are diagnosed in combination.

3. If the currently diagnosed alarm is related to more than one alarm, and all these alarms have not already been considered in conjunction with each other, then the diagnosis procedure becomes more complex. The problem arises in deciding how the diagnosis for the new alarm should be combined with the other sets of diagnoses.

There are basically two alternative options, either the new alarm causes can be combined with each of the previous diagnoses separately, or the previous diagnoses can all be combined with the new alarm causes, to form a single new diagnosis for all the alarms in question. The former option has two main drawbacks:

1. If the causes of the new alarm are combined with a number of separate alarm diagnoses, then the diagnosis execution time will increase.

2. More importantly, the diagnosis method will result in at least two alternative alarm diagnoses, describing the causes of overlapping groups of alarms. The difficulty lies in deciding how this information should be presented to the process operator.
For example, consider two sets of alarms \( \{A, B, C\} \) and \( \{X, Y, Z\} \) which were diagnosed separately, because there were no causal connections between the alarms contained in either set. If a new alarm \( P \) is potentially related to both alarms \( A \) and \( Y \), then using the first approach two distinct sets of alarms would be generated, \( \{A, B, C, P\} \) and \( \{X, Y, Z, P\} \). If a process operator wished to investigate the causes of the new alarm, the two alternative diagnoses would need to be consulted. Furthermore, any subsequent alarms, related to alarm \( P \) would also need to be conjugated with both diagnoses.

Unfortunately, the alternative approach of combining all the alarms that are related to a new alarm is not without its own problems. The main difficulty lies in the increased order of the cutsets, which is because each cutset must specify the causes of at least two unrelated alarms. For example, consider the following cutsets for the two unrelated alarm groupings \( \{A, B\} \) and \( \{X, Y\} \):

**The causes of alarms \( A \) and \( B \)**

\[
\begin{align*}
1 & \{i, j\} \\
2 & \{k, l\}
\end{align*}
\]

**The causes of alarms \( X \) and \( Y \)**

\[
\begin{align*}
1 & \{p, q\} \\
2 & \{r, s\}
\end{align*}
\]

If the causes of a new alarm \( D \) are as follows, then it can be clearly linked with either alarm groupings:

\[
\begin{align*}
1 & \{s\} \\
2 & \{i\}
\end{align*}
\]
When all five alarms are combined the following cutsets will be derived:

1. \( \{i,j,r,s\} \)
2. \( \{k,l,r,s\} \)
3. \( \{i,j,p,q\} \)
4. \( \{k,l,p,q,i\} \)
5. \( \{k,l,p,q,s\} \)

If the new alarm had been conjugated with the two alarm groupings independently, then the following two groups of cutsets would have been generated:

1. \( \{i,j\} \)
2. \( \{k,l,s\} \)
3. \( \{k,l,i\} \)

and

1. \( \{p,q,i\} \)
2. \( \{p,q,s\} \)
3. \( \{r,s\} \)

As can be seen from the cutsets shown above, the smallest explanation for all five alarms contains four elements. This compares with a minimum cutset order of two in the above lists.

In general, a cutsets frequency of failure will vary inversely to the number of elements within it. The frequencies of the explanations for all five alarms will therefore be considerably smaller than the
explanations for the two alternative alarm groupings. Since the operator will be looking for the most plausible explanation for the new alarm, the explanations for all five alarms would appear less attractive.

However, as discussed in the following chapter, the ranking procedure used within the KBS prioritises alternative explanations based on their relative frequencies of occurrence. As a consequence the relative merits of the higher order cutsets will appear to be similar to the relative merits of the lower order cutsets.

Whilst both methods were noted to suffer from drawbacks, the second of the two approaches was considered to suffer from the least disadvantages, and hence has been implemented within the KBS.

4.2.1 Taking Into Consideration Latent Alarm Failures

For the reasons discussed in Chapter 3, the causes of an alarm are only developed to deviations of the first alarmed process variable in each causal path. As a result, when the diagnosis procedure searches for those alarms which are either causes or consequences of a new alarm, the causal search is also restricted to the first alarm in the causal path. To illustrate the problem, consider the buffer tank system shown in Figure 4.1. For example, within this system deviations in the discharge flow \( f_2 \) would only be traced upstream as far as the tank level \( l_1 \).

Whilst this diagnostic strategy greatly simplifies the searching task, it is not robust enough to handle latent failures of the alarm instrumentation. For example, in Figure 4.1 if the upstream flow \( f_1 \) deviates low enough to cause an alarm, then the tank level will start to decrease. Providing that the downstream alarm limits are set appropriately, this will eventually result in a low level alarm and a low discharge flow alarm. In this situation the diagnosis procedure will be able to relate all three alarms. However, if the level sensor fails invariant, the two flow alarms will be treated as being independent, since they cannot be connected by the intermediate event.
The diagnostic strategy has consequently been modified to take account of latent alarm failures. Three possible scenarios are now considered by enhanced causal searching procedure:

1. The first situation has already been described in the previous example, whereby an intermediate alarm fails inactive between two active alarms. If the latest alarm is observed to coexist with its distant relation, then the two alarms are considered in combination with each other. However, rather than just combining the causes of the new alarm with those of the other alarm(s), a slightly more complex approach is adopted.

   In order to take account of the failure of the intermediate alarm, all the causes of the connecting un-alarmed variable deviation are tagged with the condition that its alarm instrumentation has failed either passively or in another deviation state. These special cutsets are then conjugated with the causes of the new alarm and the causes of the previously diagnosed alarms. The possibility that the two events are really independent is taken into consideration by separately conjugating the causes of the active alarms. At this stage the two sets of alarm explanations are then added together and processed as normal.

   For example, re-consider the three alarms A, B, and C whose alarm fault trees are shown in Figures 3.6a, 3.6b, 3.6c and 3.7. If alarms A and C activate without alarm B, the KBS will first identify that alarm A can only be related to alarm C if variable Q is HIGH. However, this should trigger alarm B. The causes of alarms A and C are therefore conjugated with the causes of variable Q being HIGH plus the extra condition that indication B does not alarm HIGH. The results of this combination are listed below:

   1. Indication B does_not_alarm HIGH and Fault Z

   2. Fault Y and indication B does_not_alarm HIGH and indication C spuriously_alarms HIGH

73
When the causes of alarms A and C independently alarming are conjugated, the following cutsets are generated:

1. Indication A spuriously alarms LOW and Fault Z
2. Fault X and Fault Z
3. Indication A spuriously alarms LOW and Indication C spuriously alarms HIGH
4. Fault X and Indication C spuriously alarms HIGH

Because both of scenarios are equally plausible, the two sets of cutsets are added together to yield six explanations.

The second scenario is best illustrated with the aid of Figure 4.4. If two alarms, B and C, are active and both are potential consequences of the process variable deviation which should be represented by alarm A, then the causes of all three alarms are combined using the previously described technique.

**Figure 4.4 The Second Latent Alarm Failure Scenario**

```
  Alarm B       Alarm C
   _         _
   \       / |
    \     /  
     \   /    
      \  /     
       \ /      
          \    
            \  
              
            Alarm A <---- Inactive alarm
```

The third situation in which a latent alarm failure is considered is again most easily described with reference to a simple diagram, namely Figure 4.5.
If alarms A and C are possibly related as shown in Figure 4.5, then providing that they cannot be related by any other means, the causes of alarms A and C are combined. As before, the causes of the process variable deviation represented by alarm B, are also considered in conjunction with those of the active alarms.

In all three situations, the KBS will only consider a single passive alarm fault between any two active alarms. This limitation is based on the assumption that two or more consecutive alarm instrumentation faults, in the same disturbance propagation pathway, will be a very rare occurrence.

As a general rule the assumption is valid. However, in the event of a common mode failure, such as an instrument cable duct being severed, the assumption breaks down. Despite this weakness, the assumption is retained because of the need to reduce the complexity of the causal searching procedure.

The causal searching strategy and the resulting actions that are taken, are summarised in the flowsheet shown in Figure 4.6.
PERFORM DEEPER SEARCH. ALLOW 1 PASSIVE ALARM FAILURE

SEARCH FOR CAUSAL LINKS WITH NEW ALARM

ANY FOUND

N

Y

ONE FOUND

N

Y

ANY FOUND

ONE FOUND

N

Y

COMBINE NEW ALARM WITH ONE DISTANT RELATION

COMBINE RESULT WITH LATENT ALARM FAILURE

ADD THE TWO LISTS TOGETHER

Y

ANY OTHER LEFT

N

COMBINE ALL THE LISTS TOGETHER

EXIT

Y

N

COMBINE NEW ALARM WITH OTHER ALARM

EXIT

CONSIDER THE NEW ALARM IN ISOLATION

EXIT

COMBINE THE TWO ACTIVE ALARMS

COMBINE THE RESULT WITH LATENT FAILURE ALARM

COMBINE ALL ALARM GROUPS RELATED TO NEW ALARM

COMBINE RESULT WITH THE NEW ALARM

EXIT

EXIT

EXIT
4.3 Combining The Alarm Causes

The task of analysing a group of related alarms, in order to determine their various root causes, is a four stage process. Firstly, the individual alarm fault trees are logically conjugated. The resulting structure is then reduced into cutset form. Secondly, the cutsets are checked for logical consistency, which involves searching each cutset for mutually exclusive faults. Thirdly, the remaining cutsets are examined to determine if they can be simplified, and finally the cutsets are checked for minimality.

Each of these four stages are discussed in more detail in the following text.

4.3.1 Combining The Lists Of Alarm Causes

Although all the causally related alarms are diagnosed in conjunction with each other, the KBS only logically AND's two sets of alarm causes together at any one time. For example, referring back to Figure 3.6, if alarms A, B and C become active then the causes of the first two alarms will initially be combined. The causes of the third alarm will then be combined with the result of the first conjugation.

Despite the fact that it is almost as easy to conjugate the causes of three or more alarms, as it is to conjugate the causes of two, the binary approach has been adopted within the KBS. This is principally because of the following two reasons:

1. The diagnosis technique has been designed to be incremental. It is assumed that when a process fault occurs, all the resulting alarms will not occur simultaneously, but rather over a period of time. The KBS will therefore attempt to diagnose the first manifestations of the fault, in order to present the process operator with the best possible assessment of the alarm causes at that time, instead of waiting until all the eventual symptoms become apparent. As more related alarms occur,
the original diagnosis will then be refined in the light of
the new information. The technique of conjugating the causes of
the most recent alarm, with the combined causes of the other
related alarms, appears to be an efficient method of implementing
this diagnostic strategy.

An effort has been made to minimise the diagnosis execution
time. When a number of alarm fault trees are combined, the number
of raw cutsets generated is equal to the product of the number
of causes of each alarm. So for example, if the causes of alarm B are combined with the causes of alarm C, the six cutsets shown below will result:

a) Alarm B spuriously fails and fault Z

b) Alarm B spuriously fails and alarm C spuriously fails

c) Fault Y and Fault Z

d) Fault Y and alarm C spuriously fails

e) Variable R in state high and fault Z

f) Variable R in state high and alarm C spuriously fails

If these six cutsets are conjugated with the causes of alarm A, eighteen cutsets will result. In reality, however, an alarm may have in the order of ten causes, so the difference between anding two or three alarms may be very significant.

The major benefit of combining the alarm causes, using a
binary method, only really becomes apparent when the cutsets are
checked for minimality. As will be discussed in Section 4.6, if
the alarm fault trees are causally related, the raw cutsets may
be reduced to a number which is comparable with the sum of the
causes of both alarms, rather than their product. This
therefore means that if the causes of two alarms are combined and
checked for minimality, and these minimum cutsets are then combined with the causes of a third alarm, the maximum number of cutsets that will need consideration at any one time, will be reduced.

It must be stated that the final number of minimum cutsets for any number of alarms, will be independent of the method used to derive them. However, since the computer processing time increases dramatically as the number of cutsets increases, if the number of cutsets that require processing is reduced, so will the diagnosis execution time.

4.3.2 The Problem Of Combinatorial Explosion

The major drawback of the diagnostic procedure described in the foregoing is the problem of combinatorial explosion. This is best illustrated with the aid of a simple example. Consider the four alarms P, Q, R and S, which have the following causes:

P \{a,b,c,d,e,f\}
Q \{d,g,h,i,j,k\}
R \{h,l,m,n,o,t\}
S \{t,u,v,w,x,y\}

Each list of causes has at least one common element with another list, therefore all four alarms would be diagnosed together. The number of explanations generated, as successive alarms causes are conjugated, can be seen in Figure 4.7. As each new alarm is considered within the diagnosis, the total number of explanations increases by about a factor of 4.3. After four alarms, with only six causes each, 476 possible explanations result.
Clearly the causal connections between alarms P, Q, R and S are somewhat tenuous, which gives rise to the dramatic combinatorial explosion. When alarms are more closely related, the resulting diagnoses can contain considerably fewer explanations. For example, in the case of alarms A, B and C, whose fault trees are shown in Figures 3.6a, 3.6b and 3.6c, only four cutsets are generated when the causes of all three alarms are combined.

Although there will be many situations when the alarms to be diagnosed will be closely related, the diagnostic strategy has to be able to cope with more tenuously related alarms. However, as mentioned earlier the diagnosis execution time is a strong function of the number of alarm causes, therefore the KBS must also ensure that the diagnostic task is tractable within a reasonable time frame.

Given that the diagnosis execution time will be a limiting constraint in many situations, there are three possible solutions to the problem of combinatorial explosion:
1 Increase the processing power of the host computer;

2 Conjugate the alarm causes in an order which will minimise the total number of explanations generated;

3 Reject the unlikely explanations from an existing diagnosis before they are combined with the causes of a new alarm.

From a theoretical point of view, the first option is the most desirable. However, the cost of solving the problem in this way could be prohibitive.

Although the second option would not reduce the total number of explanations for any set of alarms, since this is independent of the order in which they are conjugated, the number of intermediate explanations for a subset of the group could be reduced, because this can be a strong function of the order of conjugation. For example, referring back to the four alarms P, Q, R and S, if alarm S is an indication fault d, then conjugating the alarms in the order S, P, Q and R would result in fewer intermediate cutsets than the alphabetic order.

By conjugating alarm causes in an optimal order, the diagnosis execution time could be reduced. This would be advantageous when a large number of related alarms were detected simultaneously, although the approach would be of limited use if the alarms were separated by time delays greater than the diagnosis execution time.

Despite the merits of the second technique, the third option was employed within the KBS because it would guarantee that the total number of alarm explanations could be controlled, albeit by rejecting diagnostic information. In the future, the diagnostic strategy could be further enhanced by implementing the second technique within the framework of the third approach.
As will be discussed in Chapter 5, the minimum cutsets that are generated by conjugating the alarm fault trees, are ranked according to their likelihood, before being presented to the process operator. The diagnostic procedure has therefore been refined by ensuring that before the causes of a new alarm are combined with the explanations for an existing pattern of alarms, the least likely causes in the ranked list are removed. For the purposes of evaluating the performance of the diagnostic method, only the first ten most likely alarm explanations are retained by the KBS. However, in any practical implementation of the software, this number could obviously be adjusted to match the processing power of the computer and/or the requirements of the operator.

The biggest drawback of eliminating some of the cutsets is that even if a fault is common to a number of active alarm fault trees, if the occurrence of the fault is considered to be unlikely, then the common failure mode might be rejected from the diagnosis. For example, consider the three simple alarm fault trees shown in Figure 4.8

**Figure 4.8** The Example Alarm Fault Trees

![Alarm Fault Trees Diagram](image-url)
Top events X, Y and Z represent active alarms and basic events A-G represent process faults. If the fault trees for alarms Y and Z are combined, then five possible alarm explanations will result, as listed below:

1. A
2. DF
3. DG
4. EF
5. EG

Assuming for this example that fault F is rare and fault A is very rare, then the solutions could be ranked in the following order:

1. DG
2. EG
3. DF
4. EF
5. A

When the third fault tree for alarm X requires to be considered with those of alarms Y and Z, if the least likely solution for the two alarms is rejected, the common mode fault A will not be identified. As a consequence, the following twelve cutsets will be assumed to cause the three alarms, whereas in actual fact, cutsets 1-4 could be replaced by the single cause A:
Furthermore, fault A may be considered to be a rare event, but it might be more likely than a combination of three independent faults.

In an attempt to address this shortcoming, the diagnosis procedure checks all the rejected cutsets to determine any common failure modes with the other list of alarm causes. If a match is found, the rejected cutsets are reinstated in the list to be combined.

4.3.3 Conjugating The Alarm Causes

As discussed in Chapters 3, the natural language fault propagation rules are pre-processed by the rulebase compiler into a nested list structure. One benefit of gathering all the causes of
the same process alarm into a single structure is the reduction in the time required by the PROLOG language to search for all the alarm causes. However, the main advantage of using a nested list structure is the ease with which the alarm cause information can be manipulated. The method used to combine the alarm cause lists illustrates the point, but before discussing the details it is first useful to summarise the information content of the compiled rules.

The compiled rule list structure is a hierarchy of three levels. The elements of the principle list represent the alternative causes of the alarm (these correspond to the individual IF-Then rules). Each of these elements is in turn a list of five elements, which contain the following information:

1. One set of faults which will cause the alarm to activate (a single cutset).

2. The assumed state of the alarmed process variable resulting from the causes in the first list, e.g. 'flow fl is_in_state low' when the cutset causes an actual variable deviation or 'flow fl is_in_state unknown' when the cutset causes a spurious alarm.

3. Any boundary conditions (other variable deviations definitely caused by the cutset).

4. A list of cutsets (taken from the first list element) bound to the variable deviations they cause. This list is initially empty, however, as the causes of a number of related alarms are combined, the same list structure is used to store the explanations for all of the alarms. In order to determine which fault caused which alarm, it is necessary associate the faults with their consequences.

5. The frequency of the faults contained within the alarm explanation cutset in terms of the number of occurrences per million hours.
The procedure used to conjugate the two lists of alarm causes takes advantage of the recursive nature of the PROLOG language. The process begins by dividing the list of causes of the first alarm into its 'head' and 'tail' parts, the head being the first element of the original list, and the tail being the remainder. The head of the first list is then passed, along with the complete list of causes of the second alarm, to the second part of the program.

At this stage the second list is divided into its head and tail parts. The first four elements of the head of the first list are then combined with their counterparts in the head of the second list. The product of the combination of the two alarms are next checked for logical consistency and simplifications, as described in the next section, and the result is stored in a third list.

The second element of the second list is then combined with the head of the first list, and the whole process is repeated. When all the elements in the second list have been considered, the program control returns to the top level, where the second element of the first list is extracted. This alarm cause is then passed down to the second part of the program, where it is conjugated with all the causes of the second list. This procedure continues until the end of the first list is reached.

4.4 Checking The Cutsets For Logical Consistency

In certain circumstances, some of the cutsets produced by combining the alarm fault trees will contain mutually exclusive events, thus rendering them logically inconsistent. The problem arises quite naturally because each alarm fault tree is only developed to describe the causes of that alarm, without taking into consideration any other alarm states. For example, re-consider the simple buffer tank system shown in Figure 4.9.
The simplified cutsets describing the causes of a high inlet flow alarm and a high tank level alarm are listed below:

The causes of a high inlet flow alarm:

1. indication flow_fl spuriously_alarms high
2. control_loop cl_1 fails high (the control valve is fully open)
3. flow f2 -> high AND control_loop cl_1 is_working

Figure 4.9 The level controlled buffer tank
The causes of a high tank level alarm:

1. indication level_ll spuriously_alarms high
2. control_loop cl_1 fails high
3. flow f1 => high AND control_loop cl_1 fails normal (stuck)

If these two alarms are observed to coexist, then they will be diagnosed in conjunction with each other, resulting in the generation of the following nine cutsets:

1. indication flow_f1 spuriously_alarms high AND indication level_ll spuriously_alarms high
2. indication flow_f1 spuriously_alarms high AND control_loop cl_1 fails high
3. indication flow_f1 spuriously_alarms high AND flow f1 => high AND control_loop cl_1 fails normal
4. control_loop cl_1 fails high AND indication level_ll spuriously_alarms high
5. control_loop cl_1 fails high
6. control_loop cl_1 fails high AND flow f1 => high AND control_loop cl_1 fails normal
7. flow f2 => high AND control_loop cl_1 is_working AND indication level_ll spuriously_alarms high
8. flow f2 => high AND control_loop cl_1 is_working AND control_loop cl_1 fails high
9. flow f2 => high AND control_loop cl_1 is_working
flow f1 => high AND control_loop cl_1 fails normal
As can be seen from the list of cutsets, numbers 6, 8 and 9 specify that the control loop must be in two mutually exclusive states. Since this is clearly impossible, cutsets 6, 8 and 9 do not provide plausible explanations for the two alarms and hence they should be deleted.

The cutsets are checked for validity by ensuring that each element of the cutset is compared with every other element. It is also necessary to detect conflicts between the alarm causes and the boundary condition and assumed variable state information. Given that there are nine different types of fault propagation and process variable state information the diagnostic procedure must therefore check all the possible logical inconsistencies between this data. Table 4.1 illustrates the checks that are made between the different types of information, which are then discussed in Sections 4.4.1 to 4.4.7.

Table 4.1 The Logical Consistency Checks

<table>
<thead>
<tr>
<th>Subsystem state</th>
<th>Variable deviation</th>
<th>Device failure</th>
<th>Device working</th>
<th>Subsystem failure</th>
<th>Subsystem working</th>
<th>Spurious alarm failure</th>
<th>Passive alarm failure</th>
<th>Boundary conditions</th>
<th>Assumed variable state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable deviation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device working</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subsystem failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subsystem working</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spurious alarm failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Passive alarm failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boundary conditions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assumed variable state</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cross check made
4.4.1 Conflicts With Process Variable Deviation States

A cutset which contains two mutually exclusive states of the same process variable is rejected, providing that the two events are required to coexist at the same point in time. The first problem that has to be addressed, is deciding whether two different states of a process variable are always mutually exclusive. For example, if the continuous range of a variable is discretized into sharp non-overlapping regions, then all the discrete states will be mutually exclusive. However, if the discretisation relies on gradual or fuzzy transitions between state intervals, then the adjacent states will not always be mutually exclusive.

For the reasons discussed in Chapter 5, the latter of the two discretisation approaches is employed within the KBS. A cutset is therefore not rejected if it contains two different states of the same process variable, providing that they are adjacent in the discretisation. The only exception to this general rule arises if one of those variable deviations is represented by an alarm, the diagnosis of which is described by the cutset.

For example, a cutset containing the following two elements:

\[ \text{flow } f1 \Rightarrow \text{low} \ \text{AND} \ \text{flow } f1 \Rightarrow \text{very\_low} \]

would not be rejected providing that the cutset was not describing the causes of either a low or very low flow alarm. This is simply because a variable deviation is assumed to have a membership function of unity within a given fuzzy range, when the alarm limit associated with that range has been reached. In this situation all other variable states must be mutually exclusive with the alarmed state.

The second major difficulty stems from the fact that two mutually exclusive process variable states, will only render a cutset invalid if both the events are expected to coexist at the same point in time. As discussed in Section 4.1, if the alarms caused by the deviation states are observed to coexist, the deviation states are also assumed to coexist.
Finally, any process variable deviations within an alarm explanation cutset are cross checked against the process variable states assumed to be caused by all the elements in the cutset. The reason for this check is most easily explained by referring to the following example. Consider the steam heat exchanger shown in Figure 4.10, which raises the temperature of a liquid stream.

Figure 4.10 Steam Heat Exchanger

![Steam Heat Exchanger Diagram]

The flowrate of steam entering the heat exchanger shell is monitored, along with the liquid outlet temperature. Figures 4.11a and 4.11b illustrate the simplified alarm fault trees for a low steam flow alarm and a low outlet temperature alarm.
Figure 4.11a  The Mini-Fault Tree For A Low Steam Flow Alarm

Figure 4.11b  The Mini-Fault Tree For A Low Outlet Temperature Alarm
If the two alarms occur, and are diagnosed, the two mini-fault trees will be conjugated because of their direct causal link. As a consequence, the following four cutsets will be generated:

1. indication flow_f2 spuriously_alarms low AND flow f2 => low
2. indication flow_f2 spuriously_alarms low AND indication temp_tl spuriously_alarms low
3. device boiler fails shutdown AND flow f2 => low
4. device boiler fails shutdown AND indication temp_tl spuriously_alarms low

The first cutset is erroneous because it includes an undeveloped process variable deviation state and the spurious failure of the alarm which would, in normal circumstances, mislead the operator to think that the said deviation state actually existed. Since the deviation will cause the alarm to activate, the spurious alarm failure is not necessary. The remainder of the cutset is of no use because the causes of this deviation have been developed elsewhere. As a general rule, the diagnosis procedure therefore rejects any cutsets which include a variable state as a cause of an alarm, and also contains a spurious alarm failure of the same variable and state.

4.4.2 Conflicts With Device Failure States

The task of detecting mutually exclusive device failure modes is not as straightforward as it is with process variables. This is because failure modes of process equipment are not restricted to the variable discretisation states. For example, a pipe blockage and simultaneous leakage in the same unit may be a very rare event, but the two faults can coexist and still possibly cause their respective alarms. The benefit of the doubt is therefore given to cutsets which contain two failure modes of the same device, that are not necessarily mutually exclusive. However, when a cutset does contain two device failure modes, which are both members of the discretisation states of
the alarmed variables, then the same checks as outlined in Section 4.4.1 are applied, except in the following case. If a device is thought to have failed in a normal state, this is deemed to be mutually exclusive with the device failing in either of the alarmed variable's adjacent discretisation states (usually high and low).

The difficulties of discerning mutually exclusive device states do not apply if one of the cutset elements is a device working condition. In such cases, the alarm coexistence heuristic can be applied to decide if the two elements refer to the same time period. However, even if they do not, the cutset can still be rejected if the device is expected to fail and be repaired too quickly. The diagnosis procedure therefore first checks to determine if the device failure mode occurs first in time. If this is the case, the estimated detection and repair time for the fault is then accessed from the rulebase. If the fault is expected to be rectified too quickly, the cutset is rejected.

As discussed in Chapter 3, it is often convenient to group together a number of device faults which exhibit similar symptoms, and treat them as a subsystem failure. Unfortunately, this practice complicates the task of searching for conflicts between device failure states. When a cutset contains a device failure and a subsystem failure, each of the causes of the subsystem fault must therefore be checked against the conjugated fault. As before, if a conflict is detected, the grouped failure mode must be expanded into separate cutsets in order to delete the logical inconsistency.

The detection of conflicts between device failure modes and subsystem working conditions essentially involves the same procedure.

Finally, if a cutset contains a device failure mode in conjunction with either a spurious alarm fault or a passive failure of an alarmed indication (i.e. the alarm doesn't activate) the diagnosis procedure will again check for logical inconsistencies. Both of the latter two faults are similar to subsystem failures, in that they can be caused by a number of alternative fault scenarios, which are lumped
together in order to keep the alarm explanations concise. As before, if a logical inconsistency is detected, the original cutset is expanded into its valid constituent parts.

4.4.3 Conflicts With Device Working Conditions

The rules used to detect conflicts with device working conditions are exactly the same as described in Section 4.4.2.

4.4.4 Conflicts With Subsystem Failure Or Working States

Conflicts between different failure modes or between working and failure modes of the same subsystem, are detected using the same method as described in Section 4.4.2. Similarly, when one subsystem fault is compared against either a failure of a different subsystem, a subsystem working condition, a passive alarm failure or a spurious alarm failure, all the combinations of the causes of each composite failure mode are considered. If two mutually exclusive events are detected, then the cutset is again expanded into a number of separate cutsets.

The only additional check involving a subsystem fault entails comparing any boundary conditions within the composite failure mode with those of the rest of the cutset, and checking that these don't conflict with the assumed states of the alarmed variables.

4.4.5 Conflicts With Spurious and Passive Alarm Failures

In addition to the checks outlined previously, the diagnosis procedure also scrutinises each cutset for mutually exclusive spurious alarm failures, and conflicts between passive and spurious failures of the same alarm. Where the alarm failures do not pertain to the same process variable, the various causes of each alarm fault are examined.
If a process variable is monitored by more than one indication and hence has duplicate sets of high and low alarms etc., these sets of alarms can disagree. When this is the case, the alarmed process variable has a likelihood of being in a number of different states, depending upon which set of alarm instrumentation is thought to be at fault. Some of these variable states may conflict with other assumed states of the variable (within the boundary conditions and the other cutset elements) therefore these conflicts must be identified.

4.4.6 Conflicts Between Boundary Conditions

For obvious reasons, the boundary conditions associated with the alarm causes in each cutset are checked for mutual exclusivity. As described in Chapter 3, these boundary conditions can take two forms, representing the expectation that process variable will or will not be in a given state. Providing that the conditions refer to the same time period, a cutset will be rejected in the following four different circumstances:

1. If a process variable is expected to be in two different states, which are not adjacent in the discretisation, then the cutset is deemed to be logically inconsistent.

2. When one boundary condition specifies that a variable is expected to be in one state, and another condition specifies that it should not be in the same state, the cutset is also rejected.

3. A cutset is considered invalid if all of the variable states have been excluded by the boundary conditions.

4. If a boundary condition conflicts with an assumed state of an alarmed process variable then the cutset is rejected.
4.4.7 Conflicts Between Assumed States Of Alarmed Variables

Finally, a cutset is rejected if the cutset elements cause different states of the same alarmed variable. Similarly, if some faults lead to an unknown variable state (instrumentation failures) whilst others cause a defined variable deviation state, the cutset is determined to be invalid.

4.5 Simplifying the Cutsets

Many of the raw cutsets produced by conjugating the individual alarm fault trees can often be simplified in one of three ways:

1. If two alarm cause lists are combined because of their common failure modes, then at least one of the resulting cutsets will contain a duplicated failure mode. Providing that the two faults are assumed to occur during the same time window, the cutset can be simplified by removing the most recent of the duplicated elements.

2. When an undeveloped variable deviation appears within a cutset, the diagnosis procedure checks the list of variable deviation states caused by the other faults within the cutset. If a match is found between the input disturbance and one of the consequent variable deviations, then providing that the respective alarms coexist at the same point in time, the undeveloped fault is removed from the cutset. For example, let us reconsider a slightly more complex form of the heat exchanger system shown in Figure 4.10.

The modified section of plant is shown in Figure 4.12. In this case, the heat exchanger liquid outlet temperature T2 is controlled by manipulating the flow of steam into the shell side of the exchanger. The liquid inlet temperature t1 is also measured, and the steam line pressure p1 is monitored instead of the flowrate.
Figure 4.12 The Modified Heat Exchanger Example

The two fault trees for a low upstream pressure alarm and a low outlet temperature alarm are shown in Figures 4.13a and 4.13b.

Figure 4.13a The Causes Of A Low Steam Pressure Alarm

The two fault trees for a low upstream pressure alarm and a low outlet temperature alarm are shown in Figures 4.13a and 4.13b.
If the two alarms occur and are diagnosed in conjunction with each other, then the following sixteen cutsets will be generated:

a. Steam line leakage AND pressure P1 low

b. Steam line leakage AND control loop TIC1 fails low

c. Steam line leakage AND T1 -> low AND control loop TIC2 fails normal

d. Steam line leakage AND indication T12 spuriously alarms low

e. Upstream steam line blockage AND pressure P1 low

f. Upstream steam line blockage AND control loop TIC1 fails low

g. Upstream steam line blockage AND T1 -> low and control loop TIC2 fails normal
h  Upstream steam line blockage AND
indication T12 spuriously alarms low

i  Boiler shutdown and pressure P1 low

j  Boiler shutdown and control loop TIC1 fails low

k  Boiler shutdown and Tl => low AND
control loop TIC2 fails normal

l  Boiler shutdown AND indication T12 spuriously alarms low

m  Indication PIl spuriously alarms low AND pressure P1 low

n  Indication PIl spuriously alarms low AND
control loop TIC1 fails low

o  Indication PIl spuriously alarms low AND Tl => low AND
control loop TIC2 fails normal

p  Indication PIl spuriously alarms low AND
indication T12 spuriously alarms low

As can be seen from the above list, four of the sixteen
cutsets, namely numbers 1,5,9, and 13, include the undeveloped low
pressure deviation fault. However, in the case of cutsets 1,5,and
9, the other fault within each of the cutsets will cause the low
pressure deviation. Since the two alarm fault trees are
conjugated, this implies that the two alarms are causally coupled,
therefore the aforementioned cutsets can be simplified by removing
the undeveloped fault. Cutset 13 is rejected because it contains
an undeveloped variable deviation and a spurious alarm for the
same deviations, as discussed in Section 4.4.1. discussed in the
previous section.

3 The third situation in which a cutset can be simplified arises if
two or more elements within that cutset share a common failure
mode. For example, consider the second order cutset, subsystem X
fails high and device P fails high, which is generated when alarms A and B are diagnosed in conjunction with each other. If the subsystem fault is caused by either device Q fails high or device P fails high or device R fails high, then the original cutset actually represents the following three cutsets:

a. device P fails high and device P fails high
b. device P fails high and device Q fails high
c. device P fails high and device R fails high

Providing that the first cutset can be simplified by removing the duplicated element, the last two cutsets can be rejected because they are non-minimum.

A minimum cutset is defined as a set of faults which does not include another cutset as a subset. So for example, in the above list cutsets 2 and 3 contain cutset 1 as a subset, therefore they cannot be minimum cutsets. The reason why non-minimum cutsets are rejected is because they do not contain any additional information which will contribute to the diagnosis of the alarms. In the given example, if device P fails high, then from cutset 1 we can deduce that alarms A and B will activate, regardless of the state of devices Q and R.

Unfortunately, the cutset simplification process does not always produce such convenient reductions in size. For example, consider subsystem failure Y which describes faults: device P fails high, device S fails high and device T fails low. If this grouped failure mode is combined in a second order cutset with subsystem fault X (from the previous example), the following 5 cutsets will be produced after simplifying the result:
a  device P fails high
b  device S fails high AND device Q fails high
c  device S fails high AND device R fails high
d  device T fails low AND device Q fails high
e  device T fails low AND device R fails high

The common cause fault is clearly identified. However, the information originally represented by the single cutset now has to be represented in five separate cutsets.

4.6 Checking The Cutsets For Minimality

As described in the previous section, non-minimum cutsets only increase the number of explanations for a pattern of alarms, without contributing any additional information to the diagnosis. Consequently, when the KBS has simplified the raw cutsets, the whole list of cutsets is checked for minimality. This often reduces their number considerably. To illustrate the point, consider the sixteen cutsets produced by conjugating the low steam pressure and low liquid outlet temperature alarm fault trees shown in Figures 4.13a and 4.13b. Of the original cutsets, only the following six are minimum:

1  Steam line leakage
2  Upstream steam line blockage
3  Boiler shutdown
4  Indication PI1 spuriously alarms low AND control loop TIC2 fails low
5 Indication PI1 spuriously alarms low AND temperature T1 ⇒ low AND control loop TIC2 fails normal

6 Indication PI1 spuriously alarms low AND indication T12 spuriously alarms low

The process of determining the minimum cutsets first involves tagging each cutset with a number which represents its size or order. The smallest cutset is then found from the list and the remainder are examined to determine if they contain this cutset as a subset. Once the smallest cutset has been compared against all the other cutsets, the process is repeated with the next smallest cutset, and so forth until all the cutsets have been checked against each other.

The task of reducing a fault tree structure into its minimum cutset form is frequently performed during risk analysis studies, and consequently many computer codes have been written to automate the process. Two well known examples are FTAP [44] and PREP [45].

Whilst it would have been desirable to use one of these existing programs within the diagnostic process, none were actually utilised. This was because of two main problems:

1 It was estimated that considerable programming effort would have been required to interface the external code to the KBS software, because of the difficulty of transferring cutset information to and from the PROLOG language environment.

2 The existing programs did not have the ability to recognise both logical inconsistencies and simplifications within the cutset information.
Chapter 5

RANKING THE INDIVIDUAL ALARM CAUSES

Chapter 4 describes the method used to derive the explanations for an alarm or group of related alarms. This chapter discusses the method used to prioritise the individual explanations before they are displayed to the operator.

5.1 Reasoning With Uncertainty

In order to assess the relative importance of alternative alarm explanations, it is necessary to use a factor which describes the degree of belief, or the certainty, of the considered hypotheses being true. Whilst numerous theories abound concerning the treatment of certainty there does not appear to be a consensus of opinion on the matter, in fact quite the reverse, as discussed by Leitch [47]. Consequently many KBS use certainty in an ad-hoc fashion, simply to obtain an answer.

There is perhaps some justification in this approach, since humans in an everyday context seldom reason using a well defined theory of certainty. However the approach here has been to work on a sound theoretical basis where possible. Leitch [47] discusses some of the problems involved and the different representations used within KBS's.
When the explanations are initially evaluated, the actual state of the process equipment is usually unknown, so it is therefore necessary to use some 'a priori' measure of the likelihood of the process units being in a failed state. This is similar in principle to an operator using his experience of equipment reliability when diagnosing the cause of a set symptoms (process indications and alarms).

The reliability of process equipment is available from the literature and failure data banks, as discussed in Section 5.8, but is usually in terms of the frequency of the equipment spontaneously failing, rather than the probability of it being in a failed state.

Probabilistic information is easier to logically manipulate compared with failure rate data. However, alarms are spontaneous events, and as such it is difficult to relate these outcomes back to the probability of the process equipment spontaneously changing into a state which causes them. Because the failure rate information is available and is a better means of describing the likelihood of spontaneous events, the method used to rank alternative alarm explanations within the KBS is based on this approach. Probabilistic information does, however, have an important part to play in the ranking procedure, as will be discussed later in the chapter.

The likelihood of each explanation for an alarm or group of related alarms is therefore calculated using Equation 5.1:

\[ L_i = \frac{X_i}{\sum_{j=1}^{n} X_j} \]  

(5.1)

where \( L_i \) is the likelihood of explanation \( i \)
\( X_i \) is the frequency of explanation \( i \)
\( n \) is the number of alternative explanations

The remainder of the chapter describes how the frequency of the different types of potential causes of an alarm are evaluated.
5.2 Frequencies Of Single Faults

The failure rate of each fault that is considered within the fault propagation models should be defined within the rulebase. In addition to the faults frequency, a number of other attributes also need to be specified. The first of these is the fault type. This can be either of the initiating or enabling variety.

Enabling faults are usually associated with protective and control equipment. They do not themselves cause a fault, but prevent it from being corrected. An example would be a control loop's sensor failing invariant in its normal state. Initiating faults are those which initially cause a process disturbance, which if unchecked might cause an undesired plant state. An example of an initiating fault would be a pipe rupture causing a loss of fluid.

The second attribute that is required is the time period for which the process equipment will be out of service following its failure. This will obviously depend on the alertness of the operators, the plant maintenance policy, the availability of spare parts and the level of redundancy of process equipment on the plant. For an initiating fault, this time may be dominated by the repair time; for an enabling fault the detection time may have a significant effect. The expertise of the process operators, fitters and plant management is therefore required if an accurate figure is to be obtained.

The presence of all the data is checked when the rulebase is compiled and the user notified of any omissions.

5.3 Frequencies Of Fault Combinations

Whilst many alarms can be caused by a single spontaneous event, some will require a combination of failures to occur before they become active. Unlike probabilities, the frequency of a combination of failures cannot be calculated from the product of the individual frequencies (not least because the result would have meaningless
dimensions). Instead all the frequencies except one must be converted into probabilities, and on multiplication a resulting frequency is obtained.

The method of converting a frequency into a probability depends on the type of failure being considered, whether it is an initiating or enabling fault. These two cases are now considered.

5.3.1 Probabilities Of Enabling Faults

The probability of a control loop or protection system being in a failed state can be estimated from the frequency data, if the out of service time period is known. If the detection and repair time for an enabling fault is $T_{dr}$ and the failure frequency is $f$, then $1/f$ represents the expected time between faults. The fraction of the time, or probability, that the control loop or protective system is not working is therefore given by Equation 5.2:

$$P_f = \frac{T_{dr}}{T_{dr} + \frac{1}{f}}$$  \hspace{1cm} (5.2)

In most cases $T_{dr}$ is significantly less than $1/f$, therefore Equation 5.2 simplifies into Equation 5.3:

$$P_f = T_{dr} \times f$$  \hspace{1cm} (5.3)

5.3.2 Probabilities Of Initiating Faults

Where an alarm or group of alarms is caused by multiple initiating events, the frequency of the first initiating event is multiplied by the probabilities of other faults occurring, whilst the first and subsequent faults are still active.
As an example consider faults A and B. Failure A has a frequency of 0.1 events/yr. and would take one day to detect and repair. Fault B occurs at the rate of 2 events/yr. and would take one hour to detect and repair. If fault A occurs first then the probability of fault B taking place whilst fault A is still unrepaired is calculated as follows:

given that fault B occurs at the rate of twice per year, the probability of it failing in one day is:

\[
\frac{1}{365} \times 2 = 0.005479
\]

The frequency of the combined event A then B is therefore:

0.005479 \times 0.1 = 0.0005479 events/yr.

However, the frequency of the initiating faults occurring in the reverse order is different. In this case:

\[
\frac{2 \times 0.1}{24 \times 365} = 0.00002283 \text{ events/yr.}
\]

Within the development KBS the order of any initiating events can only be resolved to the time scan in which the resultant process deviation was detected. Furthermore, because of the difficulties in estimating the fault propagation time delays (outlined in Chapter 4) the faults are assumed to occur during the same time scan in which the resultant alarms are detected. Because of these restrictions and assumptions, there may appear to be no clear first fault within a cutset.
For example, it may not be uncommon for two related alarms to be detected during the same scan interval (this of course depends on the frequency of alarm checking). Because of the possible instrumentation failures, some of the resulting cutsets will contain independent faults which have notionally occurred during the same scan interval. When this is the case, the alternative fault ordering scenarios are evaluated and the results combined in a weighted average.

Each weighting factor effectively represents the probability that the faults occurred in a certain order. The theory behind the derivation of these weighting factors is outlined below:

Consider two faults X and Y which have failure rates of $f_x$ and $f_y$ respectively. If the scan interval is $T_s$, then fault X may occur during some fraction of $T_s$, termed $T_o$. The probability of this being the case is $f_xT_o$.

In order for fault X to occur first, fault Y must not occur during time $T_o$. The probability of X occurring and Y not occurring in time $T_o$ is therefore:

$$f_xT_o(1 - f_yT_o) \quad (5.4)$$

Similarly the probability of fault Y occurring and X not occurring is:

$$f_yT_o(1 - f_xT_o) \quad (5.5)$$

One of the two faults must have occurred first, therefore the normalised probability of fault X occurring first is as shown below:

$$\frac{f_xT_o(1 - f_yT_o)}{f_xT_o(1 - f_yT_o) + f_yT_o(1 - f_xT_o)} \quad (5.6)$$

If we assume that $f_x$ and $f_y$ are small numbers, as is usually the case, then the product of $f_x$ and $f_y$ will be insignificant compared to either
fx or fy. Consequently, the probability of fault X occurring first or fault Y occurring first simplifies to Equations 5.7 and 5.8 respectively:

\[
\begin{align*}
\frac{fx}{P_{xy}} &= \frac{fx}{fx + fy} \quad (5.7) \\
\frac{fy}{P_{yx}} &= \frac{fy}{fx + fy} \quad (5.8)
\end{align*}
\]

Returning to the previous example, if events A and B are assumed to have taken place in the same time scan, then the resultant frequency of the cutset would be evaluated as:

Frequency of AB = 0.0005479 f/yr.

Frequency of BA = 0.00002283 f/yr.

\[
f = \frac{0.1}{2 + 0.1} \times 0.0005479 + \frac{2}{2 + 0.1} \times 0.00002283
\]

\[
= 0.00004783 \text{ f/yr.}
\]

Cutsets containing more than two initiating events can be considered using exactly the same theory. In this case the common down time for all the preceding faults is used to calculate the probabilities. To illustrate the method consider the alarm explanation which contains the three initiating events A, B and C. A summary of the time history and detection and repair times is shown overleaf in Figure 5.1.
Figure 5.1 The Failure History Of Events A, B and C

A | *<-------------------DRTa----------------->|
   |                                       |
B | *<-------------------DRTb----------------->|
   |                                       |
C | *<-------------------DRTc----------------->|
   |                                       |
   | <-----DRTabc----->|
   |         |

Sample Intervals

where DRTi is the detection and repair time for fault i
* denotes the scan interval during which the resultant alarms were detected.

As can be seen from Figure 5.1 the three events are assumed to occur during different scan intervals. Because the time scan is the minimum resolvable unit of time the detection and repair times DRTa, DRTb and DRTc are also in terms of the nearest number of scan intervals.

The time frame during which it is possible for all three faults to coexist is given by DRTabc. The probabilities of events B and C are therefore calculated from their failure frequencies and multiplied by this common time period. If faults A and B had been detected during the same time scan, the two alternative orders for the three events would have been considered, namely:
A then B then C or

B then A then C

The same rules would then apply in calculating the event probabilities.

5.4 Frequencies Of Variable Deviations

Monitored variable deviations are an important constituent of the fault propagation models because they allow complex plants to be broken down into more manageable units. They also provide a means of refining a particular diagnosis because the monitored variable states are by definition observable.

As stated previously, the likelihoods of alternative alarm explanations are calculated from their frequencies of occurring. Therefore when a process variable deviation is being considered as a cause, the frequency of the event is required.

This frequency measure has to reflect the historical experience of the variable being in its deviation state, analogous to the failure frequency of process equipment. However, since the variable is by definition observable, the actual state of the variable (when the alarm was detected) can also be taken into consideration.

These two factors are taken into account by multiplying a base frequency for the event occurring, by the likelihood that the variable was in its deviation state when the fault was detected. The derivation of these two figures is discussed in the following sections.
5.4.1 Calculation Of The Base Frequency

The base frequency for each process variable deviation is estimated when the rulebase is compiled from all the causes of that variable deviation, except other variable deviations. The variable deviation causes are excluded for two reasons:

1. The plant state will be unknown.

2. The base frequencies of the other deviations might not have been calculated. The alternative approach of retaining the intervariable dependencies could only be solved by iteration, and then would be time consuming, if at all possible. The base frequency is therefore based on an 'a priori' measure of the frequency of the variable deviation.

In many situations this approach yields a reasonable base failure frequency. However, when the causes of an alarm are predominantly other variable deviations, the technique is less applicable. This problem has scope for further investigation.

5.4.2 Interpreting The Monitored Variable Indications

The task of diagnosing a process plant fault relies heavily on the indicated values of the monitored variables. For this reason care must be taken to ensure that the indications are interpreted correctly, given that instrument failures are common and often cause confusion, as discussed by Andow [25]. The situation can become even more complex when a variable is monitored by multiple indications. The redundancy often enables the variable state to be predicted with greater confidence, unless the indications disagree.

The problems of validating the integrity of instrumentation equipment have been considered before by Anyakora and Lees [48] and Bellingham and Lees [49]. The emphasis of this work has been towards the analysis of the indication signal. The approach that is used within the development KBS however, is not based on this previous
work. Instead the indications are interpreted using a more probabilistic method. Despite this, the alternative techniques are not incompatible because any additional information gained from the signal analysis could be effectively used within the probabilistic method, as discussed in Chapter 9.

The degree of confidence that a process variable is in a given state is based on two factors, the reliability of the instruments and the indicated values. The method of interpreting an indication is therefore a two stage process. Firstly, the indications have to be discretized into a logic state, and then the potential faults of the instruments are taken into account. These two independent stages are described in the following sections.

5.4.3 Continuous Variable Discretisation

The information flow models chosen to represent the propagation of process disturbances are necessarily simplistic. The major problem that is encountered when applying them to real systems is interpreting the meaning of the discretized states high, normal and low etc..

The simplest method of assigning a state to a process variable is to divide the range of that variable into a number of discrete intervals. This is in effect how most process variables are treated in alarm systems. If the variable crosses an alarm threshold it is assumed to be in that alarmed state, otherwise it is normal. Whilst the approach seems to fit in well with the treatment of alarms and is computationally inexpensive, it is unsuitable for alarm diagnosis for the following reasons:

1. The discretisation is not very meaningful in certain circumstances. For example, if a variable has a value which is close to the boundary between two state intervals, the assertion that the variable is definitely in one state and not the other does not represent the situation accurately.
Another complication arises because 'steady state' process variables often fluctuate slightly about a mean value. If the mean value is approaching an interval boundary then the discretized state will change at twice the frequency of the process noise. Using this information to assess the relative merits of a number of alternative explanations for a fault condition would prove difficult.

To overcome these limitations the basic concept of fuzzy intervals, proposed by Zadah [50] has been adopted. Instead of using a crisp set of mutually exclusive states, a process variable is now allowed to exhibit a membership of more than one state. Figure 5.2 illustrates three fuzzy membership functions for the states low, normal and high. When the variable has a value of 0.6, its set of fuzzy membership functions is low:0.3, normal:0.7 and high:0.0.

Figure 5.2 Fuzzy Logic Membership Functions

The major benefit of using fuzzy intervals is that the fault propagation models can remain relatively simple and yet be used more meaningfully. When the likelihoods of each explanation for a fault condition are evaluated, the fuzzy membership functions can be used as
a measure of the strength of the process deviations. As a consequence, the ranking of the alternative solutions will be more sensitive and yet less prone to process noise.

The drawback of the technique is that more information is required to define a fuzzy interval. Within the development KBS, the limitation has been imposed that an indication will only exhibit a non-zero membership function of two discrete states simultaneously. For simplicity, the fuzzy envelopes are also defined with straight line boundaries.

The problems associated with choosing the fuzzy discretisation envelopes are discussed in more detail in Chapter 7.

5.4.4 Modelling Instrumentation Faults

Whilst the fuzzy envelopes enable continuous variables to be discretised into multivalued logic states, they assume that the sensing equipment is in working order. Clearly this will not always be the case, especially when there are conflicting indications of the same process variable. The KBS therefore modifies the raw variable discretisations to take account of potential instrumentation faults.

In order to evaluate the impact and probability of instrumentation failures, the indication interpretation method needs a model which describes the failure modes of every indication. Within the KBS this is achieved by constructing an If-Then rule which lists all the alternative reasons why an indication is reading its current value. For example, consider the simple instrumentation system shown in Figure 5.3.
The indication pil will output a signal of X pressure units if either of the following two scenarios are true:

1. The process pressure pl is as indicated (within the instruments tolerance)

2. The pressure sensor has failed in state X and the plant pressure is in an unknown state

Initially a single generic rule was used to describe all the reasons for an indication being in any of its discretised states. In the case of the pressure indication this would have taken the form:

```
rule 2:
    if
        device ps_1 fails X
    or
        pressure pl reads X
    then
        indication pi_1 reads X.
```

However, after considering a number of process plants in more detail it was realised that the generic approach would not adequately describe all the failures modes of certain configurations of
instrumentation. For example, if an isolation valve is placed between pressure sensor ps_l and the process plant, to facilitate maintenance of the transducer, the causes of the pressure sensor failing invariant (normal) will include the accidental closure or blockage of the valve. In this case the normal process pressure will effectively be locked behind the valve. The fault 'device valve fails closed' should therefore be considered as a cause of the sensor erroneously reading normal when the probabilities of the pressure being in deviation are being evaluated. The failure of the isolation valve will not, however, cause the pressure indication to erroneously read high or low.

To overcome this problem, without considering the valve as part of the pressure sensor, the causes of each discretized state of every indication are now represented within a separate rule. For example, the two rules for high and normal pressure indications, when an isolation valve is present, are listed below:

**rule 3** :

if

device ps_l fails high
or
pressure pl reads high
then

indication pi_l reads high.

**rule 4** :

if

device ps_l fails normal
or
device valve_l fails closed
or
device valve_l fails blocked
or
pressure pl reads normal
then

indication pi_l reads normal.
If it is assumed that the instrument rules describe all the reasons why an indication is reading a given state, the probability of the alternative scenarios can be evaluated by multiplying their frequencies of failure by their detection and repair times. The probability of the sensor working is calculated as unity minus the sum of the probabilities of the failure modes. Because the existing state of the instrumentation is being evaluated, rather than its change of state from working to failed, it is more appropriate to use probabilities than frequencies of failure.

For example, referring back to Figure 5.3, consider the situation where pressure indication pil is reading a high pressure signal. If pressure sensor ps_1 has a failed high frequency of 0.7 faults per million hours and a detection and repair time of one hour, as reported in [51], the probability of it being in a failed high state is \(0.7 \times 10^{-6}\) (as calculated using Equation 5.3). The probability of pressure pl being in an unknown state is therefore \(0.7 \times 10^{-6}\) and in a high state \(1 - 0.7 \times 10^{-6}\) i.e. \(0.9999993\).

The indicated value of pressure pl can be discretised into a set of three fuzzy membership functions as follows:

\[ \{ \text{low}: X_L, \text{normal}: X_N, \text{high}: X_H \} \]

However, the unknown state of the pressure can only be represented by the set of three unknown fuzzy membership functions:

\[ \{ \text{low}: U_L, \text{normal}: U_N, \text{high}: U_H \} \]

If the probabilities of the two different interpretations of the indication are used to weight the fuzzy membership functions, the following relationships can be defined for the membership functions of the actual pressure pl:
low: \( X_L \times 0.9999993 + U_L \times 0.7 \times 10^{-6} \)

normal: \( X_N \times 0.9999993 + U_N \times 0.7 \times 10^{-6} \)

high: \( X_H \times 0.9999993 + U_H \times 0.7 \times 10^{-6} \)

In the more general case when there are \( K \) interpretations of an indication, the membership function for discretised state \( i \) can be calculated using Equation 5.9:

\[
M_{fi} = \sum_{j=1,K} P_j \times X_{ij}
\]

(5.9)

Where

- \( M_{fi} \) is the membership function of state \( i \)
- \( P_j \) is the probability of interpretation \( j \)
- \( X_{ij} \) is the membership function of state \( i \) in interpretation \( j \)

Clearly the membership functions of a process variable can only be determined if some assumptions are made about the 'unknown state' membership functions. For the purposes of the research project two alternative approaches were considered:

1. Assume that the unknown variable state is the same as the state for which the probability is being determined. For example, if a high pressure \( p_l \) is hypothesised as a cause of another variable deviation, the KBS will use the indication interpretation function to determine the probability of pressure \( p_l \) being high. In this situation the unknown variable state could be assumed to be high, regardless of the indicated value;

2. In the absence of any additional information, assume that there is an equal chance of the process variable being in any of its discretised states.

From a numerical point of view the two methods yield very similar results, because in most cases the probability of an indication being in a failed state is small. The relative validity of the assumptions was therefore deemed to be the most important issue.
It is acknowledged that the assumption of a process variable having an equal chance of being in any of its discretised states is somewhat unrealistic. Furthermore the assumption implies that the plant is only operating normally for a fraction of the time. However, once the KBS was installed on a process plant, the default set of unknown membership constants could be updated with the actual time averaged membership function constants.

The second approach also has two other advantages over the first, namely:

1 When the probability of process variable being in state X is being considered and the indication is also in state X, a probability of unity will not be returned, since some allowance is made for the variable to be in other states.

2 The set of membership functions can be calculated once and will remain the same regardless of the state for which the probability is being determined. This will be computationally more efficient.

For the above reasons the second approach was adopted within the KBS. The unknown state membership constants are therefore calculated as follows:

\[(state(1): l/n, state(2): l/n, \ldots, state(n): l/n)\]

As mentioned earlier, this set could then be updated at a later stage with plant specific information.

Returning to the original example, in the case of the low state of pressure p1, the failure weighted membership function will evaluate to:

\[0.7 \times 10^{-6} \times 0.3333 + 0.9999993 \times X_L\]

The major drawback of using a separate indication rule for each discretised variable state, is that when the indication is discretised in the transition region between two states, two rules are applicable.
In this situation the KBS selects the rule for the state with the highest membership function, unless one of the states is 'normal'. This exception is made because the faults which will cause an indication to fail normal (invariant), such as a closed isolation valve, are assumed to only cause the indication to read in a state which will be discretised as 100% normal. In the transition region between normal and either high or low, the non-normal indication rule is therefore selected.

5.4.5 Multiple Indications Of The Same Variable

When the same process variable is monitored by more than one indication, as might be expected on hazardous plant, interpreting the indications becomes a little more complex. This is essentially because the method has to take account of all the indicated values and instrumentation failure mechanisms. To illustrate the point consider the following example system:

Figure 5.4 Duplicate Indications Of A Process Variable

```plaintext
Pressure Indicator pi1
Pressure Sensor ps_1
Process Pressure p1

Pressure Indicator pi2
Pressure Sensor ps_2
Process Pressure p1
```
The pressure is monitored by two independent sensors $ps_1$ and $ps_2$ which indicate values of X and Y pressure units respectively. Following the discretisation process the two sets of fuzzy membership functions which describe the indications are:

\[ pil ( \text{high} : P_{IXH}, \text{normal} : P_{IXN}, \text{low} : P_{IXL} ) \]

\[ pi2 ( \text{high} : P_{IYH}, \text{normal} : P_{IYN}, \text{low} : P_{IYL} ) \]

In order to interpret the two indications, the individual causes of them being in their respective states have to be considered in combination. For the example system, because there are only two devices which can be either working or failed, there are four second order cutsets which describe all the possible interpretations of the situation, namely:

1. Pressure $p_1$ is in state X AND pressure $p_1$ is in state Y
2. Sensor $ps_1$ has failed in state X AND pressure $p_1$ is in state Y
3. Pressure $p_1$ is in state X AND sensor $ps_2$ has failed in state Y
4. Sensor $ps_1$ has failed in state X AND sensor $ps_2$ has failed in state Y

As with the alarm explanations, on combining the individual causes it is possible to generate logical inconsistencies within the cutsets. For example, if the values of the indications X and Y disagree by more than the normal error range then both the pressure sensors cannot be working correctly simultaneously. In this situation, the first explanation cutset will therefore be invalid.

The KBS avoids logical inconsistency by pre-checking every indication of the same variable to determine if they are in 'agreement'. In order to decide if two indications are in agreement the difference between the indicated values is compared against that which can be explained in terms of instrument error. For example, if
The maximum error expected in any indication of variable Q is plus or minus \( E_q \), any two indications of Q, e.g. A and B, can be in different by the amount specified by Equation 5.10, without either of the instruments being suspected as faulty:

\[
\frac{A \times E_q + B \times E_q}{100}
\]

(5.10)

The instrument engineer's expert knowledge is therefore required to decide what the maximum permissible error in an indication should be. This is then stored in the knowledge base for each monitored variable.

If the related indications are deemed to be in agreement, the KBS crudely reconciles them to a single value, by calculating their arithmetic average. The indications rules are then selected on the basis of this reconciled value. For indications \( p_{1l} \) and \( p_{2l} \) the reconciled reading would be calculated as:

\[
\frac{X + Y}{2}
\]

Which would then be discretised as

\[ Z \text{ ( high : } P_{1Z_H}, \text{ normal : } P_{1Z_N}, \text{ low : } P_{1Z_L} ) \]

Based on the values of \( P_{1Z_H}, P_{1Z_N} \) and \( P_{1Z_L} \), the appropriate indication rules would then be selected using the following criteria:

1. The 'high' rules if \( P_{1Z_H} > 0.0 \)
2. The 'low' rules if \( P_{1Z_L} > 0.0 \)
3. The 'normal' rules if \( P_{1Z_N} = 0.0 \)
If the 'high' rules were selected the following explanations cutsets would be generated:

1 Pressure pl is high AND pressure pl is high
2 Sensor ps_1 has failed high AND pressure pl is high
3 Pressure pl is high AND sensor ps_2 has failed high
4 Sensor ps_1 has failed high AND sensor ps_2 has failed high

When the related measurements do not agree, at least one of the indications is always considered to be in error. The indication rules for each measurement are therefore selected on the basis of their separate values, and logically combined as before. For example, consider the following discretisations of indications pil and pi2.

$pil$ (low: 0.0, normal: 0.5, high: 0.5)

$pi2$ (low: 0.0, normal: 0.2, high: 0.8)

Assuming that the indications are too far apart to be in agreement the KBS will pick separate indication rules to suit each measurement. In this case the 'high' rules would be selected in both cases.

When the rules are logically conjugated the following explanations will be generated:

1 Pressure pl is high AND pressure pl is high'
2 Sensor ps_1 has failed in high AND pressure pl is high'
3 Pressure pl is high AND sensor ps_2 has failed high'
4 Sensor ps_1 has failed in high AND sensor ps_2 has failed high'
Although the 'high' rules are used for both indications, the two measurements are still deemed to be in disagreement. As a consequence the first cutset will be deleted as logically inconsistent.

Finally, when the indication cutsets have been checked for consistency and simplified, the valid minimum cutsets are evaluated using the same method as described previously. The two alternative sets of fuzzy membership functions therefore evaluate to:

**Agreeing Indications**

\[
\begin{align*}
W_L &= P(ps_1, Z)P(ps_2, Z)0.33^* + (1 - P(ps_1, Z)P(ps_2, Z))Z_L \\
W_N &= P(ps_1, Z)P(ps_2, Z)0.33^* + (1 - P(ps_1, Z)P(ps_2, Z))Z_N \\
W_H &= P(ps_1, Z)P(ps_2, Z)0.33^* + (1 - P(ps_1, Z)P(ps_2, Z))W_H \\
\end{align*}
\]

**Disagreeing Indications**

\[
\begin{align*}
W_L &= P(ps_1, X)Y_L + P(ps_2, Y)X_L + P(ps_1, X)P(ps_2, Y)0.33^* \\
W_N &= P(ps_1, X)Y_N + P(ps_2, Y)X_N + P(ps_1, X)P(ps_2, Y)0.33^* \\
W_H &= P(ps_1, X)Y_H + P(ps_2, Y)X_H + P(ps_1, X)P(ps_2, Y)0.33^* \\
\end{align*}
\]

where

\[
\begin{align*}
P(ps_1, I) &= \text{probability of device } ps_1 \text{ failing in state } I \\
P(ps_2, I) &= \text{probability of device } ps_2 \text{ failing in state } I \\
W_I &= \text{interpreted membership function for state } I \\
\end{align*}
\]

When a process variable is monitored by one or two indications it is only possible to interpret a set of indicated values in one way. They can either agree or disagree. However, if there are three or more indications of the same variable, the same set of indicated values can, in certain situations, be interpreted in more than one way. For example, consider variable R which has three indications X, Y and Z.
Because of random and systematic errors in the instruments, the three indications will invariably read differently. Figure 5.5 represents an example spread of the three readings:

**Figure 5.5 The Spread Of Indications Of Variable R**

```
---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---*---...
perfectly possible for X and Y and Y and Z to be in agreement simultaneously. If this is the case, the following two interpretations of the situation are valid:

1 Indications X and Y are indicating variable R being in state 0.5*(X + Y) AND Indication Z is indicating variable R being in state Z.

2 Indications Z and Y are indicating variable R being in state 0.5*(Z + Y) AND Indication X is indicating variable R being in state Z.

The indication rules selected in either scenario may be different depending on the values of 0.5*(X + Y) and 0.5*(Z + Y). Both situations therefore need to be evaluated.

The KBS handles this potentially complex situation by cross checking each indication as described previously. For the case where there are N measurements of the same variable, this involves C_{bin} binary checks, where C_{bin} is defined by Equation 5.11:

$$C_{bin} = \frac{N(N - 1)}{2}$$  \hspace{1cm} (5.11)

For each set of C_{bin} true or false values, the possible interpretations of the indications are retrieved from a general lookup table. Because the number of lookup table entries is equal to two to the power of C_{bin}, only eight different entries are required for three indications, but sixty four are required for four. The KBS has therefore only been designed to model three indications of the same variable. It was envisaged that this would be more than sufficient for most applications, however, the code was written to easily allow this limit to be extended.

Finally, the cutsets explaining each mutually exclusive interpretation of the set of indications are derived by the KBS. These are then added, and evaluated as normal to yield a single set of fuzzy membership functions for the process variable.
5.4.6 Common Instrumentation Devices

The method of indication interpretation described in the previous section can contend with multiple indications of the same process variable and multiple instrumentation devices in series from the process variable to the final indication. The most important shortcoming of the approach is its inability to correctly model common instrumentation devices, shared between different indication paths.

To illustrate the problem consider a third example system of indication devices shown in Figure 5.6.

Figure 5.6 Common Instrumentation Paths

```
Both the indications are derived from a single pressure sensor ps_1, but then the signals are processed by separate transducers tr_1 and tr_2. The causes of the separate indications pil and pi2 being in their respective states are now:
$p_{11}$ indicates $X$ pressure units if

1. Pressure $p_1$ is in state A
2. Transducer $tr_1$ has failed in state A
3. Sensor $ps_1$ has failed in state A

$p_{12}$ indicates $Y$ pressure units if

1. Pressure $p_1$ is in state B
2. Transducer $tr_2$ has failed in state B
3. Sensor $ps_1$ has failed in state B

On combining the causes nine explanations result, namely:

1. Pressure $p_1$ is in state A AND pressure $p_1$ is in state B
2. Pressure $p_1$ is in state A AND Transducer $tr_2$ failed in state B
3. Pressure $p_1$ is in state A AND sensor $ps_1$ failed in state B
4. Transducer $tr_1$ has failed in state A AND pressure $p_1$ is in state B
5. Transducer $tr_1$ has failed in state A AND Transducer $tr_2$ failed in state B
6. Transducer $tr_1$ has failed in state A AND sensor $ps_1$ failed in state B
7. Sensor $ps_1$ has failed in state A AND pressure $p_1$ is in state B
Sensor ps_1 has failed in state A AND transducer tr_2 failed in state B

Sensor ps_1 has failed in state A AND sensor ps_1 failed in state B

When closely inspected it can be seen that explanations 3 and 7 are incorrect, regardless of the values of X and Y (or A and B). In both cases one part of the explanation assumes that all the devices in the indication path are working, whilst the other states that one of the devices is in a failure state.

The only way to overcome this problem is to explicitly state in the indication rules when the instrumentation devices are assumed to be working in addition to stating when they are at fault. This is implemented within the knowledge base in terms of the "does_not_fail" prolog operator. For example, the rule for a high indicated pressure (monitored via pil) is as follows:

rule 5:
if
    pressure pl reads high and
    device tr_1 does_not_fail high and
    device ps_1 does_not_fail high
or
    device tr_1 fails high
or
    device ps_1 fails high
then
    indication pil reads high.

Note that only the sub clause that represents the correct operation of the instrumentation equipment requires the working states to be specified.
By including the working states into the explanations the anomalies can easily be detected:

1 Pressure p₁ in state A AND sensor ps₁ is working AND transducer tr₁ is working AND sensor ps₁ failed in state B

2 Sensor ps₁ has failed in state A AND pressure p₁ is in state B AND sensor ps₁ is working AND transducer tr₂ is working

The check for conflicting states of the same device is performed during the internal consistency check. If the indicated values of X and Y are different, explanation 9 will also be rejected on the same grounds. At this point the working state information is disregarded.

5.5 Frequencies Of Grouped Failure Modes

Grouped failure modes are used to reduce the total number of alarm explanations by representing a number of faults as one failure mode. Each grouped failure mode is effectively a small fault tree in its own right, and as such can be treated in exactly the same way as an alarm fault tree. The frequency of a grouped failure mode is therefore calculated as the sum of the frequencies of the faults it represents.

For the reasons discussed in Section 5.3, it is also necessary to calculate an effective detection and repair time for a grouped failure mode. The derivation of the theory used to estimate group detection and repair times is as follows:

Consider a grouped failure mode A which represents four faults W, X, Y and Z. If A is conjugated with fault B, the frequency of the result (if A precedes B in time) can be calculated as shown overleaf:
\[ F_A \times (F_B \times DRT_A) \]  \hspace{1cm} (5.12)

where \( F_A \) is the frequency of fault A

\( F_B \) is the frequency of fault B

\( DRT_A \) is the detection and repair time of fault A

The term in parentheses evaluates to the probability of fault B occurring whilst fault A is still active.

If the four failure modes \( W, X, Y \) and \( Z \) had not been grouped together, the same situation would have been described in terms of four cutsets, the total frequency of which would evaluate to:

\[ F_W \times F_B \times DRT_W + F_X \times F_B \times DRT_X + F_Y \times F_B \times DRT_Y + F_Z \times F_B \times DTR_Z \]  \hspace{1cm} (5.13)

Clearly Equation 5.12 should equate to Equation 5.13, therefore on simplification Equation 5.14 will result:

\[ F_A \times DRT_A = F_W \times DRT_W + F_X \times DRT_X + F_Y \times DRT_Y + F_Z \times DTR_Z \]  \hspace{1cm} (5.14)

Since \( F_A \) is equal to \( F_W + F_X + F_Y + F_Z \), the detection and repair time for the grouped failure mode A can be solved as:

\[ DRT_A = \frac{F_W \times DRT_W + F_X \times DRT_X + F_Y \times DTR_Y + F_Z \times DTR_Z}{F_W + F_X + F_Y + F_Z} \]  \hspace{1cm} (5.15)

As might be expected the result is in fact a frequency weighted average of the individual detection and repair times.
5.6 Frequencies Of Spurious Alarm Failures

When an alarm activates there are two possible causes, either the process variable is in a deviation state or the instrumentation has failed in a misleading state. The frequency of the latter is calculated from the cutsets generated during the indication interpretation technique.

To illustrate the method, reconsider the instrumentation shown in Figure 5.4. If indications pil and pi2 are in agreement, the alarm detection program will only generate their alarms when the average of the two indications crosses an alarm threshold. In this case the spurious failure of both alarms will be considered together. Otherwise their spurious failures will be considered separately.

The indication interpretation subroutine will then be called to interpret the state of pressure pl. The resultant set of fuzzy membership functions will be ignored, however. Instead the explanation cutsets are screened to remove any interpretations which do not involve the alarmed indication(s) being in a failed state.

Returning to the example, if indications pil and pi2 are both alarmed low, the following list of cutsets will be generated by the indication interpretation program:

1. Pressure pl is in state low AND pressure pl is in state low
2. Sensor ps_1 has failed in state low AND pressure pl is in state low
3. Pressure pl is in state low AND sensor ps_2 has failed in state low
4. Sensor ps_1 has failed in state low AND sensor ps_2 has failed in state low

The objective of the KBS is to determine the frequency of both indications spuriously alarming, therefore only those cutsets which
include failure modes causing both indications to be misleading are retained from the above list. In this example, only cutset number 4 is retained. The frequency of all the retained cutsets are then evaluated and summed.

5.7 Probabilities Of Passive Alarm Failures

If an indication is in an alarmed state but a related measurement is not, e.g. pil has alarmed high but pi2 reads normal, all the causes of the alarmed variable deviation are multiplied by the probability of the non-alarmed indication passively failing.

The method used to evaluate the probability of an indication passively failing is essentially the same as that described in the previous section. The indication interpretation subroutine will have already been called to calculate the spurious failure frequency of the active alarm. The same raw cutsets are then screened to remove those failure modes which will not have caused the given indication to passively fail.

The probabilities of the remaining cutsets are calculated by multiplying the frequency by the detection and repair time of each cutset element. The results are then logically ORed.

5.8 Failure Rate Data Sources

The process equipment failure rate data is important to the fault diagnosis methodology described in thesis. Having said this, in the case of the two example applications of the KBS, the absolute accuracy of the failure rate information has not been a major consideration. Instead the emphasis of the research work has been to investigate the problems of systematically generating and using the fault propagation information for use in real-time alarm diagnosis. The purpose of this section is therefore to discuss some of the problems associated with obtaining the failure rate data, and to explain the mitigating factors.
Although there is a vast resource of process equipment world-
wide from which to draw accurate and reliable failure rate data, in
reality such information is often scarce and of dubious applicability. This is principally because the reliable reporting of process
equipment faults is not a common practice within many of the process
industries. This is because such reporting is either not deemed to be
cost effective or in the interest of the manufacturing company. The
notable exception to this generalisation is the nuclear industry,
where the risks associated with plant failures are recognised to be so
much greater.

Despite the efforts principally of the nuclear industry in
gathering and documenting failure rate data, the task of obtaining
realistic frequencies of failure for any particular application is
further hampered for the following reasons:

1 The maintenance strategy of the plant on which the data was
collected, and that of the plant on which it is to be applied, may
well be different.

2 The quality of the maintenance staff will vary.

3 The environment factors will also vary from plant to plant.

4 The way in which a particular piece or section of process plant
will be operated, will undoubtedly be different.

5 Some of the faults are so rare that they have not been reliably
reported as yet.

Some of the above mentioned factors, such as the maintenance
intervals and the environmental conditions, can be taken into
consideration using existing theories of failure rate and reliability.
Unfortunately, many of the more qualitative differences are not so
easy to take into account. Given these problems, the task of obtaining
accurate and realistic failure rate data is therefore very difficult.
However, in regard to the KBS, the problems are not quite so serious because of the way in which the failure rate data is actually employed. Unlike risk analysis where an absolute figure for a reliability or frequency of an undesired event is required, the KBS uses the failure rate data for comparative purposes only. The problems which arise because of differences in operational conditions and maintenance strategies are therefore less significant, providing that the different items of equipment are treated the same, relative to each other.

For example, if an alarm can be caused by either a two inch pipe rupture or a valve leakage, then using nuclear industry failure rate data for these faults of $1 \times 10^{-9}$ F/h and $1 \times 10^{-8}$ F/h respectively, from Wash 1400 [51], the relative likelihood of the two faults would be calculated as follows:

$$\frac{1 \times 10^{-9}}{1 \times 10^{-9} + 1 \times 10^{-8}} = 9.1\%$$

Likelihood of pipe rupture

$$\frac{1 \times 10^{-8}}{1 \times 10^{-9} + 1 \times 10^{-8}} = 90.9\%$$

Likelihood of pipe leakage

If in reality the actual process plant on which the alarm resides is less well maintained, then the absolute failure rate values will be greater, but the relative likelihood of the two faults should remain roughly constant, providing that, for example, the valves are not maintained at the expense of the pipework.

There are three main sources of failure rate data, namely risk analysis case studies, databanks and the literature. The former, such as the WASH 1400 Nuclear Reactor Safety Study [51] and the Canvey
Island Report [52], are quite informative because the source and application of the information, along with any assumptions made, are usually discussed in some detail.

Failure rate databanks exist within many of the larger process and nuclear industries, as well as bodies specialising in safety and reliability, eg. The Systems Reliability Directorate (SRD). Like the information in the literature, such as the OREDA data book, the quality of the data from these sources is often variable; in some circumstances quite detailed, and in others very sketchy. For the purposes of this research project, failure rate data from references [51-53] was utilised.

It is envisaged that given sufficient diagnostic feedback, a fault diagnosis system could modify its 'a priori' figures for failure frequencies, based on the faults that were observed to occur on the process plant. However, the adaptive process would have to be performed very carefully, because by definition, very little plant specific information would be obtained for the rarer faults.
Chapter 6

THE IMPLEMENTATION OF THE FAULT DIAGNOSIS SYSTEM

The aim of the research project has been to develop a methodology for diagnosing the causes of process plant alarms. In order to test the ideas and discover any practical limitations of the techniques employed, a number of programs have been developed. The most significant of these is the knowledge based system which is intended to provide the process operator with support in diagnosing faults.

To enable the problems of process dynamics and the real time aspects of the KBS to be investigated, a number of auxiliary programs have also been written. These have been designed to simulate or play back the dynamic behaviour of a process plant and mimic the operation of a process monitoring computer. The implementation of the process simulation software and the KBS is described in Sections 6.1 and 6.2.

6.1 The Process Simulation Software

An overview of the process simulation software and its relationship with the KBS is shown in Figure 6.1. Apart from the KBS, all the auxiliary software was written in Vax FORTRAN 77. The choice of languages was limited to those available on the MicroVax computer designated for the project, these being Vax LISP, Vax Fortran and the POPLOG languages Common LISP, POP11 and PROLOG. The decision to use FORTRAN 77 was based on its ease of interfacing with the VMS system service routines and the author's familiarity with the language.
The contents of the process variable database and the function of the process simulation tasks are discussed in Sections 6.1.1 to 6.1.8.

6.1.1 Process Variable Database

The process variable database is in fact a collection of FORTRAN common blocks, which reside in physical memory when the fault diagnosis system is running. The same common blocks are defined within
the simulation tasks. After compilation, the programs are linked with the shared image and then the memory can be accessed as if it were local variable storage.

The database essentially contains four different types of information:

1. The first set is the actual values of the monitored process variables. In the present database up to 30 variables can be stored, each one being kept for 30 time scans. After 30 scans the information is overwritten with the latest data. The process variable database is quite small and unsophisticated, but it was deemed adequate for the purpose of simulating the behaviour of the process monitoring computer.

2. The second set of information defines the attributes of each variable in the database. These include the variable type, the variable name and the alarm limits, where applicable. Every variable in the database can have up to five alarm limits associated with it, as well as a hysteresis band to prevent process noise from repeatedly triggering the alarm. In addition to this, the name of each limit is required along with a logical flag to specify which side of the limit is the alarmable state. Duplicate or redundant measurements of the same variable are also identified so that any alarms activate when the reconciled value of the measurement crosses a limit value.

3. The third set of data is the alarm message stacks. When a variable deviation is initially detected by the alarm detection task, a message is placed in the unaccepted alarm message stack. This stack can hold up to 100 messages. When the operator accepts the alarm messages they are then transferred to an accepted alarm message stack. This stack can contain up to 50 messages and as new alarms are accepted, the old messages are written to an alarm log before being overwritten.
The fourth set of information contains the internal variables used within the dynamic simulation. This would not be found in any traditional process variable database, but is included here to enable the output from the dynamic simulation to be copied more easily into the monitored variable array.

6.1.2 The Database Initialisation Program

When the database is installed in physical memory, it is initially empty. Therefore before the other process simulation tasks are run the application specific information, such as alarm limit definitions, needs to be written into the database.

This task is performed by the database initialisation program, which simply contains the assignments for the database variables. Once the program has executed it is not required until the next database installation.

6.1.3 The Alarm Detection Task

The alarm detection task runs as a detached process on the MicroVax. Once every scan interval it checks the value of each monitored variable against it's alarm limits, if any have been defined. If a deviation state is detected and the alarm has not been disabled, then the name of the process variable, the deviation state and the time when the alarm was detected is placed in the unaccepted alarm message stack. If the stack is full, the program waits until a space becomes available.

To prevent separate indication of the same variable alarming at different points in time, the program checks the agreement of any related indications. The measured values of those indications which do agree are then averaged and the result tested against the alarm limits. If the reconciled measurement is in an alarm state, all the averaged indications are placed in an alarmed state.
Repeated messages are prevented from being placed in the stack by setting a flag for the alarm in question, once the message has been queued. This flag is cleared when the variable returns to its non-alarmed state.

Access to the process variable database is co-ordinated using the MicroVax VMS lock manager [54]. This facility is provided within the VMS operating system to allow a number of processes to share a common resource. The lock manager prevents the undesirable situation where one task is reading from the database whilst another is writing to the same memory.

The alarm detection program is terminated by setting a flag in the process variable database using the supervisors console.

6.1.4 The Supervisor's Console Task

The supervisor's console program was written to allow the user to monitor and modify the information in the database. This proves useful when debugging the process simulation software and introducing fault conditions into the dynamic simulation of the reactor charging system.

When the program is executed, the user is presented with a menu of options. These include observing the monitored variable values, changing the alarm limit values and listing the accepted and unaccepted alarm message stacks. In addition the user can stop the various detached processes, introduce faults into the dynamic simulation and display a mimic diagram of the process, which is updated at twice the scan frequency.

6.1.5 The Reactor Charging System Dynamic Simulation

The dynamic simulation of the reactor charging system was initially developed to test and improve the operation of the KBS. Prior to its development the user had been required to manually adjust the values of the process variables.
In common with other workers in this field, the benefits of the simulation were soon realised. The variety and magnitude of the faults that can be synthesised far exceeded those that could have been safely simulated or observed on real process plant. Additionally, the behaviour and configuration of the process units can be modified with relative ease.

The description of the important elements of the dynamic simulation is included in Appendix B.

6.1.6 The Plant Monitoring Task

The principal function of the plant monitoring task is to simulate the operation of scanning the process sensors and updating the database with new information. In practice, because the output from the simulation is already in the database, this simply involves transferring the data from one memory location into another, once every scan interval.

The simulation output could have been written directly into the monitored variable array, thus avoiding the necessity for an additional detached process. However, this would have imposed further demands on the simulation software such as synchronising the data transfer and managing the information already present in the database. It was therefore decided to allow the simulation program to be free running and independent of the other software.

Since the plant monitoring task copies the data once every scan interval and hibernates for the remaining time, it consumes very little CPU time.

6.1.7 The Process Log Playback Task

Instead of writing a dynamic simulation for the batch distillation process, a number of faults were simulated on the actual plant and the key process variables were logged.
In total 19 process variables were logged at scan intervals of between 5 and 30 seconds. As described in Chapter 7, the data was copied onto floppy disk and then transferred to the LUT Chemical Engineering MicroVax computer. To enable the simulated faults on the plant to be diagnosed by the KBS, the information was replayed back into the process variable database at the same rate at which it was logged.

In order to achieve this the plant monitoring program, described in the previous section, was adapted. Rather than using the output from the dynamic simulation, the variable values were read from four data files and then written to the database.

6.1.8 Installing And Starting The Process Simulation Software

For each modelled system a Digital Command Language (DCL) command procedure was written to install the database, initialise its contents and then initiate the appropriate detached processes. For the reactor charging system, the dynamic simulation, the plant monitoring program and the alarm detection tasks are executed. When the batch distillation plant is being considered the process log playback program and the alarm detection program are utilized.

6.2 The Development KBS

A schematic diagram of the program modules used within the development KBS is shown in Figure 6.2. The run-time system comprises five modules, the control program, the inference engine, the operator interface, the knowledge base and the interface to the process variable database. The sixth module is run independently to preprocess or compile the rulebase.

The remainder of the chapter describes the key functions of these modules in more detail, and other related issues.
6.2.1 The Languages Used Within The Development KBS

As discussed in Section 6.1, the languages that were available on the project MicroVax computer were FORTRAN 77, Vax LISP and the POPLOG languages PROLOG, POP11 and COMMON LISP. PROLOG was initially chosen to develop the KBS principally because of the previous work of Andow [25]. In the paper Andow describes a pilot study that was conducted to evaluate the application of IKBS techniques to process plant fault diagnosis.
In the conclusions, PROLOG is described as a language that is well matched to the computational requirements of the application because of its search and pattern-matching features. It is however, noted that PROLOG does lack some of the useful features of conventional languages. Andow suggests that a system which combines PROLOG with procedural languages, such as POPLOG, is probably the best way forward.

Given the findings of Andow, a simple prototype fault diagnosis system was written in PROLOG, in part to evaluate the suitability of the language. The following features of PROLOG were found to be useful:

1 PROLOG is a logic oriented language which is well equipped to describe the type of knowledge chosen for the development KBS.

2 The inherent goal driven searching mechanism is also useful for certain fault diagnosis strategies.

3 The ease of representing and manipulating textual information is seen as a positive advantage, in addition to the languages ability to manipulate list structures.

A fuller description of the PROLOG language is given by Clocksin and Mellish [55], Bratko [46].

The major disadvantage of POPLOG PROLOG is its speed of execution. Being an interpreted language, programs written in this dialect of PROLOG do not run as fast as their equivalent in FORTRAN 77. However, the language was found to be excellent for developing the KBS.

As described in Section 6.2.2 and 6.2.5, some software has also been written in POP11, which interfaces relatively easily with PROLOG and the VMS system service routines.
6.2.2 The KBS Control Program

The control program co-ordinates the operation of the KBS software. Although the fault diagnosis system is operator driven, it does not idle in between processing the user commands. Instead it continually performs a number of tasks. These include reading input from the users terminal, checking the database for new alarms, updating the time and date on the terminal screen and retrieving process variable and alarm information from the database.

The main structure of the control program is an infinite loop. As this cycles, the various tasks are called sequentially. When a completed command input line is received, the text is then passed to the operator interface for processing. After the processing is completed, the program control returns to the infinite loop.

The POP11 language was used to encode the above operation for two reasons. Firstly it is easier to access the process variable database and the VMS system service routines from POP11 and secondly loop structures in POP11 are easier to construct and comprehend than in PROLOG.

The function of the four tasks is described in more detail in Appendix C.

6.2.3 The Implementation Of The Inference Mechanism

The inference mechanism that has been developed for the fault diagnosis system is specific to its domain of application. This is principally because the diagnostic method is programmed into the inference engine, rather than being defined within the rulebase in terms of meta-rules. The justification for this approach is that the inference engine includes more specialised functions to combine logic trees and reduce them to minimum cutsets.

Having discussed the functionality of the inference mechanism in Chapters 4 and 5, is not re-iterated here.
6.2.4 The Rulebase

The fault diagnosis system rulebase essentially contains three types of information:

1. A description of the key process variables and their indications;
2. The causal relationships between the key process variables and the effects of the process unit failures;
3. The process equipment failure rate data.

The rulebase information for the initial prototype KBS was simply created using a text editor and then loaded by the fault diagnosis system at run-time. This initially proved to be adequate, but as the rulebase became increasingly more complex, a parser program was written to check for syntax errors. With the development of the rulebase compiler, the parser program was incorporated into the compiler. The rulebase that is now used directly by the fault diagnosis system is therefore the output from the compiler. The contents of this rulebase are discussed in Section 6.2.7. The information that needs to be specified prior to compilation is discussed in more detail in Appendix D.

6.2.5 The Interface Between The KBS And The Process Variable Database

In order to evaluate the KBS using simulated process variable information, it was necessary to interface the software to the process variable database. Although the POP11 language can be linked with external object code modules, it is specified in the documentation that these cannot be shared images in the current version of POPLOG. Unfortunately the database is a shared FORTRAN image, therefore another method of communicating between POP11 and the database was sought.
The simplest solution to the problem involves using a separate FORTRAN server program, which can link to the database, connected to the POPLOG system via mailboxes. A mailbox is in effect a data pipe which can connect the output channel of one program to the input channel of another, as discussed in reference [54]. The mailboxes can buffer information and are used in a number of VAX based KBS's to communicate between processes, [29,56].

Within the development KBS two mailboxes are defined, one to transfer data to the FORTRAN server and the other to receive the data back. When the KBS requires to access the database, a request is first coded into a specific ASCII format, and then the resulting string is written to the POPLOG output mailbox.

The FORTRAN server program hibernates until it receives a request from its input mailbox. After interpreting the request, the database is read or updated and any resulting information is converted into ascii ready for the transfer back. In the mean time the KBS hibernates until it receives the result back via its input mailbox.

To enable the different types of information within the process variable database to be accessed, a communications protocol has been developed. This specifies how the KBS requests should be encoded and how the various data types such as reals, integers, characters and logical variables should be transferred between the two processes. Since POP11 is the base language within the POPLOG system, a number of POP11 procedures have been defined to implement the protocol and effect the data transfer.

The standard POPLOG interface between POP11 and PROLOG has been used to transfer the information between the two languages.

6.2.5.1 Calculation Of The Fuzzy Logic Membership Functions

When the numerical value of a process variable is requested from the database, the FORTRAN server program also calculates the set of fuzzy membership functions. Each fuzzy range envelope is defined in
terms of three co-ordinate pairs, in ascending value of the process variable. The three points define two straight lines, which can be interpolated to calculate the values of a fuzzy membership function. For example, consider the three fuzzy range envelopes shown in Figure 6.3

The three points for the low and high envelopes are A, B and C and G, H and I respectively. The normal range is defined by points D, E and F, but a filter imposed by the FORTRAN program constrains the maximum value of the fuzzy membership function to 1.0. This therefore enables the trapezium DJKF to be effectively defined.

Figure 6.3 The Definition Of The Fuzzy Membership Functions

6.2.6 The Operator Interface

It is recognised that the user interface is a very important aspect of any fault diagnosis system. However, a detailed investigation of this subject area was considered beyond the scope of the research project. Despite this, all reasonable efforts have been
made to display the diagnostic output of the KBS in an informative way. A description of the four user displays supported by the KBS is included in Appendix E.

6.2.7 The Rulebase Compiler

The rulebase compiler essentially has three distinct functions:

1. To parse the source code to the compiler. Any syntax errors not identified by the PROLOG interpreter are then reported to the user. In addition, the rulebase information is checked for completeness.

2. To gather together the individual causal rules according to the process variable deviations they cause. The more natural language format of the rules is then converted into a form which can be more readily processed by the inference engine.

3. To interrogate the rules to determine the potential consequences of each basic fault, subsystem failure and process variable deviation. This information is then used to identify common cause failure modes.

When the compiler is executed the user is first prompted for the name of the PROLOG source code. If a suitable filename is supplied, the user is then asked if he/she requires the error and warning messages to be written to a log file. The rulebase is then parsed and compiled. Finally, the compiled rulebase information is written to an output file with the same name as the input file, apart from the file type which is changed to 'rul'.

The operation of the rule parser and compiler is discussed in more detail in Appendix F.
Chapter 7

THE APPLICATION OF FAULTFINDER TO THE SYSTEMATIC
GENERATION OF THE ALARM DIAGNOSIS INFORMATION

This chapter describes how the FAULTFINDER suite of programs was used to generate the alarm diagnosis information. The modelling of the reactor charging system and the batch distillation plant are described in Sections 7.1 and 7.2 respectively. Finally, a discussion of the modelling problems encountered is included in Section 7.3.

7.1 The Reactor Charging System

As discussed in Chapter 6, the reactor charging system is a hypothetical section of process plant which is based on a previous example discussed by Andow and Lees [57]. A schematic diagram of the modelled system is shown in Figure 7.1 and a manifest of the plant equipment is given in Table 7.1.

The function of the system is to provide reactor R1 with a constant flowrate of ethanoic acid. The fluid is drawn from a storage tank (tank 1) through a short leg of pipe and into the pump. From there the acid is pumped down a 40 m pipeline, through a control valve, and up through a 4 m vertical section of pipe into the elevated buffer tank (tank 2).
Figure 7.1 The Reactor Charging System
Table 7.1 The Manifest Of Units In Figure 7.1

<table>
<thead>
<tr>
<th>Unit Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cv 1</td>
<td>Flow control valve</td>
</tr>
<tr>
<td>trans 1, trans 2, trans 3</td>
<td>Differential pressure transducers</td>
</tr>
<tr>
<td>ls 1, ls 2</td>
<td>Level sensors</td>
</tr>
<tr>
<td>fs 1, fs 2</td>
<td>Orifice plates</td>
</tr>
<tr>
<td>pipe 2, pipe 3, pipe 4, pipe 5, pipe 6 and pipe 7</td>
<td>Process lines</td>
</tr>
<tr>
<td>ps 1, ps 2</td>
<td>Pressure sensors</td>
</tr>
<tr>
<td>pump 1</td>
<td>Centrifugal pump</td>
</tr>
<tr>
<td>react 1</td>
<td>Reactor</td>
</tr>
<tr>
<td>tank 1</td>
<td>Storage tank</td>
</tr>
<tr>
<td>tank 2</td>
<td>Elevated buffer tank</td>
</tr>
<tr>
<td>ts 1, ts 2</td>
<td>Temperature sensors</td>
</tr>
<tr>
<td>valve 1, valve 2, valve 3, valve 4, valve 5, valve 6, valve 7, valve 8, valve 9</td>
<td>Isolation valves</td>
</tr>
<tr>
<td>cnt 1</td>
<td>Level controller</td>
</tr>
</tbody>
</table>

From tank 2 the liquid flows under gravity, through a constriction provided by valve 1, into reactor 1 which is maintained at a pressure of 0.3 barg. The flowrate into the reactor is therefore regulated by the level in the buffer tank, which is in turn controlled by manipulating the flowrate into the tank, using level sensor ls 2, controller cnt 1 and control valve cv 1. The normal flowrate through the system is 0.1 m³/min, which is achieved with a buffer tank level of 0.5m.

Ethanoic acid was chosen as the process fluid for three reasons:

1. It is a common feedstock in a number of chemical processes;
2. The reactant is corrosive and toxic, and hence poses a potential hazard to plant personnel;
The relatively high freezing temperature of 16.6 °C requires the pipes and tanks to be electrically traced, which adds an interesting facet to the problem.

For the purposes of demonstrating the modelling and fault diagnosis techniques, the reactor charging system is well instrumented. Eight process variables are measured on the plant, in addition to the controller setpoint and output, and the values of these indications are stored in the process variable database. The names assigned to the process variables and their indications are listed in Table 7.2.

Apart from the flow of ethanoic acid into the buffer tank (f1), all the process variables are monitored by single indications. The redundancy in the flow instrumentation is included principally to demonstrate some of the features of the indication interpretation technique. In addition, it can also be seen that the controller input signal, which will be the same as the buffer tank level indication, is modelled separately as a signal variable. The reason for this distinction is discussed later.

The process variable alarm limits were selected using engineering judgement and by experimenting with the dynamic simulation of the process. In order to improve the realism of the studied example, two sources of process noise were introduced into the simulation. Firstly, it was assumed that the level of fluid in the storage tank would be expected to rise and fall between 3.5 and 1.5 m, as the reactor consumed the ethanoic acid, and as it was subsequently replaced from external sources. Given the large volume of the storage tank (50 m³), the transition in level was very gradual.

The second source of process noise was introduced by superimposing a sinusoidal pressure deviation in the reactor pressure p2. An amplitude of 0.01 bar (peak to peak) was selected on the basis that it perturbed the downstream flowrate by 5%. The period of the waveform was chosen so that the disturbance affected the buffer tank.
level. A value of 300 s was found to be suitable for this purpose. The noise was prevented from propagating upstream by allowing the level controller some deadband around its setpoint.

Table 7.2 The Process Variable And Indication Names

<table>
<thead>
<tr>
<th>Description</th>
<th>Variable</th>
<th>Indication</th>
<th>Key in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage tank level</td>
<td>11</td>
<td>stg_tnk_lev</td>
<td>A</td>
</tr>
<tr>
<td>Buffer tank level</td>
<td>12</td>
<td>buf_tnk_lev</td>
<td>F</td>
</tr>
<tr>
<td>Pump discharge</td>
<td>p1</td>
<td>pump_pl_pres</td>
<td>C</td>
</tr>
<tr>
<td>pressure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reactor pressure</td>
<td>p2</td>
<td>reac_rl_pres</td>
<td>J</td>
</tr>
<tr>
<td>Flow of ethanoic acid</td>
<td>f1</td>
<td>buf_acd_f1</td>
<td>D</td>
</tr>
<tr>
<td>into the buffer tank</td>
<td></td>
<td>buf_acd_f1_2</td>
<td>E</td>
</tr>
<tr>
<td>Flow of acid into the reactor</td>
<td>f2</td>
<td>reac_acid_f1</td>
<td>H</td>
</tr>
<tr>
<td>Upstream pipeline</td>
<td>t1</td>
<td>inlet_temp</td>
<td>B</td>
</tr>
<tr>
<td>temperature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Downstream pipeline</td>
<td>t2</td>
<td>outlet_temp</td>
<td>I</td>
</tr>
<tr>
<td>temperature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controller output</td>
<td>s1</td>
<td>cnt_l_output</td>
<td>G</td>
</tr>
<tr>
<td>Controller input</td>
<td>s2</td>
<td>cnt_l_input</td>
<td>F</td>
</tr>
<tr>
<td>Controller set point</td>
<td>s3</td>
<td>cnt_l_setpt</td>
<td>K</td>
</tr>
</tbody>
</table>

In most cases, three alarm limits are defined for each process variable, namely high, low and very low. The exceptions are the storage tank level which has high and low alarms, the pipeline tracer temperatures which only have low alarms, and the reactor pressure which has high, low, very high and very low alarms.

Two main criteria were used to select the alarm limit values:

1. Clearly the limits were chosen so that they were outside the normal operating range of the variables concerned;
The alarms were designed to give the operator as much time as possible to correct for the fault. For example, the buffer tank level only fluctuates by approximately 0.1%. Therefore the level alarm limits can easily be set outside the normal range of the variable, and yet be sensitive enough to give the operator a reasonable time to correct for the fault.

As discussed in Section 5.3.2.1, the process variables are discretized into a number of states such as high, low and normal. The shape and range of the fuzzy envelopes are based on the alarm limit values and the process noise levels. The normal variable range is always defined so that an indication will be discretized as 100% normal, if its value is within the normal operating bounds. For example, consider the fuzzy discretisation graph shown in Figure 7.2.

Figure 7.2 The Fuzzy Discretisation Graph for Pressure P1

The pump discharge pressure will fluctuate between 5.63 and 5.37 barg, as a consequence of the process noise. Therefore the 100% normal line \( (h-i) \) extends between these two pressure limits.

Clearly the normal range discretisation convention is somewhat arbitrary since the difference between a completely normal process variable value and one slightly in deviation is subjective. As a
consequence there are no compelling reasons why a system modeller should strictly adhere to the convention. However, if a fuzzy membership function is interpreted as a measure of the confidence that a variable is in a given state, then the approach described above provides a good starting point.

The fuzzy transition regions between the different states are defined so that when an alarm occurs, the membership function for the variable being in its alarmed state is unity. For example, from Figure 7.2 it can be seen that when the high pressure alarm is triggered, the indication is discretized as 100% high and 0% normal.

Whilst the two aforementioned criteria enable the fuzzy envelopes bordering the normal range to be defined precisely, it is less obvious how the very low or very high fuzzy envelopes should be specified. The alarm criterion can again be used to fix the 100% very high or very low points. However, it is difficult to decide with confidence on the gradient of the fuzzy transition envelopes.

For the purposes of modelling the reactor charging system, the very low/low and very high/high transition regions were defined with the same gradient as the low/normal and high/normal transition regions respectively. For example, in Figure 7.2 the gradient of line ab is the same as the gradient of line ef.

7.1.1 Modelling Process Plant Using Faultfinder

The main objective of the modelling task was to develop an accurate and complete fault propagation model of the reactor charging system, which could then be used to derive the fault trees, describing the causes of the process alarms. As discussed in Chapter 3, the FAULTFINDER suite of programs were used for this purpose.

The systematic generation of the alarm analysis information is very attractive for the reasons listed overleaf:
The task of modelling large and complex process plants requires considerable time and effort, as reported by Patterson [5] and Felkel and Zapp [11]. Any tools which assist in this process will therefore result in economic savings.

There is the problem of ensuring that most of the significant causes of a process fault have been identified, as discussed by Andow and Galluzzo [58]. However, if a systematic method of deriving the causes of a process disturbance can be developed, to the stage where it yields reliable results, then the technique should lessen the chance of human error and oversight.

Once the fault propagation model of the process has been refined, the task of maintaining and updating the diagnostic information, as the plant is modified, should be greatly eased.

The accuracy and completeness of any systematically generated fault propagation information, will always depend on the quality of the model describing the behaviour of the considered process, however well developed the methodology is for interrogating this data structure. Within the FAULTFINDER suite of programs, the fault propagation model is specified in terms of a configuration of individual unit models, linked by their common process variables. The component models describe how process variable deviations propagate through the unit, and how they are initiated within the unit.

The behaviour of the individual units can be described using three different knowledge representations, namely propagation equations, event statements and decision tables. For example, consider the open tank unit shown in Figure 7.3.

The flow into the tank (QIN) is only influenced by the upstream pressure gradient (GIN) whilst the flow out of the tank (QOUT) is both a function of the downstream pressure gradient (GOUT) and the tank level (L3VES). The tank level is simply a function of the inlet and outlet flowrates. To enable the unit model to be linked with components upstream and downstream, a number of inlet and outlet ports must be defined. In the case of the open tank model, three are
specified, the inlet (port 1), the outlet (port 2) and the vessel port (port 3). The latter allows the internal variables such as the level, temperature and composition to be observed.

Figure 7.3 The Open Tank Unit

The propagation equations are essentially the same as the signed functional equations described by Martin-Solis et al. [21]. For example, the relationship between the tank level and the inlet and outlet flows can be specified using Equation 7.1:

\[ L_{3VES} = F(Q_{IN}, -Q_{OUT}) \]  

(7.1)

This equation states that a level deviation can be caused by a inlet flow deviation (of the same sign or direction) or an outlet flow deviation of the opposite direction.

The event statement also can be used to specify the propagation of a variable deviation through a unit. For example, consider the following event statement:

V Q_{IN} LO; L_{3VES} LO

This statement represents the causal relationship between a low pressure gradient (analogous to flow within FAULTFINDER) at the inlet to an open tank and the level within that tank. The first half of the
statement describes the cause, the second half, delimited by the colon, the consequent event. A 'V' precedes the cause to indicate that it is a variable deviation, as opposed to a prime cause failure.

However, the main function of the event statement is to describe the relationship between process equipment faults (prime cause failures) and the variable deviations they cause. For example, a leakage of the tank will cause both the level and the discharge flowrate to decrease. This causal relationship is described in the following form:

F LK-LP-EN:L3VES LO,Q2OUT LO

Again the cause is specified first, preceded by an 'F' to indicate that it is an equipment fault, followed by the consequences, which are delimited by commas. The consequences may or may not be mutually exclusive.

Finally, the decision table method of knowledge representation enables AND logic to be introduced into the model. For example, the temperature in the tank (T3VES) will usually only be a function of the upstream temperature. However, if the downstream temperature deviates AND there is reverse flow back into the vessel, then the tank temperature will also deviate. The decision table for the causes of a low tank temperature is shown below:

V Q2OUT REV V U2OUT LO T T3VES LO

The two cause events, Q2OUT REV and U2OUT LO are listed first, followed by the consequence T2VES LO. The second cause (V U2OUT LO) represents the reverse propagation of the temperature deviation back into the tank model.

When the unit model is complete, the FAULTFINDER model generation program (MODGEN) converts the user supplied data into mini-fault trees, relating process variable deviations at one port to causes at the other ports, and faults within the unit. For example, in
the system shown in Figure 7.3 a low tank level will be caused by either a low inlet flow, a high outlet flow or a tank leakage. The mini-fault tree that would result is illustrated in Figure 7.4.

Figure 7.4 The Causes Of A Low Tank Level

A more detailed account of the modelling of fault propagation is given by Kelly and Lees [23] and within the FAULTFINDER methodology manual [59].

7.1.2 Generating The System Specific Unit Models

Although the facilities provided by the MODGEN program do simplify the modelling process, the task can still be quite time consuming, especially if a number of the unit models have to be generated from first principles. In order to help alleviate this problem, a library of standard unit models has been developed. These are based on the modelling experience gained through the application of FAULTFINDER to a variety of example systems.

Unfortunately, the library of standard models cannot cater for every item of process equipment, and those unit models which are included may not exactly describe the behaviour of the units in the considered system. As a consequence, the user is still required to
create specialised models and refine the existing models for most applications. However, this requires considerably less effort than the alternative.

In order to model the reactor charging system it was necessary to create five new types of unit model, namely an electrical pipeline tracer, an orifice plate, a differential pressure transducer, a combined level sensor and isolation valve, and a reactor model. Apart from the electrical pipeline tracer, all the unit models were derived from examples of similar equipment from the FAULTFINDER library.

The electrical pipeline tracer was simply modelled as a heater unit which would cause low temperature deviations at its outlet ports, if there was an external low temperature and either the device or its power supply failed. The outlet ports were connected to the process pipe models. It was decided that the electrical tracer subsystem should be modelled as a utility, rather than being integrated into the pipe models, because it would be easier to detect the common mode failures of either the tracing device or the power supply unit.

The orifice plate model was derived from the standard library model for a flow sensor. The two main modifications were the removal of the power utility port and the addition of an extra signal output port. The outputs from the orifice plate were designed to feed the differential pressure transmitter model. The latter was based on a forward acting controller model, and converted the differential pressure signal into an electrical signal. The device therefore required a connection to the instrument power supply utility.

The combined level sensor and isolation valve model was created so that the blockage and inadvertent closing of the isolation valve faults, would be included in the causes of the level control loop failing stuck/normal. At first the two units were modelled separately, but it was found that the fault tree synthesis program (FAULT) did not develop the causes of an inactive control loop to the isolation valve model.
The problem is due to the way in which the FAULT program traces the causes of the control loop invariant fault. The program starts at the control element (the control valve) and develops the causes of the control signal not changing (SIG NCHA). At present, the program terminates when the sensor model is reached, since only the signal variable can exhibit the 'NCHA' state. The problem could be solved if the other process variables could exhibit this state, since the manual valve blockage would simply cause no change in the measured level.

The combined model was generated by adding the valve blockage and valve closed faults to the causes of the level sensor failing stuck/normal. An extra port was also defined so that a failure of the electrical pipe tracer unit could cause the valve to become completely blocked by frozen ethanoic acid.

Finally, the reactor model was created simply as a dummy unit which could source high and low pressure deviations. A vessel port was defined so that the variable deviations could be monitored by a pressure sensor.

The tailoring of the standard FAULTFINDER library models for the reactor charging system involved three different types of modification:

1. The failure modes of each model were scrutinised to determine their relevance to the current application. Where they were not deemed to be appropriate, they were either replaced, supplemented or deleted. For example, the standard model for an open isolation valve has two very similar failure modes which give rise to a restriction of flow through the device, namely HV-F-SH and HV-D-SH. The former represents the device failing shut whilst the later describes the action of an operator closing the valve. It was decided that the manual valves in the reactor charging system would not fail closed of their own accord, as distinct from becoming blocked, therefore the HV-F-SH failure mode was removed from the model.
Some of the models were amended by adding extra port definitions. For example, the reverse acting controller model was given an extra signal output port, so that the process monitoring computer could read the signal. The same effect could have been obtained by using a signal splitter model. However, this would have resulted in an extra unit in the system configuration. The process pipe model was also modified so that the causes of a low temperature deviation within the unit could be traced to faults within the electrical pipeline tracer model.

For certain models it was also necessary to modify the way in which the units propagated process variable deviations. For example, the buffer tank level controller was designed to operate with both proportional and integral action. To enable the dynamic simulation of the reactor charging system to be as realistic as possible, the controller gain and integral time constants were determined using a standard design technique. Because of the particular characteristics of the level controller, it was observed from the dynamic simulation that a high or low deviation in the level sensor or set point inputs would drive the controller output very high or very low respectively. Furthermore, the response time for the controller output to become saturated was very short.

The standard library model for a reverse acting controller is shown below. The comments following the '%' symbol have been added here to clarify the meaning of various items:

1) MODEL NUMBER NAME
5 CONTROLLER (REVERSE ACTING)

NO. OF ENG. ASSUMPTIONS/DESCRIPTIONS: 4
% No. of comments in 2)
NO. OF PROPAGATION EQUATIONS: 1
NO. OF EVENT STATEMENTS: 9
NO. OF DECISION TABLES: 0
NO. OF FAILURE MODES: 1
% No. of distinct modes
% of failure. Used for
% modelling trip systems
2) ENGINEERING ASSUMPTIONS AND DESCRIPTIONS

PORT 1, INPUT SIGNAL FROM SENSOR % comment line 1
PORT 2, OUTPUT SIGNAL TO VALVE % comment line 2
PORT 3, SET POINT, PORT 4, UTILITY % comment line 3
PNEUMATIC; REVERSE ACTING % comment line 4

3) PROPAGATION EQUATIONS

S2SIG=F(-S1SIG,W3IN) % the causes of output

4) EVENT STATEMENTS

F CNT-F-LO:S2SIG LO,S2SIG NONE % device fails low
F CNT-F-HI:S2SIG HI % device fails high
F CNT-STK:S2SIG NCHA % device fails stuck
O CNT-MAN:S2SIG NCHA % controller in MANUAL
V S1SIG NCHA:S2SIG NCHA % input signal frozen
F SIG-CB:S2SIG NONE % pneumatic line blocked
F SIG-PB:S2SIG LO % pneumatic line blocked
V S4UTL NONE:S2SIG NONE % no instrument air
S NORMAL:S2SIG SOME % normal state of device

5) DECISION TABLES

N/A % no decision tables

6) SUPPLEMENTARY INFORMATION

NORMAL STATE:SOME OUTPUT SIGNAL IS PRESENT % additional comments

NO MULTI-COMPONENT FEATURES % model does not
% involve any chemical
% separation processes
The propagation equation defines how the controller output signal (S2SIG) is both a function of the set point (W3IN) and the input signal (S1SIG). From this propagation equation the following three mini-fault trees can be derived:

**Figure 7.5a The Causes Of A High Output Signal**

```
S2SIG HI
   |
   OR
  /|
S1SIG LO  W3IN HI
```

**Figure 7.5b The Causes Of A Low Output Signal**

```
S2SIG LO
   |
   OR
  /|
S1SIG HI  W3IN LO
```
As can be seen from Figure 7.5b, a high input signal (S1SIG) is considered as a cause of a low output signal (S2SIG). In theory this mini-fault tree is correct. However, because of the fast response of the control loop, a mildly high deviation in S1SIG will cause S2SIG to be driven to state 'NONE' almost immediately. Since the relationship between S1SIG HI and S2SIG LO is only fleeting, it was decided that the reverse acting controller model should be modified, so that the previously described mini fault tree would not be generated. This was simply achieved by removing the reference to the controller input signal from the propagation equation and inserting the following two event statements:

V S1SIG LO:S2SIG HI
V S1SIG HI:S2SIG NONE

The complete controller model and those for the other units in the system are included in Appendix G
7.1.3 The System Configuration

The reactor charging system was modelled as a network of 50 units and 63 connections, as shown in Figure 7.6. The failure modes of the instrument cabling, the analogue-to-digital conversion equipment and the monitoring computer were not considered in the study; therefore the outputs from the sensor and transducer units are terminated with dummy tail units.

Although all the process equipment from the upstream storage tank through to the buffer tank inlet pipe is heated by the same electrical tracer, only two pipe units (7 and 15) are actually connected to the tracer model in the configuration. The simplification is made because from the level of plant instrumentation, it is only possible to discriminate between blockage faults (caused by the freezing ethanoic acid) upstream or downstream of the pressure sensor unit (11). By removing the unnecessary 'thermal links' between the process units and the pipe tracer, the system configuration is greatly simplified. For the same reasons, the downstream electrical tracer model is only connected to the pipe unit (36) and the combined level sensor and isolation valve (25).

The system configuration was generated by specifying the type of each unit, and the connections between the various units, to the MASTER program. In addition to this network information, it was also necessary to define the functionality of the control loop and detail the effects of the ethanoic acid freezing in the process lines.

The level control loop was defined by specifying the controlled variable, the manipulated variables, the units in the control loop and whether the loop was of the feedforward or feedback type. The consequences of the ethanoic acid freezing in the process units were described by relating the materials failure to partial and complete blockage faults within the pipe models.
Figure 7.6 Block Diagram For The Reactor Charging System
Once the system configuration had been entered, the MASTER program gathered together the individual unit models. When this task was complete, the program requested the name of the top event model, which defined the variable and deviation state being considered as the top event. For the purposes of this study, the deviation states of the sensor output signals were selected as the top events. The MASTER program then created the input files for the fault tree synthesis program.

7.1.4 Generating The Alarm Fault Trees And Cutsets

The fault trees were synthesised using the FAULT program, by simply entering the filename containing the results from the MASTER program. On completion, a graphical display of the fault trees could then be obtained by running the PLOT program, which produced an output file suitable for a 132 column line printer. For example, the fault tree for the causes of a low flow alarm, as indicated by unit 19 (indication buf_acd_fl_l), is shown in Figure 7.7. The key to the failure mode mnemonics is provided in Table 7.3. The equivalent output from the PLOT program is included in Appendix H.

As can be seen from Figure 7.7, the causes of a low signal output from the transducer unit (19) are first developed to either a failure of the transducer unit itself or of the orifice plate. The fault tree then traces the causes of a low flow through the orifice plate unit (represented in terms of a low pressure gradient C 15 LO) in two main branches; these being the low aperture failure of the control loop and a control loop stuck failure in combination with other potentially compensatable faults.

The control loop fails stuck/normal branch is developed to include failure modes of the control valve, the controller, the combined level sensor and isolation valve, the pipeline tracer unit and the process power utility. The potentially compensatable causes of a low flow deviation include a partial blockage of the buffer tank inlet pipe (21), and leakage and blockage faults in the process units connecting the storage tank to the orifice plate unit.
Figure 7.7 The Low Flow Alarm Fault Tree

LOW SIGNAL AT UNIT 19

OR

S 17 LO

OR

TND.F. LO UNIT 16

OR

S 16 LO

OR

SEN.F. LO UNIT 16

OR

G 15 LO

AND

OR

CL.F.LA LOOP 1

OR

S 30 LO CV.F.LA UNIT 14

A

OR

Q20 LO Q15 LO

B C

OR

D

CL.STK LOOP 1
Table 7.3 The Failure Mode Mnemonics For Figure 7.7

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Failure Mode Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL-F-LA</td>
<td>Control loop fails low aperture</td>
</tr>
<tr>
<td>CL-STK</td>
<td>Control loop stuck</td>
</tr>
<tr>
<td>CNT-F-LO</td>
<td>Controller fails low</td>
</tr>
<tr>
<td>CNT-MAN</td>
<td>Controller in manual mode</td>
</tr>
<tr>
<td>CNT-F-NM</td>
<td>Controller fails normal/stuck</td>
</tr>
<tr>
<td>COMP-BLK</td>
<td>Complete blockage</td>
</tr>
<tr>
<td>CV-F-LA</td>
<td>Control valve fails low aperture</td>
</tr>
<tr>
<td>CV-STK</td>
<td>Control valve fails stuck</td>
</tr>
<tr>
<td>EXT-COLD</td>
<td>External low temperature (from environment)</td>
</tr>
<tr>
<td>HEATFAIL</td>
<td>Electrical pipe tracer fails</td>
</tr>
<tr>
<td>HV-D-SH</td>
<td>Hand valve directed shut</td>
</tr>
<tr>
<td>HV-D-OP</td>
<td>Hand valve directed open</td>
</tr>
<tr>
<td>LK-LP-EN</td>
<td>Leak to a low pressure environment</td>
</tr>
<tr>
<td>PART-BLK</td>
<td>Partial blockage</td>
</tr>
<tr>
<td>POW-LOSS</td>
<td>Power low (utility supply)</td>
</tr>
<tr>
<td>SEN-F-LO</td>
<td>Sensor device fails low</td>
</tr>
<tr>
<td>SEN-F-NM</td>
<td>Sensor fails normal/stuck</td>
</tr>
<tr>
<td>SET-P-LO</td>
<td>Set point is low</td>
</tr>
<tr>
<td>TND-F-LO</td>
<td>Transducer fails low</td>
</tr>
</tbody>
</table>

7.1.5 Converting The Fault Trees Into The Fault Propagation And Indication Rules

As discussed in Chapter 3, the fault tree information is not stored as a complex hierarchy of AND and OR logic, but instead is reduced to its minimum cutset form. However, in order to minimise the number of cutsets that are generated, faults which exhibit very similar symptoms or result in the failure of a subsystem, are grouped together. For example, in the case of the fault tree shown in Figure 7.7, the control loop fails stuck/normal and the control loop fails low aperture branches are both treated as subsystem failure modes, and hence are represented as single compound faults when the minimum cutsets are derived. In addition, because it is impossible to
distinguish between many of the leakage and blockage faults between the storage tank and the orifice plate, as detectable with the level of plant instrumentation, many of these faults have also been grouped into single compound faults, as shown below.

The faults shown in Figure 7.7 were grouped together manually. In the case of the control loop failure modes it would not have been too difficult to automate this process, for the following two reasons:

1 The structure of the synthesised tree naturally highlights the relationships between the failure of the control loop as a whole and the faults within components of the loop (sensor, controller etc.);

2 The degree to which faults within the control loop can be individually identified, with the level of process instrumentation, does not vary with the type of control loop failure mode.

Unfortunately the same is not generally true of blockage and leakage faults within the process unit models. For example, within one alarm fault tree it may not be possible to resolve a blockage fault between two process units. However, within another tree it may well be possible to discriminate between leakages in the same two units because of an intermediate flow sensor. The identification of sensible groupings of process units therefore requires more skill, in terms of cross referencing information from a number of fault trees. As a result the task would be more difficult to automate.

Since the faults were consolidated into groups, there was little advantage in using a cutset generation code such as FTAP [44] or PREP [45], despite the fact that the FAULT program produced the appropriate input files. The concise form of the minimum cutsets for the fault tree shown in Figure 7.7 are listed overleaf, along with the cause of the subsystem and group failure modes:
The causes of the CL-F-LA Loop 1

1 CV-F-LA Unit 14
2 CNT-F-LO Unit 28
3 SET-P-LO Unit 29

The causes of the CL-STK Loop 1

1 CV-STK Unit 14
2 CNT-F-NM Unit 28
3 CNT-MAN Unit 28
4 SEN-F-NM Unit 25
5 HV-D-SH Unit 25
6 EXT-COLD Unit 44 & POW-LOSS Unit 48
7 EXT-COLD Unit 44 & HEATFAIL Unit 48

The causes of a PIPELINE_SECTION PART-BLK

1 PART-BLK Unit 15
2 PART-BLK Unit 14
3 PART-BLK Unit 13
The causes of a PIPELINE SECTION LEAKAGE

1. LK-LP-EN Unit 13
2. LK-LP-EN Unit 15
3. LK-LP-EN Unit 14

The causes of a PUMP SECTION PART-BLK

1. PART-BLK Unit 10
2. PART-BLK Unit 9

The causes of a PUMP SECTION LEAKAGE

1. LK-LP-EN Unit 10
2. LK-LP-EN Unit 9

The causes of a TANK SECTION PART-BLK

1. PART-BLK Unit 8
2. PART-BLK Unit 7
3. PART-BLK Unit 6

The causes of a TANK SECTION LEAKAGE

1. LK-LP-EN Unit 8
2. LK-LP-EN Unit 7
3. LK-LP-EN Unit 6

Once the alarm cutsets had been manually derived, they were separated into those which caused the actual process variable to deviate and those which caused the alarm to spuriously activate. The former were used to derive the fault propagation rules, whilst the later were used as the basis of the indication rules.
Finally, in certain cases the fault propagation rules were edited to include the extra control loop working conditions or the expected variable state boundary conditions, as discussed in Chapter 3. For example, from Figure 7.7 it can be seen that a failure of either the electrical pipe tracer module or of the process power supply, in combination with a low environmental temperature, will cause the flow into the buffer tank to be restricted. However, because the pipe tracer temperatures are directly monitored by indications 'inlet_temp' and 'outlet_temp', these can be used to either confirm or reject the aforementioned hypotheses for the causes of the low flow.

The complete uncompiled rulebase describing the fault propagation behaviour of the reactor charging system is included in Appendix I.

7.2 Modelling The Batch Distillation Process

The second system is a pilot scale batch distillation plant, based at the BP research centre, Sunbury-On-Thames. Four main criteria were used to select the process plant, as listed below:

1 The process had to be sufficiently small, in order that it could be modelled by a single person within a reasonable time scale, yet complex enough to provide an interesting example;

2 It was desirable that disturbances could be introduced into the process, in order to simulate the failure of process equipment;

3 The process had to be well instrumented;

4 There had to be a data connection to the computer running the fault diagnosis software.

The second criterion was necessary, given the infrequency of real faults within a section of process plant that would satisfy the first criterion.
Unfortunately, in the selection of the batch distillation plant all four criteria were not completely met. However, given the difficulty of satisfying the criteria, a good compromise was thought to have been achieved. The two main drawbacks of using the batch distillation plant were as follows:

1. The process was batch not continuous;

2. The process data could not be directly interfaced to the knowledge based system.

The first problem was minimised by keeping the rate of product removal quite low, so that the liquid composition within the reboiler remained relatively constant. Although the liquid composition was not directly measurable, it was estimated from the temperature and pressure as being 18% methylbenzene, 82% ethylbenzene at the start of the fault simulation experiments, and 4% and 96% respectively on completion.

The KBS software could not be directly interfaced to the process instrumentation, because a suitable computer running all the support software was not interfaced to the process. Whilst the code could have been adapted to execute on those computers which were interfaced to the process, this task would have been both time consuming and lacking originality. The problem was therefore overcome by logging the process variables whilst disturbing the process, e.g. switching off the electricity supply to the reboiler heater. The live process data was then saved to data files. On completion, these files were copied onto floppy disk and transferred to the departmental MicroVax computer.

7.2.1 System Description

A simplified schematic diagram of the batch distillation plant is shown in Figure 7.8, and the associated manifest of equipment is given in Table 7.4.
Figure 7.8 The Schematic Diagram Of The Batch Distillation Plant
Table 7.4  The Manifest Of Equipment In Figure 7.8

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bl, B2</td>
<td>Vacuum buffer vessels</td>
</tr>
<tr>
<td>C1</td>
<td>Condenser</td>
</tr>
<tr>
<td>E1</td>
<td>Column</td>
</tr>
<tr>
<td>Hl</td>
<td>Reboiler</td>
</tr>
<tr>
<td>L1</td>
<td>Differential level sensor</td>
</tr>
<tr>
<td>PIC5</td>
<td>Pressure controller</td>
</tr>
<tr>
<td>PT4, PT5</td>
<td>Absolute pressure sensors</td>
</tr>
<tr>
<td>PY5</td>
<td>Electro-pneumatic converter</td>
</tr>
<tr>
<td>R1, R2</td>
<td>Receivers</td>
</tr>
<tr>
<td>S1</td>
<td>Bursting disc</td>
</tr>
<tr>
<td>TE1A,TE3,TE4,TE5,TE6,TE7,TE8,TE10,TE11,TE12,TE13,TE14,TE15,TE16,TE17</td>
<td>Temperature primary elements</td>
</tr>
<tr>
<td>TT1,TT3,TT4,TT5,TT6,TT7,TT8,TT10,TT11,TT12,TT13,TT14,TT15,TT16,TT17</td>
<td>Temperature transmitters</td>
</tr>
<tr>
<td>TIC5,TIC6,TIC7,TIC15,TIC16,TIC17</td>
<td>Temperature controllers</td>
</tr>
<tr>
<td>TY1, TY5, TY6, TY7, TY15, TY16, TY17</td>
<td>Power regulators</td>
</tr>
<tr>
<td>V1/5</td>
<td>Control valve</td>
</tr>
<tr>
<td>V1/1, V1/2, V1/3, V1/4, V2/1</td>
<td>Motor valves</td>
</tr>
<tr>
<td>V3/2, V4/4, V4/1, V4/2, V4/6, V4/5</td>
<td>Normally open valves</td>
</tr>
<tr>
<td>V5/2, V6/3, V6/4, V6/5, V8/3, V8/4</td>
<td></td>
</tr>
<tr>
<td>V3/1, V3/2</td>
<td></td>
</tr>
<tr>
<td>V5/3, V6/1, V6/2, V8/1, V8/2, V11/1</td>
<td></td>
</tr>
<tr>
<td>WE1, WE2</td>
<td>Weight cell</td>
</tr>
</tbody>
</table>

The basic function of the process is to distil off various fractions or 'cuts' from the reboiler liquid. The power input to the reboiler is manipulated in one of three different ways, depending upon the chosen control strategy. For startup purposes, the temperature
within the reboiler can be ramped up to a predetermined value using the supervisory computer. Secondly, the reboiler liquid temperature can be controlled to a specific setpoint value.

However, the strategy which is normally used to control the column is achieved by measuring the liquid flowrate descending into the reboiler, and adjusting the heater power to provide the required boilup rate. The actual flowrate is determined by periodically closing valve V2/1 and measuring the time it takes for the fluid in the level gauge to rise by a specified amount. The setpoint for this control loop is in turn adjusted in order to maintain a given reflux ratio, once the product removal rate has been specified. Unfortunately, whilst the fault simulation experiments were being performed, the column liquid flowmeter was out of service, and hence the reboiler power was adjusted manually. Despite this, the control loop was modelled as if the flowmeter was working.

The second major control action within the system, enables the column to be operated at a reduced pressure. If the condenser pressure is observed to increase above the setpoint value then the pressure control valve V1/5 opens, allowing the excess vapour to be drawn into the vacuum unit. A number of other controllers, namely TIC5, TIC6, TIC7, TIC15, TIC16, TIC17 are also present to ensure that the heat losses from the column walls, the product offtake pipework and collection vessels are minimised. All the aforementioned control loops are implemented within separate microprocessor units and networked to both the supervisory and data logging computers, via a common data bus.

Unfortunately, because of the constraints of the data logging software, only 14 out of the 47 possible process measurements (including controller setpoints and outputs etc.) were actually recorded in real time. However, those variables which did not fluctuate vary rapidly, such as the condenser cooling water inlet temperature, were also manually recorded and added to the data files at a later stage. The list of variables used within the KBS is shown in Table 7.5
Although there were two hardwired alarms associated with the process plant, namely for high reboiler liquid and condenser vapour temperatures, these signals were not accessible to the data logging computer. Furthermore, the logging computer could not be notified when the supervisory computer detected an alarm condition, based on its own internal limit values. As a consequence, the alarm detection program described in Chapter 6 was used to generate the alarm signals, based on the replayed process data.

Given that the batch distillation plant was frequently operated with different feedstocks and at different pressures, very little specific alarm limit information existed for the conditions at which the fault simulations were performed. However, because the alarm diagnosis methodology had been developed for continuous rather than batch process plant, and that the alarm strategies in each case are different, this data would have been of limited use even if it had been available.

The actual alarm limit values were therefore again derived on the basis of engineering judgement. In this case there was very little noise associated with the process measurements, but because of the gradual drift in the system composition, the alarm limit values still required careful selection. The definition of the separate alarm limits for each of the fault simulation experiments was initially considered, but on further investigation it was decided that the drift in the steady state variable values could be tolerated, whilst maintaining sensitive alarm limits. For example, throughout all the fault simulation experiments the steady state reboiler temperature only changed by 4.1 deg. C from 122.6 to 126.7 deg. C. Therefore, by setting the high and low alarm limits just outside this 4.1 deg. C bandwidth, significant deviations in the reboiler vapour temperature could still be detected.
Table 7.5  The Definitions Of The Variables Used Within The KBS

| Description               | Variable | Indication | Key in | |
|---------------------------|----------|------------|--------|
| Reboiler liquid temp.     | t1       | t11        | A      |
| Reboiler liquid temp.     | t4       | t14        | B      |
| Reboiler vapour temp.     | t3       | t13        | C      |
| Condenser vapour temp.    | t8       | t18        | H      |
| Cooling wtr. inlet temp.  | t10      | t10        | I      |
| Cooling wtr. outlet temp. | t11      | t11        | J      |
| Column top temp.          | t12      | t12        | G      |
| Column middle temp.       | t13      | t13        | F      |
| Column bottom temp.       | t14      | t14        | E      |
| Distillate offtake temp.  | t16      | t16        | K      |
| Column pressure           | p5       | p5         | N      |
| PIC5 set point            | pic5_set_pt | pic5_set_pt | O  |
| PIC5 output               | pic5_output | pic5_output | P  |
| PIC mode (auto/manual)    | pic5_mode | pic_mode  | P      |
| Vacuum pressure           | p4       | p14        | Q      |
| Receiver R2 weight        | wr1      | wr1        | L      |

The configuration of unit models used to represent the batch distillation process is shown in block diagram form in Figure 7.9. As with the reactor charging system the individual unit models are included in Appendix G.
Figure 7.9 The Block Diagram For The Batch Distillation Column

Power utilities to units 3, 7, 13, 18, 22, 24, 26, 28
29, 38, 40, 69, 70, 84
Air utility to unit 71
7.2.2 The Modelling Simplifications

For the purposes of this study the following simplifications were made during the modelling process:

1 The distillation column wall temperature controllers and the miscellaneous thermal tracing equipment were not considered. Since these devices are principally required when fluids other than the mixture of ethyl and methylbenzene are being distilled, they were considered to be an unnecessary complication.

2 Like the reactor charging system, the failure modes of the process monitoring computers, the digitisation equipment and the instrument data bus were not explicitly represented within the system configuration. As before, the instrument signals are terminated with dummy tail units.

3 The modelling of the condenser distillate removal process was simplified. In reality, the product offtake rate is controlled by a simple sequence of valve operations. Firstly, valve V1/1 is opened. This causes the condensed liquid to flow into the section of pipe bounded by valves V1/1 and V1/2, providing that both the bleed pipe and valve V4/2 are clear. After a set interval valve V1/1 is closed, trapping a known volume of liquid in the 'T' section of pipe. Finally, valve V1/2 is opened and the known quantity of liquid flows into the collection vessel R2. By controlling the rate of this valve cycle, the product withdrawal rate is fixed.

Unfortunately the standard fault tree methodology cannot easily describe and analyse sequences of events. A powerful facility has been developed within FAULTFINDER to handle certain types of events sequences, such as those associated with sequence controlled plant. However, at present it is difficult to satisfactorily apply the technique to modelling deviations in the distillate flowrate within the studied plant, as discussed by Mullhi [60]. FAULTFINDER does have the ability to model the causes
of the flow deviations in themselves, but the problem arises in attempting to relate the consequences of such disturbances back to a valve sequencing error.

Given that the distillate flowrate was not directly measured within the batch distillation plant, it was decided that the valve sequencing operation could be modelled more simply. Consequently, the two on/off motor valves (V1/1 and V1/2) have been represented by continuously variable control valves. For the same reason V1/4 is also modelled in the same way.

4 For the same reasons as discussed in Section 7.1.1, the pressure sensor model was combined with the normally open manual valve model. Additionally, the vacuum drum model was also combined with the normally closed manual valve model, in order to avoid unnecessary complication of the system configuration.

5 Although two product collection vessels existed on the process plant, only one was actually used for the duration of the fault simulation experiments. Therefore, only one collection vessel was represented within the system configuration (unit 82).

6 The pressure control loop was modelled as a trip loop system because of the asymmetric nature of its control action. During the process tests performed on the pilot plant, the pressure control valve was observed to stay closed for the majority of the time. When the system did over-pressure, the valve only opened very briefly.

The normally closed state of the pressure control valve and its infrequent opening was attributed to the self regulating behaviour of the column pressure. The pressure in a closed distillation column is a function of the heat duty supplied in the reboiler and the heat duty removed by the condenser and through heat losses. A positive heat duty imbalance will cause the system pressure to increase, a negative imbalance will cause it to decrease.
In the case of the pilot plant, the reboiler heat duty was manipulated to maintain a constant internal liquid reflux flowrate. The heat duty removed by the condenser was essentially proportional to the rate of vapour reaching the condenser. As a consequence if the reboiler duty increased, the excess heat manifesting itself in terms of a greater vapour flowrate, was removed at the same rate by the increased vapour condensation at the top of the column. The column pressure therefore did not deviate significantly as the reboiler duty was adjusted.

If the column pressure did rise as a result of inerts leaking into the system, or a significant reduction in the condenser heat transfer coefficient, the pressure could still be maintained by drawing off the excess vapour to a vacuum pump. However, if the column pressure fell because of an absence of reboiler duty, the pressure controller could not take any compensating action. Since the FAULTFINDER package assumes that a control loop has the ability to correct for both high and low deviations of the controlled variable, the trip loop model was thought to best represent the functionality of the pressure control loop.

The electrical power supply to the reboiler heating elements had to be modelled in terms of material flowrate, rather than electrical current. This analogy was used because FAULTFINDER does not currently allow variables other than the major process variables to be included in the fault propagation models. Those variables that can be considered include material flowrate, pressure gradient, absolute pressure and the ability to relieve pressure, level, temperature, composition and instrument signals. For the same reasons, the weight of material in the collection vessel was modelled in terms of level.
7.2.3 Generating The System Specific Models

Of the 28 different types of unit model required to represent the fault propagation behaviour of the batch distillation process, only nine were newly created for this application, namely:

1. The reboiler
2. The distillation column
3. The condenser
4. The vacuum unit
5. The vacuum drum
6. The electro-pneumatic converter
7. The product collection vessels
8. The combined pressure sensor/open valve
9. The bursting disc

However, it was also necessary to slightly modify many of the existing models.

The three units which required the greatest modelling effort were the reboiler, the distillation column, and the condenser. Fortunately, Kelly [61] had previously considered the problems of modelling continuous binary distillation process, and the resulting models provided a good starting point for the currently described work. However, the process of tailoring these models to describe the behaviour of the batch distillation process was not a trivial task, as discussed below.

7.2.3.1 Modelling the Condenser Unit

The column condenser was the most difficult unit to describe from a fault propagation modelling standpoint, because it effectively interfaced the distillation column model to the remainder of the process. Figure 7.10 illustrates the model port definitions; the textual model is shown overleaf.
Figure 7.10 The Condenser Model Port Definitions

Cooling water in 4 | 5 Water out
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
| | | -- 3 Vapour outlet
| | | 7 Phase |
| Vessel port 6 | | change |
| | | port |

\-------------/ | |
| | |
| Column vapour in 1 | 2 Condensate outlet

1) MODEL NUMBER NAME
302 COLUMN CONDENSER

NO. OF ENG. ASSUMPTIONS/DESCRIPTIONS: 3
NO. OF PROPAGATION EQUATIONS: 11
NO. OF EVENT STATEMENTS: 11
NO. OF DECISION TABLES: 0
NO. OF FAILURE MODES: 1

2) ENGINEERING ASSUMPTIONS AND DESCRIPTIONS

THE CONDENSER CONDENSES ALL THE INCOMING VAPOUR NORMALLY
THE LIQUID IN THE CONDENSER IS NOT SIGNIFICANTLY SUBCOOLED
THE LIQUID COMPOSITION IS ONLY A FUNCTION OF THE INLET COMP.

3) PROPAGATION EQUATIONS

R1IN=F(-P6VES)
P3OUT=F(P6VES)
G1IN=F(Q1IN)
Q2OUT=F(G1IN)
Q3OUT=F(G3OUT)
G4IN = F(Q4IN, Q5OUT)
Q5OUT = F(G4IN, G5OUT)
T5OUT = F(T4IN, G1IN)
XA6VES = F(XA1IN)
T6VES = F(T1IN)
Q7VES = F(G4IN, T4IN)

3) EVENT STATEMENTS

V Q3OUT SOME: P6VES LO
V Q3OUT REV: P6VES HI
V Q7VES LO: P6VES HI, Q2OUT LO
V Q7VES NONE: P6VES HI, Q2OUT NONE
F FOULING: Q7VES LO
V G1IN NONE: P6VES LO, T6VES LO
V XC1IN HI: XC6VES HI
V XC6VES HI: P6VES HI
V XD6VES HI: XD2OUT HI
F LK-HP-EN: XC6VES HI
F INT-LEAK: XD6VES HI, G4IN HI, Q5OUT LO

5) DECISION TABLES

N/A

6) SUPPLEMENTARY INFORMATION

NORMAL STATE: N/A

MODEL IS MULTI-COMPONENT

Under normal circumstances, the condenser unit easily has the capacity to condense all the vapour flow emanating from the distillation column. As a consequence, the column pressure will not rise significantly even if the vapour flowrate increases. The factors which will cause the condenser pressure to increase are either a reduction in the heat transfer rate to the cooling element, or the presence of non-condensable gases, such as air or nitrogen. The latter
was considered to be an important failure mode, since the system was operated at a reduced pressure. Conversely, either a total lack of vapour entering the condenser, or an excess of vapour flowing into the vacuum unit, will cause the pressure to decrease.

The rate of heat transfer is modelled using an extra internal port (number 7) in a similar way to Kelly's [61] reboiler model. The variable Q7VES effectively represents the rate at which material changes from the gaseous to the liquid phase, and is directly related to the flowrate through the cooling element and the inlet temperature of the water. The fault 'FOULING' signifies that the heat transfer surface has become fouled, which also causes the rate of heat transfer to decrease.

Unfortunately, the presence of non-condensable gases is not so easy to model. Because of the difference in behaviour between condensable and non-condensable gases, it is important to be able to distinguish between the two. However, while FAULTFINDER enables up to 20 different composition components to be represented within the fault propagation models, it is not currently possible to specify high or low flow deviations of individual components.

In order to model the effects of a high flowrate of air leaking into the column, it is therefore necessary to consider that both the mole fraction of the air component AND the overall vapour flowrate has deviated high. Unfortunately, this unduly complicates the fault tree synthesis procedure. To overcome this problem, the presence of a non-condensable gas in a sufficient abundance to cause a high pressure deviation, is simply represented by the term XC6VES HI. However, to enable standard unit models to be used downstream of the condenser, the deviation Q3OUT REV is also considered to represent a reverse flow of non-condensable gas back into the unit. A leak of air into the condenser is represented by the fault 'LK-HP-EN'.

Since the condenser normally condenses all the vapour flow, the system is in a natural state of equilibrium. Consequently, under normal circumstances the pressure control valve to the vacuum unit is very rarely opened to any degree. (This was verified experimentally by
increasing the setpoint to the pressure controller, and observing that the condenser pressure did not increase.) Since the actual pressure control loop was modelled as a trip loop within the system configuration, the column pressure is not directly related to the flow Q3OUT using propagation equations. Instead the deviation Q3OUT SOME is used to relate a low column pressure deviation to a spurious opening of the trip/control valve.

Both the temperature and the composition within the condenser model are directly related to the values of these variables at the vapour inlet port. However, since the vapour flowrate is effectively the heating medium, a total loss of vapour flow is also considered to be a cause of a low temperature deviation.

Whilst deviations in the condenser pressure will have an immediate effect on the vapour flowrate entering the unit, the pressure differential between the condenser and the distillation column will be very transient. Therefore, the condenser model does not relate deviations in the vapour input flowrate to condenser pressure deviations. Instead pressure deviations within the distillation column are related to those within the condenser using the relief variable. Condenser inlet flow deviations are solely related to faults within the distillation column.

The liquid flowrate leaving the condenser unit is normally a function of the condenser's ability to condense the vapour, and the vapour flowrate. However, if the condenser cooling water starts to leak into the condensate stream, the flowrate of liquid leaving the unit could increase dramatically. Furthermore, the temperature of the leaking fluid would be considerably lower than that of the condensate. Therefore, if the leak was of sufficient magnitude, the column temperature could plummet. In order to distinguish between a flow of condensate and leaking cooling water, the same approach to that used with non-condensable gases has been employed. In this case, the deviation XD6VES HI represents a high flowrate of leaking cooling water. The fault 'INT-LEAK' describes the source of the problem.
7.2.3.2 Modelling The Distillation Column

The batch distillation column was initially modelled in terms of three separate sub-units, one associated with each temperature measurement (T12 to T14). Whilst this approach enabled the fault propagation behaviour of the column to be represented, the resulting fault trees were not very concise. The column was therefore modelled as a single unit using the propagation equations, event statements and decision table illustrated below. As can be seen in Figure 7.11, the resulting model has seven ports, two inlet, two outlet and three vessel ports.

Figure 7.11 The Distillation Column Port Definitions

Vapour outlet 2  4 Reflux inlet
/|\  \|/
|   |
/--------\                   /---------> 5  Upper vessel port
|   |                      |   |
|   |  /---------> 6  Middle vessel port
|   |
|   |   /---------> 7  Lower vessel port
|   |
\---------/   
|     |
/\  \|/
 |
Vapour inlet 1  3 Reflux outlet

1) MODEL NUMBER   NAME
301              DISTILLATION COLUMN

NO. OF ENG. ASSUMPTIONS/DESCRIPTIONS: 3
NO. OF PROPAGATION EQUATIONS: 17
NO. OF EVENT STATEMENTS: 19
NO. OF DECISION TABLES: 1
NO. OF FAILURE MODES: 1
2) ENGINEERING ASSUMPTIONS AND DESCRIPTIONS

THE CONDENSER CONDENSES ALL THE INCOMING VAPOUR NORMALLY
THE LIQUID IN THE CONDENSER IS NOT SIGNIFICANTLY SUBCOOLED
THE LIQUID COMPOSITION IS ONLY A FUNCTION OF THE INLET COMP.

3) PROPAGATION EQUATIONS

P5VES=F(-R2OUT)
P6VES=F(-R2OUT)
P7VES=F(-R2OUT)
XA5VES=F(G4IN/Q2OUT)
XA6VES=F(G4IN/Q2OUT)
XA7VES=F(G4IN/Q2OUT)
T5VES=F(-XA5VES)
T6VES=F(-XA6VES)
T7VES=F(-XA7VES)
Q2OUT=F(G2OUT,G1IN)
G1IN=F(Q1IN,Q2OUT)
G4IN=F(Q4IN)
Q3OUT=F(G4IN)
T2OUT=F(T5VES)
T3OUT=F(T7VES)
XA2OUT=F(XA5VES)
R1IN=F(-P7VES)

3) EVENT STATEMENTS

V G1IN NONE: P5VES LO, P6VES LO, P7VES LO
V P5VES LO: T5VES LO
V P6VES LO: T6VES LO
V P7VES LO: T7VES LO
V T1IN HI: T7VES HI
V T7VES HI: T6VES HI
V T6VES HI: T5VES HI
V T5VES HI: T2OUT HI
V XC5VES HI: P5VES HI
The first three propagation equations define the causes of a pressure deviation at each of the three vessel ports. As discussed previously, the pressure within the column is principally related to the condenser pressure, by means of the relief variable. The vapour flowrate from the reboiler is obviously an important influence on the column pressure. However, this causal coupling is also a function of the status of the condenser, and therefore is not included within the propagation equations.

For example, if the reboiler vapour flowrate deviates high or low, then providing that the condenser is working normally, the column pressure should not deviate significantly. The column pressure will fall if the reboiler ceases to provide any vapour, and will rise if
the condenser stops condensing all the vapour (assuming that the trip loop does not activate). The effects of zero reboiler vapour flow are described using event statements.

Implicit within the model is the assumption that the pressure drop across the packing will not change with the vapour flowrate. In reality, if the flowrate does deviate, then the column pressures will also deviate to varying degrees, P7VES experiencing the greatest change.

The model could simply be modified to take account of this effect by including GIN HI as a cause of P7VES HI or even P6VES HI. The decision to include this relationship depends upon the significance of the pressure deviations with respect to the temperature alarm limits. For the sake of simplicity, in this study they have been discounted. The effect of a leakage of non-condensable gases is modelled in the same way as the condenser.

The composition at the vessel ports is only related to the reflux ratio, represented by the term ‘(G4IN/Q2OUT)’. In practice the column composition is also a function of the reboiler composition, but as will be discussed later, this was assumed to be constant. A complete loss of reflux will also cause the fraction of the more volatile component to decrease. However, as Kelly [61] discusses, the deviation G4IN NONE needs to be conjugated with the condition Q2OUT SOME to ensure that the causes of the fault are correctly developed.

The temperature at the vessel ports will be a function of both the pressure and the composition at those ports. However, as can be seen from the model, the pressure influence is not included within the temperature propagation equations. Whilst a low pressure deviation will cause the column temperatures to drop sharply, a high pressure deviation will have a much slower response. This is essentially because the reboiler liquid has to heat up to the new equilibrium boiling temperature. In the mean time the column temperature can actually fall because of the temporary cessation of vapour flow. The effect of a low pressure deviation is therefore described separately within three event statements.
Following an increase in the column and reboiler pressure, the high temperature deviation will start at the base of the column and propagate upwards. Within the model a high temperature deviation is therefore related to a previous cause temperature deviation further down the column. The high temperature deviation at the base of the column is then related to a high pressure deviation, through the reboiler unit model.

Finally, the effect of a high flow of leaking cooling water is described by the following three event statements:

\[
\begin{align*}
V & \text{XD5VES HI: T5VES LO} \\
V & \text{XD6VES HI: T6VES LO} \\
V & \text{XD7VES HI: T7VES LO}
\end{align*}
\]

7.2.3.3 The Reboiler Model

The vessel port definition for the reboiler model are shown in Figure 7.12, and the textual model definition is given beneath.

Figure 7.12 The Reboiler Model Port Definitions

Vapour Outlet 2 1 Reflux inlet

```
/|\  \|/  \\
|   |   |
/------------------\
|                     \\
| 4 External vessel ports |
| 3 Heat/Electrical input |
| 6 Internal |
| vessel port |
\-------------------/
1) MODEL NUMBER NAME
300 REBOILER

NO. OF ENG. ASSUMPTIONS/DESCRIPTIONS: 3
NO. OF PROPAGATION EQUATIONS: 17
NO. OF EVENT STATEMENTS: 19
NO. OF DECISION TABLES: 1
NO. OF FAILURE MODES: 1

2) ENGINEERING ASSUMPTIONS AND DESCRIPTIONS

THE CONDENSER CONDENSES ALL THE INCOMING VAPOUR NORMALLY
THE LIQUID IN THE CONDENSER IS NOT SIGNIFICANTLY SUBCOOLED
THE LIQUID COMPOSITION IS ONLY A FUNCTION OF THE INLET COMP.

3) PROPAGATION EQUATIONS

P4VES = F(-R20UT)
T4VES = F(P4VES)
T5VES = F(P4VES)
Q6VES = F(G3IN, -P4VES)
G3IN = F(Q3IN)
Q2OUT = F(G2OUT, Q6VES)
T2OUT = F(T4VES)

3) EVENT STATEMENTS

F HEAT-FL: Q6VES NONE
V P4VES HI: Q6VES NONE
V Q6VES NONE: P4VES LO, T4VES LO, T5VES LO
F LK-HP-EN: P4VES HI, X2COUT HI
V XD1IN HI: T4VES LO, T5VES LO, Q2OUT HI

4) DECISION TABLES

N/A
SUPPLEMENTARY INFORMATION

NORMAL STATE: N/A

MODEL IS MULTI-COMPONENT

This reboiler model is essentially based on Kelly's [61] partial reboiler model. However, as with the distillation column, the pressure within the unit is principally related to the column pressure, using the relief variable. Again a zero boilup rate is considered to cause a low column pressure.

The liquid and vapour temperatures, T4VES and T5VES, are normally in equilibrium, and should be functions of both the liquid composition and the unit pressure. Because the reboiler operates in a batch mode, the mole fraction of the more volatile component will gradually decrease with time. However, since the alarm diagnosis methodology has principally been developed for continuous rather than batch plant, the system is assumed to behave in a pseudo-steady state for the purposes of modelling the system. Consequently, the reboiler composition is also assumed to be in a steady state. The reboiler temperatures are therefore solely related to the unit pressure.

The boilup rate is modelled in the same way as the condensation rate within the condenser, using the dummy variable Q6VES. The fourth propagation equation relates deviations in this variable to both the heater power input G31N and the column pressure. Note that the column pressure is included even though it only has a transitory effect on the boilup rate. The failure mode 'HEAT-FAL' represents a failure of the heating element.

Finally, the vapour flowrate leaving the reboiler is modelled as a function of both the downstream pressure gradient and the boilup rate. However, if the liquid inlet stream becomes saturated with leaking cooling water, this is assumed to flash in contact with the
reboiler contents and cause a high flow of a steam mixture. Eventually, this leaking cooling water is considered to cool down the reboiler temperature.

7.2.3.4 Creating The Remaining Models

The remaining units, namely:

4 The vacuum unit
5 The vacuum drum
6 The electro-pneumatic converter
7 The product collection vessels
8 The combined pressure sensor/open valve
9 The bursting disc

were not particularly difficult to model. The vacuum unit was based on a dummy tail unit, with the addition of the following event statement:

F VAC-LOSS:R1IN NONE,G1IN NONE

The failure mode 'VAC-LOSS' representing the vacuum units inability to draw in vapour.

The vacuum drum model was essentially a modified tank model, with an extra failure mode 'HV-D-OP', representing the erroneous opening of the vacuum drum vent valve. The collection vessel model was very similar, except that the manual valve fault was considered within a separate unit.

The electro-pneumatic converter model was virtually a copy of the standard forward acting controller model, the main difference being the removal of the setpoint input port, and the substitution of unique failure mnemonics. The combined pressure sensor / open manual valve model was created in the same way as the level sensor / open manual valve model, described in Section 7.1.2. Finally, the bursting disc model was developed so that the fault 'DIS-FAIL' would cause the reverse flow of non-condensable gases back into the condenser unit.
The FAULTFINDER suite of programs are a powerful collection of tools for improving the speed and efficiency of the process modelling task. This should be enhanced even further as the library of standard models expands to take account of any additional modelling experience gained. Despite the obvious strengths of the package, a number of modelling difficulties were identified whilst the two example systems were being modelled. These have been classified into three groups, those associated with the implementation of the methodology (the software), the methodology itself, and the fault tree representation technique in general. The following text discusses these problems in more detail.

The main limitation of the FAULTFINDER software is the restriction on the number of separate units allowed within a system configuration. At present only 100 individual units can be considered. The upper limit was imposed because the computers on which the software had previously been implemented were limited in memory capacity. As a general rule, the larger the system configuration becomes, so the resulting fault trees become larger. The number of units was therefore restricted so that the software would execute on such machines.

The alarm fault trees which are used to derive the diagnosis rules, are only required to relate the causes of one alarm to deviations in the nearest alarmed variable. Consequently, only those units connecting the diagnosed alarm to the cause alarms are actually required to generate the appropriate fault tree. However, building a new system configuration for each alarm is clearly time consuming, so for convenience the whole system is usually modelled in total. As is demonstrated by the batch distillation example, it is not difficult to approach using 100 units when modelling quite a small system. Therefore it would still be a time consuming task to model a realistically sized section of process plant.
Since FAULTFINDER is now implemented on a computer which is not restricted by physical memory size, the historical restrictions on configuration size no longer apply. Therefore this upper limit could be increased. Furthermore, if it were possible to specify 'break points' in the system configuration, so that the fault trees were not developed past these points, then the alarm fault tree synthesis procedure would be greatly eased. It is not envisaged that the implementation of either of these two modifications would require a large amount of programming effort.

Most of the modelling difficulties arose because of the limitations or ill defined aspects of the fault tree synthesis methodology. The most trivial of these problems was the restriction on the types of variable that could be represented. Therefore, whilst modelling the batch distillation plant it was necessary to describe the heater power in terms of mass flowrate, and the collection vessel weight in terms of level. Furthermore, it would also have been more convenient if it had been possible to model flowrates of individual components within the distillation column. Clearly this is not a significant problem, but it is an area for future work.

Another limitation of the FAULTFINDER fault tree synthesis methodology, from the point of generating the alarm diagnosis information, is the restriction on the number of variable deviation states that can be modelled. For example, the permissible deviation states of the major process variables, pressure, flowrate, temperature and composition are shown in Table 7.6.

Most of the deviation states are self explanatory, however, in the case of the pressure variable, the state REV signifies a large pressure loss, and the NOR state represents a lack of upstream pressure relief. A more detailed list of the process variables and their deviation states is included in reference [59].

208
Table 7.6 The Permissible Variable Deviation States

<table>
<thead>
<tr>
<th>Deviation</th>
<th>Pressure</th>
<th>Flowrate</th>
<th>Temperature</th>
<th>Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HI</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>LO</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>NONE</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SOME</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REV</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NOR</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Since a very low deviation state could not be represented within FAULTFINDER, the fault trees for the very low pressure, flow and level alarms of both systems were modelled using the NONE deviation state. Unfortunately, a NONE deviation state is not defined for the temperature variable, therefore it was difficult to differentiate between low and very low temperature deviations in the batch distillation column, from the modelling standpoint.

The limited number of deviation states also caused difficulties in correctly developing the causes of a low buffer tank inlet flow alarm. As described in Section 7.1.2, a high buffer tank level was not considered as a cause of a low inlet flow alarm, because the integrating action of the control loop would have driven the control valve completely closed, long before a high level alarm was reached.

However, in reality if the tank discharge flow deviates low, this will cause a slight rise in the tank level. When the control loop corrects for the level increase, the inlet flow will decrease to match the outlet flow, which may result in an upstream low flow alarm. Unfortunately, because of the modification of the controller model, this fault propagation mechanism cannot be traced by FAULTFINDER, unless the low discharge flow deviation is modelled as a cause of a high tank level.
Finally, as can be seen from Appendix I, failures of either the electrical pipeline tracer or of the process power supply, are not listed as causing the high or low pump discharge pressure alarm. This is because it is assumed that the effects of the ethanoic acid freezing upstream or downstream of the pressure sensor will be self compensating. However, this information is difficult to represent within the fault propagation models and therefore the high and low pressure alarm fault trees contain these freezing faults.

As was briefly discussed in Section 4.2, one of the most limiting aspects of the fault tree technique is the difficulty of representing features such as time delay and the sequencing of events. For example, a high pressure deviation within the batch distillation column will initially cause the column temperatures to fall. However, after a short time the column temperatures will deviate higher than their original values. A high column pressure may therefore cause both low and high temperature alarms.

Whilst it is possible to represent both the short and long term effects of the pressure deviation, it is difficult to clearly portray their sequence in time using the fault tree technique. Despite this problem it was still possible to derive valid alarm diagnosis rules for the consequences of a high column pressure deviation by simply ignoring their sequence. The temporal aspects are then resolved when the diagnosis procedure uses the information. This issue is discussed in more detail in Chapter 9.

Whilst many of the problems described above were annoying, most of them were not caused by any fundamental limitations of the fault tree methodology. Rather they were due to the historical evolution of the FAULTFINDER code or 'grey areas' within the fault tree synthesis methodology. Given the resources it is expected that these difficulties could be overcome.
The previous chapter described how the alarm fault tree information was generated for the two studied systems. This chapter describes how the diagnostic rules, derived from this fault tree information, are used in conjunction with the alarm combination rules and ranking procedures (discussed in Chapters 4 and 5) to diagnose alarms. The methodology is illustrated by working through four example fault scenarios.

8.1 The Reactor Charging System

To assist with the explanations, the schematic diagram for the reactor charging system is shown in Figure 8.1.

8.1.1 Example 1: Loss Of Fluid From The Buffer Tank T2

Within the reactor charging system the buffer tank is a key piece of equipment because it smoothes out fluctuations in the upstream flowrate, and provides a constant head of fluid for the reactor. A loss of fluid from the tank can be caused by many failures, such as a leakage in the unit itself or in the discharge pipework. However, one of the most likely causes is the accidental opening of the tank drain valve. The following example is therefore based on the simulated consequences of this fault.
Figure 8.1 The Reactor Charging System
A loss of buffer tank fluid, due to 75% of the normal inlet flow leaking through valve 7, was simulated using the dynamic model of the reactor charging system. The following alarms were detected relative to the introduction of the fault:

1. Indication buf_acd_fl_1 alarms high during scan 2; (tank inlet flow alarms high as measured by transducer 1)
2. Indication buf_acd_fl_2 alarms high during scan 2; (tank inlet flow alarms high as measured by transducer 2)
3. Indication buf_tnk_lev alarms low during scan 11; (tank level alarms low)
4. Indication reac_acd_fl alarms low during scan 23; (tank outlet flow alarms low)
5. Indication buf_tnk_lev alarms very low during scan 45. (tank level alarms very low)

Note that the reactor charging system was monitored with a scan interval of 10 seconds.

8.1.1.1 Diagnosing The High Tank Inlet Flow Alarm

When either of the first two alarms is diagnosed, the KBS will check to determine if the process variable deviation represented by the diagnosed alarm, in this case flow f1 → high, is also represented by any other alarms. For example, if the 'buf_acd_fl_1 high alarm' is considered, then the status of the 'buf_acd_fl_2 high alarm' will also be checked, since they represent the same process variable deviation. In this example the two alarms agree, therefore they will be diagnosed together. If they did not agree, the spurious failure of the disagreeing alarm would be combined with the causes of the diagnosed alarm, as discussed in Chapter 4.
Having checked the alarms for consistency, the KBS then checks any previously diagnosed alarms which are still active, to determine if they are causally related to the latest alarm. In this example the high flow alarms are the first two to occur, therefore they are considered in isolation.

Using the reactor charging system rulebase information, included in Appendix I, the following ranked list of alarm causes will be generated:

1. device valve_7 fails opened @ 2 and control_loop cl_1 working @ 2 (95.784%)
2. device tank_2 fails leakage @ 2 and control_loop cl_1 working @ 2 (2.21%)
3. control_loop cl_1 fails high @ 2 (1.4743%)
4. indication buf_acd_fl_1 spuriously_alarms high @ 2 AND (0.5157%) indication buf_acd_fl_2 spuriously_alarms high @ 2
5. device valve_6 fails leakage @ 2 and control_loop cl_1 working @ 2 (0.0147%)
6. flow f2 => high @ 2 and control_loop cl_1 working @ 2 (0.00028%)
7. device pipe_6 fails leakage @ 2 and control_loop cl_1 working @ 2 (0.000147%)
8. device pipe_5 fails leakage @ 2 and control_loop cl_1 working @ 2 (0.000147%)
9. signal set_pt => high @ 2 and control_loop cl_1 working @ 2 (0%)

Note that all the alarm causes are tagged with the scan number during which the alarm was detected.
As can be seen, the erroneous opening of the tank drain valve (valve 7) has by far the greatest likelihood (shown in parentheses). This is due to the assumption that the maloperation of the process, as a result of human error, will be more likely than the spontaneous failure of the process equipment. For the purposes of the study, the human failure frequency was based on the estimation that an operator will make ten erroneous actions during a forty year working lifetime. This equates to 130 faults/10^6 hours.

Despite the fact that the buffer tank inlet flow is manipulated by the level controller, low or very low level deviations are not included in the above list. This is because many of the causes of these deviations can, in their less severe forms, also cause the upstream flow to deviate high, without the level deviating sufficiently low to trigger an alarm. The faults are therefore directly related to the high flow alarms, with the condition that the level controller is functioning correctly. This is to ensure that the logical consistency of the alarm explanations is maintained later.

8.1.1.2 Diagnosing The Low Buffer Tank Level Alarm

When the low buffer tank level alarm occurs in scan 11, the KBS will detect that the new alarm is causally related to the earlier alarms. This is because the alarms share many common failure modes and coexist at the same point in time. The causes of the low buffer tank level alarm include explanations 1, 2, 5, 6, 7, and 8 from the previous list (without the control loop working condition). However, the following list of faults, incompatible with or unrelated to the first two alarms, also cause the low level alarm:

1. flow f1 => low @ 11
2. flow f1 => very low @ 11
3. indication buf_tank_lev spuriously_alarms low @ 11

215
device valve_6 fails opened and control_loop cl_1 fails normal @ 11

pressure p2 \(\rightarrow\) low and control_loop cl_1 fails normal @ 11

The control loop fails normal fault is meant to describe the control loop failing invariant in its normal operational state.

Since there are less than ten different causes of the high flow alarms, all these faults are combined with the causes of the low level alarm. A brief description of how the combined alarm causes are derived is given in the following text. The discussion follows the order of the causes of the high flow alarms.

The failure mode 'device valve_7 fails opened @ 2 and control_loop cl_1 working @ 2' from the causes of the high flow alarms, will remain unchanged when combined with the cause 'device valve_7 fails opened @ 11' of the low level alarm. This is because when the two cutsets are ANDed and then simplified, the duplicate failure mode of valve 7 will be removed. Since the faults are associated with different scan intervals, the earliest occurrence of the valve_7 fault is retained. As a consequence, when the cutset 'device valve_7 fails opened @ 2 and control_loop cl_1 working @ 2' is combined with any other causes of the low level alarm, all the resulting cutsets will be non-minimum.

Using the same reasoning the following cutsets for the three alarms can also be shown to be minimum:

1. device tank_2 fails leakage @ 2 and control_loop cl_1 working @ 2
2. device valve_6 fails leakage @ 2 and control_loop cl_1 working @ 2
3. flow f2 \(\rightarrow\) high @ 2 and control_loop cl_1 working @ 2
4. device pipe_6 fails leakage @ 2 and control_loop cl_1 working @ 2
5. device pipe_5 fails leakage @ 2 and control_loop cl_1 working @ 2
The 'control_loop cl_1 fails high @ 2' fault clearly does not appear as a common cause of all three alarms. Therefore, it must be ANDed with each of the causes of the low level alarm. However, six of the eleven resulting cutsets can be rejected on the basis that they are non-minimum. These cutsets are listed below:

1. device valve_7 fails opened @ 2 and
   control_loop cl_1 fails high @ 11

2. device tank_2 fails leakage @ 2 and
   control_loop cl_1 fails high @ 11

3. device valve_6 fails leakage @ 2 and
   control_loop cl_1 fails high @ 11

4. flow f2 => high @ 2 and
   control_loop cl_1 fails high @ 11

5. device pipe_6 fails leakage @ 2 and
   control_loop cl_1 fails high @ 11

6. device pipe_5 fails leakage @ 2 and
   control_loop cl_1 fails high @ 11

The cutsets are non-minimum because they are superset of the six minimum cutsets described previously; providing that the 'control_loop cl_1 working' condition is ignored. Since this extra condition is only included in the cutsets to ensure that logical consistency is maintained, this is a valid assumption.

When the failure mode 'control_loop cl_1 fails high @ 2' is combined with either of the two causes 'flow f1 => low @ 11' or 'flow f1 => very_low @ 11', the resulting cutsets will also be rejected. This is because both the causes of the low level 12 will conflict with the assumed high state of flow f1, caused by the control loop failing high. However, when the control loop failure mode is combined with the spurious failure of the level alarm, the following valid cutset will be generated:
There are three causes of the control loop failing high, namely:

1. The control valve cv_l failing high (high aperture);
2. The controller cnt_l failing high output;
3. The level sensor ls_2 failing low.

but only one cause of the level alarm spuriously indicating a low level, namely level sensor ls_2 fails low. If the two sets of causes are conjugated, it can be seen that only one minimum cutset will therefore result, the spontaneous low failure of level sensor ls_2.

Finally, when the 'control loop cl_l fails high' fault is combined with the last two causes of the low level alarm, the cutsets will be rejected. This is because the same control loop cannot be failed in its normal output state and a high state, if the alarms caused by those two control loop failure modes coexist at the same point in time.

The minimum cutsets that are generated when the fault 'indication buf_acd_fl_l spuriously_alarms high @ 2 and indication buf_acd_fl_2 spuriously alarms high @ 2' are ANDed with the causes of the low buffer tank level alarm, are listed below:

1. indication buf_acd_fl_l spuriously_alarms high @ 2 and indication buf_acd_fl_2 spuriously_alarms high @ 2 and flow fl => low @ 11
2. indication buf_acd_fl_l spuriously_alarms high @ 2 and indication buf_acd_fl_2 spuriously_alarms high @ 2 and flow fl => very_low @ 11
Because the spurious failure of level sensor ls_2 is a minimum cutset, the combination of all three indications spuriously alarming is rejected on the grounds of minimality.

The last cause of the high inlet flow alarms, 'signal set_pt => high @ 2 and control_loop cl_1 working @ 2' does not produce any minimum cutsets when combined with the causes of the low level alarm. The cutsets are rejected because they are either non-minimum or they require mutually exclusive states of the buffer tank inlet flow or the control loop, to coexist at the same point in time.

The eleven minimum cutsets for the three alarms, ranked according to likelihood, are listed below:

1. device valve_7 fails opened @ 2 and (97.5%)
   control_loop cl_1 working @ 2

2. device tank_2 fails leakage @ 2 and (2.25%)
   control_loop cl_1 working @ 2

3. device ls_2 fails low @ 2 (0.233%)

4. device valve_6 fails leakage @ 2 and (0.015%)
   control_loop cl_1 working @ 2

5. flow f2 => high @ 2 and (0.0003%)
   control_loop cl_1 working @ 2
device pipe_6 fails leakage @ 2 and (0.00015%)
control_loop cl_1 working @ 2

device pipe_5 fails leakage @ 2 and (0.00015%)
control_loop cl_1 working @ 2

indication buf_acd_fl_1 spuriously_alarms high @ 2 and (6.4x10^{-10} %)
indication buf_acd_fl_2 spuriously_alarms high @ 2 and
flow fl => very_low @ 11

indication buf_acd_fl_1 spuriously_alarms high @ 2 and (7x10^{-11} %)
indication buf_acd_fl_2 spuriously_alarms high @ 2 and
device valve_6 fails opened @ 11 and
control_loop cl_1 fails normal @ 11

indication buf_acd_fl_1 spuriously_alarms high @ 2 and (4x10^{-12} %)
indication buf_acd_fl_2 spuriously_alarms high @ 2 and
flow fl => low @ 11

indication buf_acd_fl_1 spuriously_alarms high @ 2 and (1x10^{-19} %)
indication buf_acd_fl_2 spuriously_alarms high @ 2 and
pressure p2 => low @ 11 and
control_loop cl_1 fails normal @ 11

As can be seen, the causes of the first three alarms are very similar to the causes of the high buffer tank inlet flow alarms. This, however, is not surprising given the large number of common failure modes. Another consequence of the large overlap in failure modes is the degree to which the raw cutsets for the three alarms have been reduced. Since there are nine causes of the high flow alarms and eleven causes of the low level alarm, there are ninety nine raw cutsets for the three alarms. By checking for minimality and logical consistency, these ninety nine cutsets have been reduced to just eleven.
Looking at the likelihoods of the last four alarm explanations, it is clear that explanations involving multiple independent failure modes are extremely unlikely, if the same pattern of alarms can be explained in terms of single failure modes.

8.1.1.3 Diagnosing The Low Tank Discharge Flow Alarm

The causes of a low tank discharge flow alarm are listed below:

1. level 12 -> low @ 23
2. device downstream_tracer fails off @ 23
3. pressure p2 -> high @ 23
4. reactor_feed_section fails partial_blockage @ 23
5. device valve_6 fails leakage @ 23
6. device pipe_6 fails leakage @ 23
7. indication reac_acd_fl spuriously_alarms low @ 23

Note that the cutset number 4 represents the following four failure modes:

1. device pipe_6 fails partial_blockage @ 23
2. device pipe_7 fails partial_blockage @ 23
3. device valve_6 fails partial_blockage @ 23
4. device valve_8 fails partial_blockage @ 23
When the above alarm occurs in scan 23, it will be related to the diagnosis of the other three alarms, since this latest alarm could be both a consequence of a low tank level, or be caused by some of the same failure modes.

Before the causes of the previous alarms are combined with those of the latest alarm, the ten most likely explanations for the three alarms are extracted from the list of eleven. As discussed in Chapter 4, this is to minimise the problems of combinatorial explosion. However, to ensure that the rejected cutset does not contain any important information, its contents are checked against the causes of the new alarm. If any common failure modes are detected, the cutset is not rejected. In this example there are no common failure modes, therefore the cutset is disregarded.

For exactly the same reasons as outlined in the previous section the alarm causes 'device pipe_6 fails leakage @ 2 AND control_loop cl_1 working @ 2' and 'device valve_6 fails leakage @ 2 AND control_loop cl_1 working @ 2' will be determined as minimum cutsets for all four alarms. In addition, because a low buffer tank level is a direct cause of a low discharge flow alarm, any of the causes of the first three alarms which will actually cause a low level, will in theory also cause the fourth alarm. Therefore, when these cutsets are conjugated with the first cause of the low discharge flow alarm, 'level 12 \rightarrow low @ 23', they will be simplified by removing the deviation state of the level variable.

Of all the explanations for the first three alarms, only one will not cause a low level in the buffer tank, namely the level sensor failing low fault. However, one of the causes of the low level is a high flow of material through the tank discharge pipeline, described as:

\[ \text{flow } f_2 \rightarrow \text{high } @ 2 \text{ and control_loop cl_1 working } @ 2 \]

If the above fault does cause the buffer tank level to deviate low enough to trigger the alarm, then it is unlikely that the low tank level will then cause the discharge flow to become low. This cutset is
therefore not considered to be a valid cause of the low discharge flow alarm. The KBS detects that the cutset is invalid because the state of the flow f2 variable in scan 2 conflicts with its assumed state in scan 23.

For similar reasons, if valve_6 is accidentally opened fully and the control loop has failed in an invariant state, the buffer tank level will fall to a new equilibrium level, where the inlet flow rate matches the outlet flow rate. The discharge flow rate will therefore increase initially, and then decrease back to match the inlet flow rate. The following cutset, which is both a cause of the first three alarms and a low buffer tank level, is therefore not deemed to be a valid cause of a low tank discharge flow rate.

indication buf_acd_fl_1 spuriously_alarms high @ 2 and indication buf_acd_fl_2 spuriously_alarms high @ 2 and device valve_6 fails opened @ 11 and control_loop cl_l fails normal @ 11

Unfortunately, if this cutset is conjugated with the first cause of the low tank discharge flow alarm, the KBS cannot detect that the fault scenario is invalid, solely from the information contained within the cutset. In order to overcome this problem the system modeller must specify in the 'a priori' causes of the low tank level, that if valve 6 is opened fully and the control loop has failed invariant in its normal state, the tank discharge flow will not deviate low. This information is then stored as a boundary condition within the cutset.

Whilst the problem is easy to resolve once the logical inconsistency has been identified, the task of detecting the inconsistency from the 'a priori' information still demands skill on the part of the system modeller. In order to ease this task it is thought that the rulebase compiler could be enhanced to identify potential problem areas. For example, the cause of the low tank level, valve 6 fails open and control loop cl_l fails normal, could be compared with the causes of the low tank discharge flow. If the fault
valve 6 failed closed was included as a cause of the flow deviation, the opposite valve failure modes could cause the compiler to prompt the user for clarification.

When the level sensor fault 'device ls_2 fails low @ 2' is combined with the causes of the fourth alarm, the following four minimum cutsets will be generated:

1. device ls_2 fails low @ 2 AND
device downstream_tracer fails off @ 23

2. device ls_2 fails low @ 2 AND pressure p2 -> high @ 23

3. device ls_2 fails low @ 2 AND
reactor_feed_section fails partial_blockage @ 23

4. device ls_2 fails low @ 2 AND
indication reac_acd_fl spuriously_alarms low

On closer inspection it can be seen that the first three cutsets contain potentially compensating faults with regard to the low discharge flow alarm. For example, in the case of explanation 1, if device ls_2 fails low, the control loop will open the inlet control valve fully allowing 120% of normal flow into the buffer tank. If the tank discharge pipe is partially constricted by solid ethanoic acid, the buffer tank level will either rise until the outlet flowrate matches the inlet flowrate, or the tank overflows. Whether or not these three cutsets will actually cause all four alarms to activate is therefore unclear. However, the lack of clarity can be tolerated, since these cutsets will have very low frequencies of failure.

Finally, when the following two cutsets are combined with the causes of the fourth alarm:

1. flow f2 -> high @ 2 AND control_loop cl_1 working
Only two valid minimum cutsets will result, namely:

1. \( \text{flow } f_2 \rightarrow \text{high} @ 2 \) AND \( \text{control_loop } c_1 \_l \) working AND \( \text{indication } reac\_acd\_fl \) spuriously alarms low

2. \( \text{indication } buf\_acd\_fl\_1 \) spuriously alarms high @ 2 AND \( \text{indication } buf\_acd\_fl\_2 \) spuriously alarms high @ 2 AND \( \text{control_loop } c_1 \_l \) fails normal @ 11 AND \( \text{device valve}_6 \) fails opened AND \( \text{indication } reac\_acd\_fl \) spuriously alarms low

The majority of the generated cutsets are rejected because the two causes of the first three alarms presuppose that the tank discharge flow is not in a low state. Therefore all the causes of the fourth alarm which will actually result in a low discharge flow, are rejected on the basis of conflicting discharge flow states.

The thirteen ranked causes of the four alarms are listed below:

1. \( \text{device valve}_7 \) fails opened @ 2 AND \( \text{control_loop } c_1 \_l \) working @ 2 (97.73%)

2. \( \text{device tank}_2 \) fails leakage @ 2 AND \( \text{control_loop } c_1 \_l \) working @ 2 (2.25%)

3. \( \text{device valve}_6 \) fails leakage @ 2 AND \( \text{control_loop } c_1 \_l \) working @ 2 (0.015%)

4. \( \text{device pipe}_6 \) fails leakage @ 2 AND \( \text{control_loop } c_1 \_l \) working @ 2 (0.00015%)

5. \( \text{device pipe}_5 \) fails leakage @ 2 AND \( \text{control_loop } c_1 \_l \) working @ 2 (0.00015%)

225
device ls_2 fails low @ 2 AND (2.7x10^{-7}%) 
indication reac_acd_fl spuriously_alarms low @ 23

device ls_2 fails low @ 2 AND (5.5x10^{-8}%) 
reactor_feed_section fails partial_blockage @ 23

indication buf_acd_fl_1 spuriously_alarms high @ 2 AND (6x10^{-10}%) 
indication buf_acd_fl_2 spuriously_alarms high @ 2 AND
flow f1 => very_low @ 11

flow f2 => high @ 2 AND (5x10^{-11}%) 
indication reac_acd_fl spuriously_alarms low @ 23 AND
control_loop cl_1 working @ 2

indication buf_acd_fl_1 spuriously_alarms high @ 2 AND (4x10^{-12}%) 
indication buf_acd_fl_2 spuriously_alarms high @ 2 AND
flow f1 => low @ 11

device ls_2 fails low @ 2 AND (4x10^{-13}%) 
pressure p2 => high @ 23

device ls_2 fails low @ 2 AND (3x10^{-14}%) 
downstream_tracer fails off @ 23

indication buf_acd_fl_1 spuriously_alarms high @ 2 AND (4x10^{-16}%) 
indication buf_acd_fl_2 spuriously_alarms high @ 2 AND
device valve_6 fails opened @ 11 AND
control_loop cl_1 fails normal @ 11 AND
indication reac_acd_fl spuriously_alarms low @ 23

8.1.1.4 Diagnosing The Very Low Buffer Tank Level Alarm

Finally, the causes of the very low buffer tank level alarm are as listed overleaf:
As can be seen from its list of causes, the fifth alarm is not a direct consequence of any of the three previously alarmed variable deviations. However, the latest alarm does share common failure modes with at least one of the previous alarms. Therefore, the causes of the very low buffer tank level alarm are conjugated with those of the other alarms.

Since there are more than ten causes of the first four alarms, the three most unlikely explanations are scrutinised, to determine if they share any common failure modes with the fifth alarm. In this example they do not, so they are rejected.

Not surprisingly, the five most likely causes of the previous alarms, are also direct causes of the very low level alarm. These faults are therefore minimum cutsets for all five alarms. Similarly, explanations 8 and 9 are minimum cutsets because they remain unchanged when they are conjugated with the very low alarm causes 'flow f1 -> very_low @ 45' and 'flow f2 -> high @ 45' respectively.
All the explanations for the first four alarms which contain the fault 'device ls_2 fails low', will generate logically inconsistent cutsets when conjugated with the causes of the latest alarm. This is simply because the explanations for the fifth alarm assume that the level sensor is either working or failed in a very low state.

The KBS identifies that the cutsets are invalid by two means:

1. The list of process variable deviation states caused by the elements within each cutset is checked. The causes of the very low buffer tank level will, by definition, cause the variable state 'level 12 -> very_low'. However, the level sensor fails low fault will not directly cause any deviation in the level variable (only indirectly through the action of the control loop). The state of the buffer tank level associated with the sensor fault is therefore 'level 12-> unknown'. When the inconsistent faults appear in the same cutset, the list of assumed variable states will include both 'level 12 -> unknown' and 'level 12 => very_low'. Since the former implies that the level sensor has failed, and the latter that the sensor is working, the cutset is rejected.

2. The alarm causes are checked to ensure that mutually exclusive states of the same device do not coexist within the same cutset. Any cutsets which include 'device ls_2 fails low' and 'indication buf_tnk_lev spuriously alarms very_low' are therefore rejected.

Finally, when the tenth explanation for the first four alarms is combined with the causes of the latest alarm, only the two minimum cutsets will be generated. These appear as cutsets 9 and 10 in the ranked explanations for all five alarms, as listed below:

1. device valve_7 fails opened @ 2 AND (97.72%)
   control_loop cl_1 working @ 2

2. device tank_2 fails leakage @ 2 AND (2.26%)
   control_loop cl_1 working @ 2
device valve_6 fails leakage @ 2 AND (0.015\%)
control_loop cl_1 working @ 2

device pipe_6 fails leakage @ 2 AND (0.00015\%)
control_loop cl_1 working @ 2

device pipe_5 fails leakage @ 2 AND (0.00015\%)
control_loop cl_1 working @ 2

indication buf_acd_fl_1 spuriously_alarms high @ 2 AND (6x10^{-10}\%)
indication buf_acd_fl_2 spuriously_alarms high @ 2 AND
flow f1 => very_low @ 11

flow f2 => high @ 2 AND (5x10^{-11}\%)
indication reac_acd_fl spuriously_alarms low @ 23 AND
control_loop cl_1 working @ 2

indication buf_acd_fl_1 spuriously_alarms high @ 2 AND (7x10^{-18}\%)
indication buf_acd_fl_2 spuriously_alarms high @ 2 AND
flow f1 => low @ 11 AND
indication buf_tnk_lev spuriously_alarms very_low @ 45

indication buf_acd_fl_1 spuriously_alarms high @ 2 AND (7x10^{-22}\%)
indication buf_acd_fl_2 spuriously_alarms high @ 2 AND
flow f1 => low @ 11 AND
flow f2 => high @ 45

As can be seen from the above list, the accidental opening of valve 7 (the buffer tank drain valve) still remains the favourite cause for all five alarms. It is important to note, however, that from the alarm symptoms alone, it is impossible to distinguish between any of the nine explanations. This example therefore illustrates the benefits of using experiential knowledge (in this case in the form of 'a priori' failure rate frequencies) to direct the process operators attention to the most probable cause of a group of related alarms.
Another interesting feature of this example is the way in which the interpretation of many of the faults can change as the pattern of alarms develops. For example, a minor leakage in the buffer tank, (16% of normal flow) will be sufficient to cause the high inlet flow alarms to activate. However, in order to explain both the high flow alarms and the low level alarm, the tank leakage must be at least 27% of the normal inlet flow. Similarly, the tank leakage must be at least 50% of the normal flow into the tank, in order for the very low level alarm to trigger. The same failure mode 'tank_2 fails leakage' is used to describe all three severities of leakage.

8.1.2 Example 2: Control Valve Cv_1 Fails Closed And Flow Transducer Trans_1 Fails Invariant

The previous example illustrates how the KBS can diagnose the alarms caused by a single process equipment failure. This second example from the reactor charging system demonstrates how the KBS can also diagnose certain types of multiple equipment failures.

The consequences of both the control valve failing closed (very low) and the flow transducer trans_1 failing invariant (normal) were simulated, with the following results:

1 indication pump_pl_pres alarms high during scan 1

2 indication buf_acd_fl_2 alarms very low during scan 1
   (indication buf_acd_fl_1 reads normal during scan 1)

3 indication buf_tnk_lev alarms low during scan 6

4 indication reac_acd_fl alarms low during scan 7

5 indication buf_tnk_lev alarms very low during scan 20

6 indication reac_acd_fl alarms very_low during scan 37
If the high pump discharge pressure alarm is diagnosed first, the following ranked list of explanations will be generated:

1. indication pump_pl_pres spuriously_alarms high @ 1 (98.9%)
2. pipeline_section fails large_blockage @ 1 (0.8%)
3. control_loop cl_1 fails very_low @ 1 (9.34x10^{-2} %)
4. device pipe_5 fails large_blockage @ 1 (7.471x10^{-2} %)
5. flow f2 \rightarrow very_low @ 1 AND (2.52x10^{-2} %)
   control_loop cl_1 working @ 1
6. level l1 \rightarrow high @ 1 (5.6x10^{-3} %)
7. signal set_pt \rightarrow low @ 1 AND (0%)
   control_loop cl_1 working @ 1

Rather untypically, the 'control loop fails very low' fault has a small failure frequency. The subsystem fault can be caused by either the level sensor failing high, the controller failing very low or the control valve failing very low. However, because the sensor and controller outputs are observable by the KBS, and are not high and very low respectively when the alarm occurs, the failure of these two devices is effectively discounted. Similarly, the controller setpoint signal is observed to be not low when the alarm occurs, and as a result it has a zero likelihood.

The remaining causes of the high pressure fault are also very infrequent, therefore the spurious failure of the pressure sensor has by far the greatest likelihood.
8.1.2.1 Diagnosing The Causes Of The Very Low Flow Alarm

The very low buffer tank inlet flow alarm shares many common failure modes with the high pressure alarm, and therefore will be diagnosed in conjunction with it. The causes of the second alarm are listed below:

1. device upstream tracer fails off @ 1 AND
   indication buf_acd_fl_l does_not_alarm very_low @ 1

2. level l1 => very_low @ 1 AND
   indication buf_acd_fl_l does_not_alarm very_low @ 1
   (pressure pl is_not_in_state high @ 1)

3. flow f2 => very_low @ 1 AND control_loop cl_1 working @ 1 AND
   indication buf_acd_fl_l does_not_alarm very_low @ 1

4. signal set_pt => low @ 1 AND control_loop cl_1 working AND
   indication buf_acd_fl_l does_not_alarm very_low @ 1

5. control_loop cl_1 fails very_low @ 1 AND
   indication buf_acd_fl_l does_not_alarm very_low @ 1

6. tank_section fails large_blockage @ 1 AND
   indication buf_acd_fl_l does_not_alarm very_low @ 1
   (pressure pl is_not_in_state high @ 1)

7. pump_section fails large_blockage @ 1 AND
   indication buf_acd_fl_l does_not_alarm very_low @ 1
   (pressure pl is_not_in_state high @ 1)

8. pipeline_section fails large_blockage @ 1 AND
   indication buf_acd_fl_l does_not_alarm very_low @ 1

9. device pipe_5 fails large_blockage @ 1 AND
   indication buf_acd_fl_l does_not_alarm very_low @ 1
10 tank_section fails leakage @ 1 AND
   indication buf_acd_fl_1 does_not_fail very_low @ 1
   (pressure pl is_not_in_state high @ 1)

11 pump_section fails leakage @ 1 AND
   indication buf_acd_fl_1 does_not_alarm very_low @ 1
   (pressure pl is_not_in_state high @ 1)

12 pipeline_section fails leakage @ 1 AND
   indication buf_acd_fl_1 does_not_alarm very_low @ 1
   (pressure pl is_not_in_state high @ 1)

13 device pump_pl fails stopped @ 1 AND
   indication buf_acd_fl_1 does_not_alarm very_low @ 1
   (pressure pl is_not_in_state high @ 1)

14 upstream_valves fails closed @ 1 AND
   indication buf_acd_fl_1 does_not_alarm very_low @ 1
   (pressure pl is_not_in_state high @ 1)

15 indication buf_acd_fl_2 spuriously_alarms very_low @ 1

The grouped failure mode 'tank_section' represents devices valve_2, valve_3 and pipe_2; 'pump_section' represents devices valve_4 and pump_1; 'upstream_valves' represents valve_2 and valve_3; 'pipeline_section' represents pipe_3, pipe_4 and control valve cv_1.

As can be seen, all the causes of the very low flow variable deviation are conjugated with the extra failure mode 'indication buf_acd_fl_1 does_not_alarm very_low'. The latter is required to explain why one flow measurement indicates a very low deviation and the other does not. The extra cutset element is ANDed to the 'a priori' information when the low flow alarm is diagnosed, as discussed in Chapter 5.
Associated with explanations 2, 6, 7, 10, 11, 12, 13, and 14 is the 'pressure pl is_not_in_state high' boundary condition. This information must be specified by the system modeller and is required by the KBS to identify which faults are incompatible with a high pressure variable deviation.

The 22 explanations for the first two alarms are listed in order of likelihood below:

1. indication pump_pl_pres spuriously_alarms high @ 1 AND (58.4%)
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1

2. pipeline_section fails large_blockage @ 1 AND (28.9%)
   indication buf_acd_fl_1 does_not_alarm very_low @ 1

3. device pump_pl fails stopped @ 1 AND (4.84%)
   indication pump_pl_pres spuriously_alarms high @ 1 AND
   indication buf_acd_fl_1 does_not_alarm very_low @ 1

4. control_loop c1_l fails very_low @ 1 AND (3.34%)
   indication buf_acd_fl_1 does_not_alarm very_low @ 1

5. device pipe_5 fails large_blockage @ 1 AND (2.67%)
   indication buf_acd_fl_1 does_not_alarm very_low @ 1

6. flow f2 => very_low @ 1 AND control_loop c1_l working @ 1 AND (1.17%)
   indication buf_acd_fl_1 does_not_alarm very_low @ 1

7. upstream_valves fails closed @ 1 AND (0.36%)
   indication buf_acd_fl_1 does_not_alarm very_low @ 1 AND
   indication pump_pl_pres spuriously_alarms high @ 1

8. level 11 => high @ 1 AND (0.2621%)
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1

9. indication pump_pl_pres spuriously_alarms high @ 1 AND (0.1%)
   indication buf_acd_fl_1 does_not_alarm very_low @ 1 AND
   device upstream_tracer fails off @ 1
10 indication pump_pl_pres spuriously_alarms high @ 1 AND (2.8x10^{-4})
  tank_section fails leakage @ 1 AND
  indication buf_acd_fl_l does_not_alarm very_low @ 1

11 indication pump_pl_pres spuriously_alarms high @ 1 AND (1.96x10^{-4})
  indication buf_acd_fl_l does_not_alarm very_low @ 1 AND
  pump_section fails leakage @ 1

12 indication pump_pl_pres spuriously_alarms high @ 1 AND (1.17x10^{-4})
  indication buf_acd_fl_l does_not_alarm very_low @ 1 AND
  tank_section fails large_blockage @ 1

13 indication pump_pl_pres spuriously_alarms high @ 1 AND (1.127x10^{-4})
  indication buf_acd_fl_l does_not_alarm very_low @ 1 AND
  pump_section fails large_blockage @ 1

14 pipeline_section fails large_blockage @ 1 AND (4.207x10^{-5})
  device trans_1 fails normal @ 1 AND
  device fs_1 fails very_low @ 1

15 indication pump_pl_pres spuriously_alarms high @ 1 AND (3.81x10^{-5})
  indication buf_acd_fl_l does_not_alarm very_low @ 1 AND
  pipeline_section fails leakage @ 1

16 level 11 => high @ 1 AND (7.81x10^{-6})
  device upstream_tracer fails off @ 1 AND
  indication buf_acd_fl_l does_not_alarm very_low @ 1

17 control_loop cl_1 fails very_low @ 1 AND (4.7x10^{-6})
  device trans_1 fails normal @ 1 AND
  device fs_1 fails very_low @ 1

18 device pipe_5 fails large_blockage @ 1 AND (3.8x10^{-6})
  device fs_1 fails very_low @ 1 AND
  device trans_1 fails normal @ 1
There are four points of interest which can be noted from these results:

1. The spurious failure of both alarms is still the most likely explanation for the two alarms. This is not usually the case when the causes of two alarms sharing common failure modes are conjugated. However, in this example the passive failure of indication buf_acd_fl_1 weights down the actual causes of the high pressure and very low flow deviations.

2. The results illustrate how grouping together faults which cause very similar symptoms can improve the clarity of the alarm diagnosis. The technique not only reduces the total number of explanations, but also means that the collection of faults has a greater likelihood than any of the individual failures, thereby more strongly directing the operators attention to the group of indistinguishable faults. There are, however, disadvantages with the method, which are discussed in Chapter 9.

3. As can be seen from explanations 14, 17, 18, 19 and 22, the 'indication buf_acd_fl_2 spuriously_alarms very_low @ 1' fault has been replaced by the two failure modes 'device trans_1 fails normal @ 1 AND device fs_1 fails very_low @ 1'. The general method
used to calculate the spurious failure frequency of an alarm is
described in Chapter 5. However, in this example where flow fl is
indicated by the instrumentation shown in Figure 8.2 and the two
flow indications disagree, the following cutsets describe the two
means by which indication buf_acd_fl_2 can spuriously alarm
very_low:

1  device trans_2 fails very_low @ 1

2  device trans_1 fails normal @ 1 AND
device fs_1 fails very_low @ 1

The first cutset implies that flow fl is in fact normal and
devices fs_1 and trans_1 are working; the second implies that the
flow is in an unknown state and device trans_2 is working.

Figure 8.2  Flow Fl Indication Instrumentation

Unfortunately, when the causes of the high pressure
deviation, which also cause the very low flow deviation, are
conjugated with the spurious failure of indication buf_acd_fl_2,
the resulting cutsets will be incorrect. The logical inconsistency
arises because the most probable cause of the spurious alarm
failure, 'device trans_2 fails very_low', implies that the buffer tank inlet flow is normal, which is inconsistent with the side effects of the high pressure alarm causes.

The logical inconsistency can only be detected if the following causes of the high pressure alarm also have the boundary condition 'flow f1 is_in_state very_low' associated with them in the 'a priori' information:

1. device pipe_5 fails large_blockage
2. pipeline_section fails large_blockage
3. control_loop cl_l fails very_low
4. signal set_pt => very_low AND control_loop cl_l working
5. flow f2 => very_low AND control_loop cl_l working

When this is the case, the 'device trans_2 fails very_low' cause of the spurious very low flow alarm will be rejected, leaving the second explanation only.

The last two explanations, which both include the fault 'signal set_pt => low', have zero likelihoods. This is again due to the fact that the set point signal was observed to be not low when the alarms occurred, and the rulebase does not describe a means by which the setpoint could be really low when it was indicated otherwise.

8.1.2.2 Diagnosing The Low Buffer Tank Level Alarm

Sixty seconds after the first two alarms, the low buffer tank level alarm will activate. Since a very low inlet flowrate is a direct cause of the third alarm, its causes will be conjugated with the ten most likely causes of the first two. The results of this combination are listed overleaf:
1. pipeline_section fails large blockage @1 AND (71.7%)
   indication buf_acd_fl_l does_not_alarm very_low @1

2. device pump_pl fails stopped @1 AND (12.06%)
   indication pump_pl_pres spuriously_alarms high @1 AND
   indication buf_acd_fl_l does_not_alarm very_low @1

3. control_loop cl_1 fails very_low @1 AND (8.33%)
   indication buf_acd_fl_l does_not_alarm very_low @1

4. device pipe_5 fails large_blockage AND (6.66%)
   indication buf_acd_fl_l does_not_alarm very_low @1

5. upstream_valves fail closed @1 AND (0.89%)
   indication pump_pl_pres spuriously_alarms high @1 AND
   indication buf_acd_fl_l does_not_alarm very_low @1

6. device upstream_tracer fails off @1 AND (0.254%)
   indication pump_pl_pres spuriously_alarms high @1 AND
   indication buf_acd_fl_l does_not_alarm very_low @1

7. indication pump_pl_pres spuriously_alarms high @1 AND (2.7x10^-2)
   indication buf_acd_fl_2 spuriously_alarms very_low @1 AND
   device valve_7 fails opened @

8. indication pump_pl_pres spuriously_alarms high @1 AND (6.2x10^-4)
   indication buf_acd_fl_2 spuriously_alarms very_low @1 AND
   device tank_2 fails leakage @6

9. indication pump_pl_pres spuriously_alarms high @1 AND (3.0x10^-4)
   indication buf_acd_fl_l does_not_alarm very_low @1 AND
   tank_section fails leakage @1

10. level 11 => high @1 AND (1.73x10^-4)
    indication buf_acd_fl_2 spuriously_alarms very_low @1 AND
    device valve_7 fails opened @6
11 indication pump_pl_pres spuriously_alarms high @ 1 AND \(6.4 \times 10^{-5}\) 
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND 
   indication buf_tnk_lev spuriously_alarms low @ 6

12 indication pump_pl_pres spuriously_alarms high @ 1 AND \(4.2 \times 10^{-6}\) 
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND 
   device valve_6 fails leakage @ 6

13 level 11 \(\Rightarrow\) high @ 1 AND \(4.01 \times 10^{-6}\) 
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND 
   device tank_2 fails leakage @ 6

14 flow f2 \(\Rightarrow\) very_low @ 1 AND \(1.85 \times 10^{-6}\) 
   control_loop cl_1 working @ 1 AND 
   indication buf_tnk_lev spuriously_alarms low @ 6

15 level 11 \(\Rightarrow\) high @ 1 AND \(4.2 \times 10^{-7}\) 
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND 
   indication buf_tnk_lev spuriously_alarms low @ 6

16 indication pump_pl_pres spuriously_alarms high @ 1 AND \(4.2 \times 10^{-8}\) 
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND 
   device pipe_5 fails leakage @ 6

17 indication pump_pl_pres spuriously_alarms high @ 1 AND \(4.2 \times 10^{-8}\) 
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND 
   device pipe_6 fails leakage @ 6

18 indication pump_pl_pres spuriously_alarms high @ 1 AND \(3.7 \times 10^{-8}\) 
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND 
   flow f2 \(\Rightarrow\) high @ 6

19 level 11 \(\Rightarrow\) high @ 1 AND \(2.6 \times 10^{-8}\) 
   indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND 
   device valve_6 fails leakage @ 6

240
20 indication pump_pl_pres spuriously_alarms high @ 1 AND (2.7x10^{-9})
indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND
device valve_6 fails opened @ 6 AND
control_valve cl_1 fails normal @ 6

21 indication pump_pl_pres spuriously_alarms high @ 1 AND (7.1x10^{-10})
indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND
flow f1 -> low @ 6

22 level 11 -> high @ 1 AND (2.67x10^{-10})
indication buf_acd_fl_2 spuriously_alarms very_low @ 1
device pipe_6 fails leakage @ 6

23 level 11 -> high @ 1 AND (2.66x10^{-10})
indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND
device pipe_5 fails leakage @ 6

24 level 11 -> high @ 1 AND (2.5x10^{-10})
indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND
device valve_6 fails opened @ 6 AND
control_loop cl_1 fails normal @ 6

25 level 11 -> high @ 1 AND (2.43x10^{-10})
indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND
flow f2 -> high @ 6

26 level 11 -> high @ 1 AND (6.99x10^{-12})
indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND
flow f1 -> low @ 6

27 indication pump_pl_pres spuriously_alarms high @ 1 AND (1x10^{-15})
indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND
pressure p2 -> low @ 6 AND
control_loop cl_1 fails normal @ 1
level 11 => high @ 1 AND (9.7x10^-17 %)
indication buf_acd_fl_2 spuriously_alarms very_low @ 1 AND
pressure p2 => low @ 6 AND
control_loop cl_l fails normal @ 6

The "real" causes of the first three alarms are in fact represented by the third cutset. As discussed in Section 8.1.2, the controller and level sensor outputs are not consistent with the failure modes required to cause the very low flow and high pressure alarms. Consequently, the 'control_loop cl_l fails very_low' failure effectively only represents the very low failure of the control valve. The fault will exhibit exactly the same symptoms as blockages in pipes 3,4 and in the control valve (represented by the fault pipeline_section fails large blockage). Therefore, the KBS can only differentiate between the two on the basis of their 'a priori' frequencies of failure. For the purposes of this study, the group of blockage faults are assumed to have a higher frequency of failure than the control valve failing very low. Consequently, the alarm cause rankings reflect this.

For the sake of brevity the diagnoses of the remaining three alarms:

4 Indication reac_acd_fl alarms low during scan 7
5 indication buf_tnk_lev alarms very_low during scan 20
6 indication reac_acd_fl alarms very-low during scan 37

are not considered in this example. Since all three are either direct or indirect consequences of a flow restriction between the pressure sensor and the buffer tank inlet, the most significant explanation for the first three alarms will also explain the last three alarms with the greatest likelihood.
8.2 The Batch Distillation Pilot Plant

The types of fault which could be simulated on the batch distillation plant were obviously restricted by safety constraints. Despite this, a number of interesting fault scenarios were investigated, the two best examples of which are described in Sections 8.2.1 and 8.2.2.

The schematic diagram of the batch distillation unit is shown in Figure 8.3.

8.2.1 Example 3: Loss Of Power To The Reboiler Heater

A loss of power to the reboiler heater was simulated by switching off the mains electricity supply to the reboiler power controller. The consequences of this action are listed below:

1 indication ti3 alarms low during scan 23
2 indication pi5 alarms low during scan 32
3 indication ti1 alarms low during scan 40
4 indication ti4 alarms low during scan 40
5 indication ti8 alarms low during scan 45
6 indication ti14 alarms low during scan 49
7 indication ti13 alarms low during scan 57
8 indication ti12 alarms low during scan 143
Figure 8.3 The Batch Distillation Process
It can be seen from the above list that both indications ti4 and til alarm during the same scan. These two indications independently monitor the reboiler liquid temperature, however, the temperature histories shown in Figure 8.4 illustrate that there is some systematic error associated with these measurements. As a result, the two indications will cross the low temperature alarm threshold during different scan intervals.

**Figure 8.4 The Reboiler Liquid Temperature Histories**

Despite the systematic error, the two measurements are assumed to be within the acceptable tolerance limits for the temperature sensors. Therefore to avoid the problem of the two indications alarming during different scan intervals, the alarm detection program averages the two readings and checks the result against the alarm limit, as described in Chapter 6. If the two measurements had been significantly different, that is the discrepancy could not have been explained in terms of an acceptable level of instrument error, then the signals would have been treated independently.
8.2.1.1 Diagnosing The Low Reboiler Vapour Temperature Alarm

When the low reboiler vapour temperature alarm is diagnosed (indication ti3) the following nine explanations will be generated:

1. control_loop_bupr_1 fails very-low @ 23 (52.5%)
2. signal_bupr_set_pt fails very_low @ 23 (21.9%)
3. pressure_p5 => low @ 23 (14.76%)
4. indication_ti3 spuriously_alarms low @ 23 (5.2%)
5. device_mains_power_switch fails switched_off @ 23 (3.4%)
6. device_power_supply fails off @ 23 (1.9%)
7. device_reboiler_hl fails air_leak @ 23 (0.15%)
8. pressure_p5 => high @ 23 (4x10^{-12} %)
9. device_condenser_cl fails water_leak @ 23 (4.9x10^{-14} %)

The boilup rate control loop failure dominates the alarm explanations with a likelihood of 52.5%. This high likelihood is due to two main factors:

1. From reference [38] it can be seen that process control computers generally have relatively high failure frequencies.

2. The outputs from the reflux flow transducer and the controller (the process control computer) are not available to the data logging computer. As a consequence, it is not possible to match the outputs from these two devices against their suspected failure modes, and reject those faults which do not agree. A failure of any of the control loop components is therefore assumed to have caused the fault, which in turn evaluates to a higher failure
frequency. In this example, if the boilup rate controller output had been observable, the associated alarm explanation would have been ranked third as opposed to first.

Unusually both high and low pressure deviations are listed as causing the low vapour temperature alarm. This is because the former will temporarily cause a cessation in vapour flow until the reboiler reaches the boiling temperature, and the latter will reduce the equilibrium boiling temperature. The low pressure deviation is, however, much more significant since the pressure is discretized as 20% low when the first alarm occurs.

Finally, an air leak in the reboiler is listed as a cause of the low temperature alarm, since the ambient air temperature is considerably lower than the normal vapour temperature. It is assumed that air leakages in the remainder of the system will have less of an impact on the vapour temperature because they will first have to propagate through the column.

8.2.1.2 Diagnosing The Low Column Pressure And Reboiler Temperature Alarms

The faults which cause the reboiler heater to stop functioning are common to all of the first four alarms, therefore they will be diagnosed in conjunction with each other. The results of this diagnosis are listed below:

1 control_loop bupr_1 fails very_low @ 23 (58%)
2 signal bupr_set-pt -> very-low @ 23 (24.3%)
3 control_loop pic5 fails low @ 23 (11.7%)
4 device mains_power_switch fails switched_off @ 23 (3.73%)
5 device power_supply fails off @ 23 (2.13%)
device valve v5/3 fails passing @ 23 (5.6x10^{-2})

device condenser_cl fails water_leak @ 23 (4.1x10^{-14})

indication ti3 spuriously_alarms low @ 23 and (8x10^{-17})
indication pi5 spuriously_alarms low @ 32 and
indication til spuriously_alarms low @ 40 and
indication ti4 spuriously_alarms low @ 40

device reboiler_h1 fails air_leak @ 23 and (7.2x10^{-17})
indication pi5 spuriously_alarms low @ 32 and
indication til spuriously_alarms low @ 40 and
indication ti4 spuriously_alarms low @ 40

pressure p5 => high @ 23 and (4x10^{-20})
indication pi5 spuriously_alarms low @ 32 and
indication til spuriously_alarms low @ 40 and
indication ti4 spuriously_alarms low @ 40

signal pic5_set_pt => low @ 23 0%

The faults associated with the reboiler boil-up rate controller are still ranked most highly, for the reasons discussed previously. However, the low column pressure deviation fault has been replaced by its various causes, some of which are common to the other two alarms.

The condenser water leakage fault has an extremely low likelihood for a single order cutset because it has three extra boundary conditions associated with it. These state that following such a leakage the column wall temperatures t12, t13 and t14 would be expected to become very low. Since their indications are not consistent with this state when the first three alarms occur, the likelihood of the condenser fault is decreased.

Finally, in sharp contrast to the boilup rate controller's set point deviation, the pressure controller’s set point deviation has a zero likelihood. This is because the KBS can cross check the state of the latter during the diagnosis and reject the explanation.
8.2.1.3 Diagnosing The Low Condenser Vapour Temperature Alarm

For the purposes of modelling the batch distillation column, the reboiler composition was assumed to remain constant. As a result the causes of the first three alarms do not include fault scenarios which will change the column composition.

The condenser composition, however, is not constrained by such an assumption and so composition changes are considered as causes of condenser vapour temperature deviations. Unfortunately, the vapour composition is not measured, so the causes of composition deviations are directly related to vapour temperature changes. For example, a high vapour flowrate from the reboiler will increase the reflux ratio, and hence increase the mole fraction of the more volatile component at the condenser. As a consequence the condenser vapour temperature will decrease. For this reason the 'control_loop_bupr_l fails high' and 'signal_bupr_set_pt => high' failure modes are included as causes of indication ti8 alarming low, as shown below:

1 pressure p5 => low
2 pressure p5 => high
3 control_loop_bupr_l fails high
4 signal_bupr_set_pt => high
5 temperature t3 => low
6 column_section fails air_leak
7 device_ancillary_pipework fails air_leak
8 device_bursting_disc fails ruptured
9 ancillary_valves fails opened
The low condenser temperature alarm is a potential consequence of the low reboiler vapour temperature alarm, and therefore it will be diagnosed in conjunction with the four previous alarms. The results of this new diagnosis are listed below:

1. control_loop bupr_l fails very_low @ 23 (58%)
2. signal bupr_set-pt => very-low @ 23 (24.3%)
3. control_loop pic5 fails low @ 23 (11.7%)
4. device mains_power_switch fails switched_off @ 23 (3.73%)
5. device power_supply fails off @ 23 (2.13%)
6. device valve v5/3 fails passing @ 23 (5.6x10^{-2})
7. device condenser_cl fails water_leak @ 23 (4.1x10^{-14})
8. device reboiler_hl fails air_leak @ 23 and (7.2x10^{-17})
   indication pi5 spuriously_alarms low @ 32 and
   indication ti1 spuriously_alarms low @ 40 and
   indication ti4 spuriously_alarms low @ 40
9. indication ti3 spuriously_alarms low @ 23 and (5.3x10^{-20})
    indication pi5 spuriously_alarms low @ 32 and
    indication ti1 spuriously_alarms low @ 40 and
    indication ti4 spuriously_alarms low @ 40 and
    ancillary_valves fail opened @ 45
10 indication ti3 spuriously_alarms low @ 23 and (2.5x10^{-20}%) 
indication pi5 spuriously_alarms low @ 32 and 
indication ti1 spuriously_alarms low @ 40 and 
indication ti4 spuriously_alarms low @ 40 and 
control_loop bupr_1 fails high @ 45

11 indication ti3 spuriously_alarms low @ 23 and (1.1x10^{-20}%) 
indication pi5 spuriously_alarms low @ 32 and 
indication ti1 spuriously_alarms low @ 40 and 
indication ti4 spuriously_alarms low @ 40 and 
collection_vessel_section fails air_leak @ 45

12 indication ti3 spuriously_alarms low @ 23 and (1.0x10^{-20}%) 
indication pi5 spuriously_alarms low @ 32 and 
indication ti1 spuriously_alarms low @ 40 and 
indication ti4 spuriously_alarms low @ 40 and 
device_valve 4/3 fails opened @ 45

13 indication ti3 spuriously_alarms low @ 23 and (1.0x10^{-20}%) 
indication pi5 spuriously_alarms low @ 32 and 
indication ti1 spuriously_alarms low @ 40 and 
indication ti4 spuriously_alarms low @ 40 and 
signal bupr_set_pt => high @ 45

14 indication pi5 spuriously_alarms low @ 32 and (2.9x10^{-20}%) 
indication ti1 spuriously_alarms low @ 40 and 
indication ti4 spuriously_alarms low @ 40 and 
pressure p5 => high @ 23

15 indication ti3 spuriously_alarms low @ 23 and (2.5x10^{-21}%) 
indication pi5 spuriously_alarms low @ 32 and 
indication ti1 spuriously_alarms low @ 40 and 
indication ti4 spuriously_alarms low @ 40 and 
indication ti8 spuriously_alarms low @ 45
16 indication ti3 spuriously_alarms low @ 23 and (1.86x10⁻²¹)%
   indication pi5 spuriously_alarms low @ 32 and
   indication ti1 spuriously_alarms low @ 40 and
   indication ti4 spuriously_alarms low @ 40 and
   device bursting_disc fails ruptured @ 45

17 indication ti3 spuriously_alarms low @ 23 and (7.36x10⁻²³)%
   indication pi5 spuriously_alarms low @ 32 and
   indication ti1 spuriously_alarms low @ 40 and
   indication ti4 spuriously_alarms low @ 40 and
   device column_el fails air leak @ 45

18 indication ti3 spuriously_alarms low @ 23 and (7.36x10⁻²³)%
   indication pi5 spuriously_alarms low @ 32 and
   indication ti1 spuriously_alarms low @ 40 and
   indication ti4 spuriously_alarms low @ 40 and
   device condenser_cl fails air leak @ 45

19 indication ti3 spuriously_alarms low @ 23 and (5.6x10⁻²³)%
   indication pi5 spuriously_alarms low @ 32 and
   indication ti1 spuriously_alarms low @ 40 and
   indication ti4 spuriously_alarms low @ 40 and
   device ancillary_pipework fails air leak @ 45

Despite the fact that the causes of the low condenser temperature alarm are much more diverse than those of the previous alarms, the conjugation of the new alarm does little to effect the seven most likely combined alarm explanations. This is principally because these seven faults will cause both a low reboiler vapour temperature and a low column pressure, either of which will lead to a low condenser temperature. The diversity of faults does, however, increase the number of unlikely faults scenarios. Fortunately, because of their insignificant failure frequencies they do not have a large impact on the likelihoods of the more probable explanations.

The number of alarm explanations is further increased because the grouped failure mode 'column_section fails air_leak' in conjunction with the spurious alarming of indications ti1, ti3, ti4 and pi5 is
expanded into explanations 12, 17 and 18. This is necessary because the 'reboiler_hl fails air_leak' fault ANDed with the spurious alarm faults is rendered non-minimum by cutset number 8.

8.2.1.4 Diagnosing The Low Column Wall Temperature Alarms

The last three alarms, indicating low column wall temperatures, lag behind the low condenser temperature alarm because of the thermal inertia of the column. Despite the dynamic differences, all four alarms are caused by exactly the same set of faults except for the spurious failures of the various indications. The last three alarms are therefore diagnosed together with the earlier alarms.

As discussed in previous sections, only the ten most likely explanations from a combined alarm diagnosis are automatically conjugated with the causes of a new alarm. The remainder are only carried forward if they share common failure modes with the new alarm. In this case explanations 11, 12, 13, 16, 17, 18 and 19 share common failure modes with the last three alarms, and are therefore combined with their causes.

The similarity in alarm explanations results in a diagnosis for all eight alarms which is virtually identical to the list in the previous section. The notable difference is that cutsets 14 and 15 are omitted. As can be seen from this example, the most likely alarm explanations change very little once the low reboiler vapour temperature and column pressure alarms have been diagnosed. This is essentially because the other alarms are either direct or indirect consequences of the same faults which caused these first two alarms. Despite the lack of diversity in the example, the results show how a relatively large number of alarms can simply be related back to a few key fault scenarios.
8.2.2 Example 4: Nitrogen Leak Into The Column And Pressure Control Loop PIC5 Fails Inactive

The effects of a high column pressure deviation (from 608 mmHg to atmospheric pressure) were simulated by opening the nitrogen purge isolation valve V11/1 and adjusting the pressure reduction valve PCV1 to allow a moderate flow of nitrogen into the column. The pressure control loop was rendered inactive by moving the set point to atmospheric pressure. Throughout the duration of the experiment the following alarms were observed to occur:

1. indication pi5 alarms high during scan 4
2. indication ti1 alarms high during scan 10
3. indication ti4 alarms high during scan 10
4. indication ti3 alarms low during scan 14
5. indication ti8 alarms low during scan 20
6. indication ti14 alarms low during scan 35
7. indication ti12 alarms low during scan 42
8. indication ti14 alarms high during scan 76
9. indication ti13 alarms high during scan 89
10. indication ti3 alarms high during scan 99

It is interesting to note that after the high pressure deviation, all the column temperatures decrease, with the exception of the reboiler liquid. However, as the reboiler liquid attains its new equilibrium boiling temperature and starts propagating vapour back up the column, the same temperatures deviate high. Although not observed, it is expected that indications ti8 and ti12 would have alarmed high if the experiment had been allowed to continue.
8.2.2.1 Diagnosing The high Column Pressure Alarm

The twelve ranked explanations for the high pressure alarm are listed below:

1. ancillary_valves fails opened @ 4 (64.4%)
2. column_section fails air_leak @ 4 (16.4%)
3. collection_vessel_section fails air_leak @ 4 (16.2%)
4. device bursting_disc fails ruptured @ 4 (2.8%)
5. indication pi5 spuriously_alarms high @ 4 (0.131%)
6. device ancillary_pipework fails air leak @ 4 (8.65x10^-2%)
7. condenser_cl fails high temperature @ 4 and (1.48x10^-2%)
   control_loop pic5 fails normal @ 4
8. device N2_purge_supply fails opened @ 4 and (6.18x10^-3%)
   control_loop pic5 fails normal @ 4
9. pressure p4 => high @ 4 (3.9x10^-3%)
10. ancillary_valves fail passing @ 4 and (7x10^-5%)
   control_loop pic5 fails normal @ 4
11. column_valves fail passing @ 4 and (4.2x10^-5%)
   control_loop pic5 fails normal @ 4
12. collection_vessel_valves fail passing @ 4 and (2.83x10^-5%)
   control_loop pic5 fails normal @ 4

Where the group 'ancillary_valves' represents valves 3/3, 5/1, 4/10 and 4/7; 'column_section' represents devices reboiler_h1, column_el
and valve 4/2; 'collection_vessel_section' represents devices vessel_r1 and valve 3/2; 'column_valves' represents valves 4/3, 6/1 and 6/2; 'collection_vessel_valves' represents valves 3/1 and 8/2.

Since there are so many potential causes of air leakages into the the reduced pressure system, extensive use has been made of the fault grouping facility. In this case, if all the grouped failure modes were expanded, the list of high pressure alarm causes would contain 103 elements. Most of these air leak causes have very low failure frequencies, therefore the accidental opening of the various vent valves, represented by the group failure mode 'ancillary_valves fails opened' is ranked as the most likely explanations for the alarm.

### 8.2.2.2 Diagnosing The High Reboiler Temperature Alarms

A high reboiler liquid temperature deviation has two possible causes, namely a decrease in the mole fraction of the more volatile component or an increase in the system pressure. Reboiler composition deviations have not been considered in this study, therefore only a high pressure deviation is considered as a viable cause of a high temperature deviation. A localised build-up of pressure in the reboiler due to the boilup rate controller failing 100% high, and the column offering a significant resistance to vapour flow, was considered. However, this failure mode was not simulated experimentally. Therefore because of the uncertainty in its applicability, it was not included as a cause of the high temperature alarm.

The two causes of a high reboiler liquid temperature alarm which are considered are listed below:

1. pressure p5 => high @ 10

2. indication ti1 spuriously_alarms high @ 10 and indication ti4 spuriously_alarms high @ 10
When these two cutsets are conjugated with the ten most likely explanations for the first alarm, the resulting diagnosis is essentially the same as that for the first alarm. The main difference is that the fifth cutset is now conjugated with the spurious alarming of indications til and ti4. As a result the fault scenario is ranked tenth and so the original explanations numbered 6 to 10 are ranked one place higher.

8.2.2.3 Diagnosing The Low Temperature Alarms Of Indications T13, T18, T112 And T114

Shortly after the high temperature alarms, indications ti3, ti8, ti12 and ti14 all alarm low. Indication ti13 is conspicuous by its absence, however, the variable does deviate low, but not sufficiently low enough to cross the alarm threshold. As can be seen from the previous example, the individual causes of these alarms are very similar. Furthermore, they are all possible consequences of a high column pressure deviation, therefore when their causes are conjugated with the first three alarms, the following explanations will be generated:

1 ancillary_valves fails opened @ 4 (64.4%)
2 column_section fails air_leak @ 4 (16.4%)
3 collection_vessel_section fails air_leak @ 4 (16.2%)
4 device bursting_disc fails ruptured @ 4 (2.8%)
5 device ancillary_pipework fails air_leak @ 4 (0.09%)
6 condenser_cl fails high temperature @ 4 and (0.01%)
   control_loop pic5 fails normal @ 4
7 device N2_supply fails opened @ 4 and (6.1x10^-3%)
   control_loop pic5 fails normal @ 4
8 pressure p4 -> high @ 4 (3.9x10^{-3}%) 

9 ancillary_valves fail passing @ 4 and (7.06x10^{-5}%) 
control_loop pic5 fails normal @ 4 

10 indication ti4 spuriously_alarms high @ 10 and (1.33x10^{-10}%) 
indication ti1 spuriously_alarms high @ 10 and 
indication pi5 spuriously_alarms high @ 4 
control_loop bupr_1 fails very_low @ 14 

8.2.2.4 Diagnosing The High Column Wall Temperature Alarms

Rather unexpectedly two of the column wall temperature alarm high before the reboiler vapour temperature. It is thought that this is due due to the column wall heaters slowing down the rate of cooling of the walls. When the vapour reappears from the reboiler, it therefore takes less time to re-heat the walls up to the new equilibrium temperature.

Originally only the following six causes of indications ti14 alarming high were considered in the 'a priori' diagnostic information:

1 temperature ti3 -> high

2 control_loop bupr_1 fails low

3 signal bupr_set_pt -> low

4 device valve 1/1 fails open and device valve 1/2 fails open

5 pressure p5 -> low

6 indication ti14 spuriously_alarms high

Explanations 2, 3 and 4 were included to represent the causes of a loss of reflux, and explanation 5 was included following the analysis of
the fault simulation results. After modelling the system with the FAULTFINDER program, it was assumed that the high column wall temperature alarms would follow the high reboiler vapour temperature alarm. Because of this assumption the former alarms were related back to the high deviation in reboiler vapour temperature deviation (the nearest alarmed variable deviation).

Clearly when the link in the causal chain (namely the high reboiler vapour temperature deviation) does not occur in the expected sequence, the strategy of relating each alarm to the nearest alarmed variable deviation is not sufficient. Dynamic effects also need to be considered.

In this case it was therefore necessary to include the high column pressure deviation as a direct cause of the two high column wall temperature deviations. When the alarms are diagnosed the following alarm diagnoses result:

1  ancillary_valves fails opened @ 4 (64.4%)
2  column_section fails air_leak @ 4 (16.4%)
3  collection_vessel_section fails air_leak @ 4 (16.2%)
4  device bursting_disc fails ruptured @ 4 (2.8%)
5  device ancillary_pipework fails air_leak @ 4 (0.09%)
6  condenser_c1 fails high temperature @ 4 and (0.01%)
   control_loop pic5 fails normal @ 4
7  device N2_supply fails opened @ 4 and (6.1x10^{-3}%)
   control_loop pic5 fails normal @ 4
8  pressure p4 => high @ 4 (3.9x10^{-3}%)
9  ancillary_valves fail passing @ 4 and (7.06x10^{-5}%)
   control_loop pic5 fails normal @ 4
indication ti4 spuriously_alarms high @ 10 and (1.37x10⁻¹¹)
indication ti1 spuriously_alarms high @ 10 and
indication pi5 spuriously_alarms high @ 4 and
control_loop bupr_1 fails very_low @ 14 and
indication ti14 spuriously_alarms high @ 76 and
indication ti13 spuriously_alarms high @ 89

If the pressure deviation hadn't been included, the diagnosis would have been quite significantly different until the high reboiler vapour temperature alarm had been diagnosed.

Finally, the high reboiler vapour temperature alarm only has two causes, a high reboiler liquid temperature deviation and a spurious failure of the sensing equipment. Given that the first nine explanations in the previous diagnosis are direct causes of a low reboiler liquid temperature, the final diagnosis contains all nine cutsets. As might be expected the only difference is tenth cutset which now contains the extra element 'indication ti3 spuriously_alarms high @ 99)

This chapter has demonstrated the application of the fault diagnosis methodology, described in this thesis, to four examples of process faults. Whilst the examples were selected to illustrate the broadest cross section of system features, clearly all aspects of the methodology have not been demonstrated. This is both because of the nature of the modelled systems and space constraints. However, the following chapter addresses the outstanding issues of significance.
Chapter 9

DISCUSSION AND CONCLUSIONS

The earlier chapters of this thesis discuss the development of an alarm diagnosis methodology which uses a fault tree representation of process plant behaviour. This is then followed by a description of the implementation of the methodology within an experimental KBS. Finally, the later chapters discuss how the KBS was used to diagnose alarms on two examples of process plant.

The first half of this chapter includes a discussion of the major points arising from the fault diagnosis methodology, based on the experience gained through its application. The remainder of the chapter is concerned with some of the wider issues relating to the task of fault diagnosis.

9.1 The Diagnosis Methodology

9.1.1 The Single Fault Assumption

The single fault assumption is often used within fault diagnosis methodologies to simplify the problem. For example, Kramer and Palowitch [28] state that within their digraph-based method it is assumed that a single fault, affecting a single node in the signed digraph (the root node), is the source of all disturbances. Other methods are not theoretically constrained to relating a pattern of symptoms to a single fault, but the practical difficulties of
considering more faults may outweigh the potential benefits. For example, the experimental fault diagnosis system based on the IQA technique, reported by Herbert and Williams [31], took five minutes to relate a pattern of symptoms to a single fault, but took approximately one hour if two faults were suspected.

9.1.1.1 Considering Multiple Fault Alarm Causes

The alarm diagnosis methodology described in this thesis has been developed so that it does not involve the single fault assumption. As discussed in Chapters 4 and 5, any number of conjugated events can be determined as a cause of a pattern of symptoms. It has been demonstrated by the worked examples in Chapter 8, that when there are no conflicting states of the same indication, a pattern of potentially related symptoms can usually be traced to at least one single order cutset. The fact that the symptoms are potentially related does not, however, cause the KBS to reject all the higher order cutsets which imply that the alarms are not related.

If the single fault assumption was incorporated into the KBS, the diagnostic task would be greatly simplified and hence the speed of execution would increase. This is because it would no longer be necessary to check the higher order cutsets for internal logical consistency. Furthermore, because all the cutsets would be first order, there would be no need to check for non-minimum cutsets. The key issue is therefore whether the extra effort of not imposing the single fault assumption justifies the improved quality of the resulting diagnoses.

For running systems where the process equipment failures will be revealed quickly, the single fault assumption can be justified because the likelihood of two or more process faults coexisting will be small. However, this is not the case when a system includes redundant equipment, e.g. standby pumps, generators, reactor coolant circuits, or trip systems and emergency pressure relief equipment. This is because the standby equipment can fail, and remain undetected in a failed
state, for some time. The probability of the equipment failing to work on demand may therefore be relatively high. For example, consider the parallel pumping system shown in Figure 9.1.

**Figure 9.1 The Parallel Pumping System**

Under normal operation only one pump is required to transfer process fluid. One pump is therefore commissioned, whilst the other acts as a standby. Once a month the operators change over the running and standby pumps. This evens the wear and tear and ensures that both pumps are run every other month.

A typical failure rate for a centrifugal pump on standby is in the order of 40 faults/10^6 hours. If the standby pump is left untested for 28 days, it will on average fail after 14 days of standby operation. The probability of the standby pump being in failed state is therefore the failure rate multiplied by the time in which it can fail, i.e. 14 days. This equates to a probability of 0.013, as shown overleaf:
If the running pump fails, the system should normally be able to recover by automatically starting the standby pump. The frequency of failure of the whole system is therefore the failure frequency of the running pump multiplied by the probability of the standby pump already being in a failed state.

If the failure of both pumps causes a low discharge flow alarm, an alarm diagnosis system should be able to consider the failure of both the running and the standby pump. When the failure of only the running pump is considered, the frequency of the fault scenario will be over-evaluated by a factor of 77 (the inverse of the probability of the standby pump failing).

9.1.1.2 Common Mode Failures

Another major limitation of the single fault assumption, especially in relation to the methodology described within this thesis, is that it could hinder the identification of common failure modes. These are particularly important because they generally have high failure frequencies. For example, let us reconsider the parallel pumping system shown in Figure 9.1. If the causes of a low discharge flow alarm are being diagnosed, the following two alternatives may be considered:

1. Both the running and standby pumps fail.

2. The suction pipe (pipe 1) fails blocked.

If the frequency of second order cutset is significantly less than the first order cutset, it could be argued that the single fault assumption was a valid approximation. However, before rejecting the second order cutset, an alarm diagnosis system should first prove that the two faults are truly independent. This is because if they are dependent, the frequency of the real cause of the two faults may be
higher than the single order cutset, and hence it would be incorrect to reject them. As an example, if the process power or turbine steam driving the pumps failed, both pumps would be inoperative. The frequency of such a utility failure could well be higher than the pipe blockage explanation.

Common failure modes are generally very difficult to detect and could potentially undermine the effectiveness of any fault diagnosis strategy. Simplifying the diagnosis problem by making the single fault assumption has many short term benefits. However, by employing the assumption, the information required to detect common failure modes (contained within the higher order cutsets) could be discarded. This is particularly true when explanations for multiple alarms are being considered. For example in Section 8.1.1 the diagnosis of both a high inlet flow alarm to the reactor charging system buffer tank and a low buffer tank level alarm includes the following cutset:

indication buf_tank_lev spuriously_alarms low @ 11 AND control_loop cl_1 fails high @ 2

This appears to be a unlikely second order cutset. However, on further investigation both these faults can be caused by the buffer tank level sensor failing low. This fault has a relatively high failure frequency and so the simplified explanation, derived from the second order cutset, is important.

9.1.1.3 Design Philosophy

The less likely, higher order cutsets may provide the operator with useful information. Andow and Galluzzo [58] argue that the operator can usually recognise the most likely (frequent) faults without the aid of a knowledge-based system. It is the infrequent faults that will present the greatest test of his diagnostic skills. It would therefore seem to be very poor (from a design philosophy viewpoint) to provide him with an aid that handles the easy faults and then expect him to be adequately equipped to deal with more complex and/or least likely faults by himself.
9.1.4 The Drawbacks Of The Multiple Fault Assumption

One of the potential drawbacks of allowing any number of faults to cause a set of related symptoms, is the possibility of interaction between faults. For example, consider the upstream section of the buffer tank system redrawn in Figure 9.2.

If a blockage occurs in pipe 3 the pump discharge pressure will increase and the downstream flow will decrease. When the two alarms occur, they will be diagnosed in conjunction with each other because of their potential relationship. The two alarms also have independent causes, for example, an increase in the upstream pressure will cause the pressure measured by sensor PS 1 to increase. Similarly, a leakage in pipe 3 will cause the flow indicated by TRANS 1 and TRANS 2 to decrease.

Unfortunately, the neither of the independent faults are consistent with the deviation caused by the other independent fault. A high upstream pressure could cause the flow through the flow sensor FS 1 to increase if there were no feedback control action, and a leakage in pipe 3 would definitely cause the pump discharge pressure to decrease. Because of this problem it is necessary to use boundary conditions to eliminate the inconsistent cutsets, but as will be discussed in Section 9.4 this information is difficult to synthesise using the current version of FAULTFINDER.

9.1.2 Reasoning With Time

As discussed in Chapter 7, the models used to describe the fault propagation behaviour of a process system within FAULTFINDER, are based on signed functional equations. Because all references to time are omitted and the steady state gains between deviations are only very crudely represented, the models are very generic. For example, it is possible to construct a general pipe model with standard failure modes, such as leakage and blockage, which can then be applied to virtually any system with little or no modification.
Figure 9.2 The Upstream Section Of The Buffer Tank System
The obvious benefit of being able to create generic models is that the time and effort required to construct the alarm tree information, and hence the diagnosis rules, is minimised. However, the very simplicity of the models means that they cannot be used to predict the final steady state value of a process variable. Similarly, it is impossible to calculate the time delays between events.

These limitations do not pose a severe problem because the task of alarm diagnosis is in effect the inverse of simulation. Rather than trying to predict what will happen given a certain process state, diagnosis involves analysing the symptoms after the event, in order to ascertain what has happened in the past. However, as discussed in Chapter 4, there is still a need to be able to decide if two events relate to the same time period, in order to detect logical inconsistencies.

Given the restrictions of the signed functional equation models and the reluctance to augment this with quantitative information, a very simple heuristic has been used to decide if two mutually exclusive events are supposed to coexist. Namely, that if the alarms caused by the two events are observed to coexist, so must the two events.

For all its simplicity the heuristic appears to work well. For example, reconsider the section of plant shown in Figure 9.2. If the upstream pressure deviates high and the control valve position remains unchanged, both the high pressure pl and the high flow fl alarms will activate. Since they are related they will be diagnosed in conjunction with each other, and will generate, amongst others, the following cutset:

1  Control_loop cl_1 fails high AND
   control_loop cl_1 fails low

The first cutset element causes a high flow and the second causes a high pressure.
Clearly this cutset is incorrect. However, because the time delay between the either of the events occurring and the alarms manifesting themselves would be small, the cutset can be rejected as mutually exclusive.

To take the reasoning one stage further, if the flow measurement did not exist, the same primary failure would cause a high pressure alarm, followed some time later by a high level alarm. Again they would be diagnosed together and the erroneous cutset generated. Providing that the high pressure alarm was still active when the low level alarm triggered, any faults which caused the first alarm would have to be consistent with those that caused the low level. The cutset could therefore be rejected.

With hard or non-rectifying faults the alarm coexistence heuristic therefore enables mutually exclusive process states to be identified. The problem arises when there is transient behaviour. For example, referring back to the previous case, if the upstream pressure decreased so that pressure $p_1$ was no longer high enough to trigger the alarm, the upstream flow $f_1$ might still be sufficient to cause the level to become high. In this situation the erroneous cutset would not be rejected.

Similarly, the heuristic may reject cutsets which are not strictly invalid if the process is oscillating. For example, consider the liquefied petroleum gas (LPG) distillation column shown in Figure 9.3.

The function of the column is to split the feed, containing light hydrocarbons ranging from ethane to heptane, into LPG (ethane, propane and butane) and the heavier components. The pressure is controlled by manipulating the overhead gas flow, and both the tower top temperature ($t_{il}$) and the pentane ($C_5$) mole fraction in the LPG stream ($a_{i2}$) are measured.
If the reboiler duty is oscillating with a period of 40 minutes, the overhead temperature and C5 composition will also oscillate with the same frequency, delayed by some time lag from the reboiler variations. Since the overhead temperature is related to the C5 composition at the top of the column, the two variables will oscillate in phase, (they will reach their maximum and minimum values at roughly the same time)

Unfortunately, the measurements of the two variables, and hence their high and low alarms, may not be in phase with each other. For example, the temperature measurement will track the real temperature very quickly, but the C5 analyser may take 20 minutes to sample and
perform its composition analysis. Because of this, a low C5 composition alarm may be coincident with a high tower top temperature alarm.

Clearly, the two alarms are caused by opposite deviations of the reboiler duty. However, both alarms could also be triggered by a high column pressure deviation, therefore they would be diagnosed together. The resulting explanations would include the following cutset, which would be rejected on the grounds of logical inconsistency:

1. Reboiler duty → high AND reboiler duty → low.

Whilst this cutset is invalid at the steady state, it does describe the cause of the two alarms in the dynamic situation.

Without more detailed knowledge of the dynamics involved in the propagation of process disturbances, it will always be difficult to reject alarm explanation cutsets with total confidence. When the process effectively changes from one steady state to another as a result of a process fault, the effects of process dynamics are less important. However, this is not the case when the process variables are highly interrelated, the system can exhibit oscillating behaviour (because of poor control), and there are significant time lags and dead times in the system.

9.1.3 The Level Of Fault Discrimination

As stated previously the FAULTFINDER suite of programs and its associated library of generic models enables process systems to be modelled quickly and new models to be prototyped with relative ease. However, whilst the level of detail represented within the standard models is suitable for generating design type fault trees, it may be more detailed than necessary for certain alarm diagnosis applications.

For example, reconsider the section of plant shown in Figure 9.2. Because there is only one pressure sensor between tank 1 and tank 2 and no other measurements from which the pressure can be inferred, it
is only possible to resolve a blockage fault as either upstream or downstream of the sensor. Despite this, because each pipe section and valve was specified in the system configuration to FAULTFINDER, the KBS rulebase includes indistinguishable faults such as pipe 3 blockage, pipe 4 blockage and pipe 5 blockage. The uncertainty is therefore whether these faults should be represented in their own right, or combined as one fault, similar to the approach adopted by Parmar [63].

The two main arguments for combining the faults are as follows:

1. The majority of the diagnosis execution time is consumed in combining potential alarm causes, checking them for logical consistency and minimality, and evaluating their failure frequency. Since the number of raw cutsets generated, and hence the diagnosis time, is proportional to the product of the number of individual alarm causes, there is a strong incentive to minimise their number.

2. The operator's display needs to be engineered so that the he/she can quickly understand the output from the diagnosis system. In order to achieve this, the operator must not be bombarded with a multitude of indistinguishable alarm explanations.

However, there are two main difficulties with grouping together alarm explanations:

1. A failure mode of a component may be indistinguishable from others, yet another failure mode of the same device may be easy to resolve. For example, a blockage in pipe 5 will have the same effect as a blockage in pipes 3 or 4, however, a leakage in pipe 5 can be detected if the level in tank 2 is falling and inlet flow is high. Since the leakage in pipe 5 can be resolved, it is more consistent if the blockage of pipe 5 is also listed as a separate fault.
If the diagnosis system returns with very concise alarm explanations, such as 'blockage between tank 1 and pressure sensor ps 1', the operator may forget to investigate all the possible sub-causes, i.e. valve 2 blocked, pipe 2 blocked, valve 3 blocked, pump 4 blocked and valve 4 blocked.

As discussed in Chapter 3, a compromise has been adopted within the KBS. When there are a number of similar causes of an alarm, the individual faults can be grouped together as a subsystem failure mode. This is then presented to the operator as a single fault. However, if clarification is required, the KBS can expand the grouped failure mode into its constituent faults.

Since the number of raw alarm explanations is reduced, the computational effort and hence the diagnosis time should theoretically decrease. Unfortunately, some of the benefit is lost due to the extra effort of checking that the elements of a grouped failure mode are logically consistent with other faults within the same cutset.

The logical extension of this approach would be to identify from the initial system configuration specified to FAULTFINDER, which device failure modes were best represented individually, and those that should be grouped together. The grouped devices could then be modelled as a single element within FAULTFINDER. This would simplify both the modelling task and the conversion of the resulting fault trees into a format suitable for the KBS.

9.1.4 The Use Of Failure Frequencies

Process alarms are generally "spontaneous" events because they will not be anticipated by the operator. Instead their function is to draw the operator's attention to a significant process disturbance, as soon as it is detected.

The alarm diagnosis methodology described in this thesis assumes that any alarm has a minimum of two alternative causes, one being a spurious sensor failure, the other a process fault which really causes...
the alarmed variable deviation. Because of this approach it is desirable to rank the alternative explanations, in order to direct the operators attention to the most probable cause first, and hence maximise the efficiency of his/her diagnostic skills.

As discussed in Chapter 5, the simplest solution to the ranking problem would be to estimate the probability of the various equipment failures. However, because an alarm indicates a change in plant status, it is not appropriate to use the probability of the equipment already being in a failed state. This is because if the equipment had already failed, the alarm would have occurred earlier. What is required is some measure of the probability of the equipment spontaneously failing. For example, if \( P_t \) is the probability of a device working at time \( t \), and \( P_t + dt \) is probability of it working \( dt \) seconds later, then if the alarm is detected at time \( t + dt \), the spurious failure probability is given by Equation 9.1:

\[
P_{dt} = P_t - P_t + dt
\]  

\( P_{dt} \) is in effect the difference in the probability of the device being in a failed state before and after the alarm. In the limit as \( dt \) tends to zero, \( P_{dt} \) tends to the first differential of the probability with time, or the failure density function as it is termed in reference [38].

The relationship between the failure density function and the in-service lifetime of electronic components and process equipment has been considered in some detail, as reported by Lees [38]. Unfortunately, the information is still very scarce and not widely applicable. As discussed in the same reference, in the absence of any detailed information, the assumption of constant hazard rate is therefore usually made. The hazard rate is defined overleaf by Equation 9.2:
\[
Z(t) = \frac{f(t)}{R(t)}
\]  

(9.2)

Where \( Z(t) \) is the hazard rate  
\( f(t) \) is the failure density function at time \( t \) (\( P_{dt} \))  
\( R(t) \) is the probability of equipment survival after time \( t \)

With a constant hazard rate the failure density function equates to:

\[
f(t) = Z \times \exp(-Z\times t)
\]  

(9.3)

Where \( Z \) is the constant hazard rate.

Within reference [38] it is also shown that the mean time to failure of a component (MTTF), as defined by Equation 9.4, equates to the reciprocal of the hazard rate, if it is assumed to be constant.

\[
\text{MTTF} = \int_{0}^{\infty} t \times f(t) \, dt = \frac{1}{Z}
\]  

(9.4)

Where MTTF is the mean time to failure of a component.

Substituting Equation 9.4 into 9.3 therefore yields Equation 9.5:

\[
f(t) = \frac{1}{\text{MTTF}} \times \exp(-t/m)
\]  

(9.5)

Finally, if it is assumed that the MTTF of a component is large, the failure density function, and hence the probability of a component spontaneously failing (\( P_{dt} \)), equates to the reciprocal of the mean life.

Since the MTTF of a component is a measurable quantity, and as discussed in Chapter 5, is reasonably well documented in terms of its inverse (the failure frequency) the latter is a good basis for ranking alarm causes, as implemented within the KBS. Furthermore, the
uncertainty in the published failure frequency data is generally more significant than the exponential term in Equation 9.5, which lends additional weight to the simplifying assumption.

Unfortunately, ranking alarm cause explanations based on their average failure frequency has two drawbacks:

1 Some alarm causes may not be spontaneous events. For example, if the operator increases the setpoint of a control loop above a high alarm limit, and the process variable exceeds the limit a little while later, the cause of the alarm is clearly the high setpoint. However, within the current implementation of the KBS, if the setpoint is accessible by the alarm diagnosis system, the high setpoint fault is treated as any other spontaneous event. The probability of the setpoint being high is multiplied by the average frequency of the setpoint being moved high.

Arguably, the operator will know when he/she has moved the setpoint above the alarm limit; some distributed control systems will not allow the operator to do this unless special confirmation is given. Estimating the average failure frequency for this non-spontaneous event is therefore difficult.

If the operator changes the setpoint intentionally, the setpoint will be below the alarm limit at one instant and above or equal to it at the next. The rate of change of the setpoint state with time will therefore be infinite, and as a consequence so should the "failure frequency". This would then force the likelihood of the alarm explanation to unity.

Alternatively, if the operator can move the setpoint above an alarm limit without knowing, the failure frequency needs to take account of this random effect. Selecting failure frequencies for operator driven changes therefore requires considerable care.

2 When using failure frequencies, the mathematics involved in evaluating the frequency of higher order cutsets becomes increasingly more complex. As discussed in Chapter 5; it is
meaningless to calculate the frequencies of these cutsets from the product of their element frequencies. For one reason the result has ridiculous units. Instead it is necessary to calculate the cutset frequency from the frequency of the oldest cutset element, multiplied by the probability of all the elements occurring together. This is not always a trivial task, especially when a number of alarms are detected during the same time scan.

In an attempt to simplify the mathematics, a method of converting the failure frequencies into probabilities, based on the alarm scan interval, was considered. The idea was founded on the argument that the diagnosis system can only resolve the activation time of each alarm to the scan interval during which it was detected. From Equation 9.1, the probability of a device spontaneously failing could then be re-defined in terms of Equation 9.6:

\[ P_{Ts} = P_t - P_{t+Ts} = f \times Ts \]  

Where
- \( P_{Ts} \) is the probability of the device failing during one scan
- \( P_t \) is the probability of the device working at time \( t \)
- \( P_{t+Ts} \) is the probability of the device working at time \( t + Ts \)
- \( Ts \) is the alarm scan interval
- \( f \) is the average failure frequency

The individual cutsets could then be ranked on the basis of their probability, calculated using Equation 9.6.

For single order cutsets, the ranking of the alternative alarm explanations would be the same as if failure frequencies had been used. This is obviously because each failure frequency would have been multiplied by the same constant value, namely \( Ts \).

Evaluating the probability of second and higher order cutsets would have been considerably easier than calculating the frequency of the same cutset. From probability theory this would simply involve calculating the product of the probabilities of the cutset elements.
The major failing of this approach, as argued by Andow [64], is that the relative ranking positions of cutsets with different orders would depend on the alarm scan interval. For example consider the following three cutsets:

1 A
2 B and C
3 D and E and F

If all six faults (A - F) have the same failure frequency of 10 faults per year, and the scan interval is one minute, then the following probabilities can be calculated from Equation 9.6. The figures in parentheses represent the relative likelihood:

1 A $1.9 \times 10^{-5}$ (99.998 %)
2 B and C $3.6 \times 10^{-10}$ (1.9 x $10^{-3}$ %)
3 D and E and F $6.8 \times 10^{-15}$ (3.6 x $10^{-8}$ %)

However, if the scan interval is ten minutes, the relative ranking of the cutsets is now:

1 A $1.9 \times 10^{-4}$ (99.98 %)
2 B and C $3.6 \times 10^{-8}$ (1.9 x $10^{-2}$ %)
3 D and E and F $6.8 \times 10^{-12}$ (3.6 x $10^{-6}$ %)

It can be seen from the likelihoods that the second cutset becomes one order of magnitude more likely as the scan interval increases. However, the third cutset becomes two orders of magnitude more likely.
Clearly, the frequency at which the process measurements are scanned should not affect the likelihood of a fault scenario causing a pattern of alarms. For this reason the method was not adopted within the KBS.

9.1.5 The Indication Interpretation Technique

Given the frequency of sensor faults on process plant, and their significance to the task of alarm diagnosis, the problem of interpreting process indications has been considered in some detail.

The method developed for the KBS, as described in Chapter 5, appears to work well in the following respects:

1. The flexibility of the indication fault modelling approach, enables a wide range of instrument configurations to be represented.

2. The ability to specify fuzzy discretisation ranges for process variable states, helps to minimise the effects of noise.

3. The technique uses a very similar approach to the alarm diagnosis methodology, and therefore integrates well within it.

Whilst the indication interpretation technique, implemented within the KBS, is complete there are a number of possible enhancements which could be included within the methodology. These are discussed in the following text:

The most limiting aspect of the current approach is the use of snapshot indication values. When investigating the probability of a process variable being in a certain state, and hence whether or not the instrumentation is working, the short term history of the indication is not considered. Clearly, this information could be useful when considering the state of the measurement.
For example, if a process measurement is known to exhibit some degree of noise, the standard deviation of the signal, for a number of previous scans, could be calculated. This could then be compared against an average value determined when the sensor was known to be working. If the discrepancy was large, this would tend to indicate that the some element of the instrumentation had failed invariant.

Conversely, some measurements exhibit very little process noise. For example, many temperature indications are very steady because of the thermal inertia of the process. If such a measurement started to exhibit process noise of an unusually high frequency, the measurement could be discounted, or at least the operator could be warned of the problem.

The subject of sensor malfunction detection has not been considered in great detail, because it warrants more attention than the scope of this thesis allows. Other workers such as Anyakora and Lees [48] and Bellingham and Lees [49] discuss the issue at greater length. However, the two very simple examples of sensor malfunction detection, described above, do serve to illustrate the potential benefits involved.

If some degree of sensor malfunction detection was incorporated within the KBS, the extra sensor state information could be integrated into the indication interpretation method relatively easily. At the present time, all the possible explanations for a set of related indications outputting their signals, are first determined in cutset form. The probabilities of the sensor failures are then calculated based on their failure frequency and detection and repair times. If an extra piece of evidence, indicating the working or failed state of an indication was available, the 'a priori' probabilities could be modified accordingly.

Another feature of the indication interpretation method, which has room for improvement, is the modelling of error between indications of the same variable. At present, any two indications are deemed to be in agreement if their difference is less than an error margin calculated as shown overleaf:
Acceptable Error Margin = \( X \times E_q + Y \times E_q \)
\[ \frac{\text{---}}{100} \] (9.7)

Where \( X \) and \( Y \) are the indicated values.
\( E_q \) is the acceptable error band.

Equation 9.7 assumes that the random error associated with an instrument, increases in fixed proportion to the indicated value. This is a valid assumption for some instruments, such as turbine flow meters, but does not hold very well for other instruments, e.g. orifice plate meters. The logical extension to this method would be to develop more specific error models for each type of sensor - sensor comparison required.

9.2 The User Interface

9.2.1 The Explanation Facility

One of the most important features of Expert Systems is their ability to explain how a particular conclusion was reached from the available information. However, the quality of these explanations often varies quite considerably between expert systems. Those that reason from deeper or more fundamental knowledge can provide a more fundamental explanation. Conversely, those expert systems which infer their conclusions from shallow or compiled knowledge, often justify their conclusions simply in terms of the rules used in the inference.

Whilst the KBS described in this thesis is broadly structured like an expert system, with a separate knowledge base, inference engine and user interface, the diagnosis system does not include an explanation facility. This is because the task of developing a comprehensive alarm diagnosis explanation system was thought to require considerable time and effort, and hence was outside the scope of the research project. The following discussion outlines some of the difficulties involved.
When a set of process faults are diagnosed as the cause of a single alarm, the cause and effect relationships are clear, providing that the user understands how the process plant works. Unfortunately, as more and more related alarms are diagnosed in conjunction with the first alarm, the interrelationships between the alarms and their root causes may become less clear. This is because the multiple alarm explanation cutsets contain no structure.

In an attempt to overcome this deficiency, the KBS retains the individual causes of each alarm, within the same PROLOG structure containing the multiple alarm cutset. For example, consider the three alarms A, B and C. Alarm A is caused by fault Z. Alarm B is caused by the process disturbance indicated by alarm A. Similarly, alarm C is caused by the process disturbance indicated by alarm B. If all three alarms are diagnosed together, the multiple alarm cutset will contain one element, namely fault Z. Since this cutset does not define the propagation of the disturbance, the KBS also retains the fact that fault Z causes alarm A, alarm A causes alarm B and alarm B causes alarm C.

Given the programming skills and the necessary time and effort, it would have been possible to engineer a diagnosis explanation facility to display the propagation information to the process operators. However, the effectiveness of such a system would still have been limited in the following respects:

1. The system would be able to display how a set of faults caused the diagnosed alarms. However, it would not be able to explain the absence of other fault scenarios. This is because unlike many rule-based systems, the information presented to the operator is not derived using forward or backward chaining strategies, or even some combination of the two. Instead, the alarm causes are conjugated with each other, checked for logical consistency, and finally reduced into minimum cutsets. These are not easy processes to explain when justifying the absence or presence of certain alarm causes.
The operators would have to have a good understanding of how disturbances propagated through the process, in order to make any valid judgements about the diagnoses. Whilst they should understand how the process "works", their understanding of how the process fails to "work" will usually be much more limited.

Unfortunately, the first limitation cannot easily be remedied without resorting to an alarm diagnosis strategy which reasoned from a very fundamental level. Only then could the diagnosis system explain why a certain fault scenario would or would not cause a set of observed alarms.

The second issue concerning the operator's diagnostic skills raises a more general point. If any fault diagnosis system is going to be used by the operational staff they will require extensive training. Furthermore, to maximise the benefit of installing such a system, the technical competence of the operators may have to improve. In the future, control rooms may be staffed by even fewer, higher qualified operators.

Another point worth discussion is that extensive use could be made of the graphic facilities common on many distributed control systems. The alarm diagnosis information could be displayed, superimposed on the operator's control graphics. The source(s) of the disturbance could be highlighted in one colour, and the propagation of the consequences represented by changing the colour of the process and instrument lines. The operators could then step through alternative alarm explanations and visualise the meaning of the diagnoses in terms of their effects on the process units, lines and instrumentation.

By employing the same graphics as used to display the process control information, the operators would be more familiar with the user interface, rather than trying to remember how to to use the diagnosis system when an alarm occurred. Furthermore, the ability to access the control loops and schemes alongside the diagnostic information would improve the operators chances of returning the plant
to a full operational state. The difficulty of this approach is that the diagnosis system would need to be well integrated with the process control system. This issue is discussed in more detail in Section 9.4.

9.2.2 The Operator Driven Diagnosis System

As discussed in Chapter 6, the diagnosis system is operator driven, which means that an alarm is not diagnosed until requested. This approach was simple to implement thus allowing the research to focus on other areas. However, from a computing standpoint this is not a very efficient approach for a real-time knowledge based system. This is because of the extremes in computational load it is likely to experience. When there are few process alarms, the diagnosis system will be idling for the majority of the time. However, following a process fault, there is likely to be a high incidence of alarms.

Given that the diagnostic task is computationally intensive, the computing resources need to be allocated efficiently, to maximise the ability of the system. However, in a critical process situation, the operator will require all his efforts to understand the alarm diagnoses, and to formulate a plant to alleviate the problem. The efficient scheduling of the KBS will not be a high priority and therefore the performance of the system may not be optimal.

Andow [25], identified the following two distinct modes of operation, and suggested that it was clearly possible ( and perhaps desirable ) to design systems that lie between these two extremes:

1 The system is directed by the operator to focus on particular abnormal events.

2 The system continuously monitors the plant state and offers advice to the operator when it is appropriate.

Ideally some sort of real-time scheduler would be required to control the operation of an alarm diagnosis system, similar to those described by Dhurjati et al [29,30] and Paterson et al [35-37]. One
method of maximising the effectiveness of the KBS would be to perform some type of pre-processing, to prioritise the alarms before their diagnosis. This could involve grouping the alarms into sets of related symptoms. The potential consequences of each set of alarms could then be predicted, and a cost penalty determined based on economic, environmental and safety considerations. The diagnosis system could then focus the operators attention on the most important set of alarms.

Even within one group of related alarms, the diagnosis system could direct the operators attention to the disturbance with the most damaging consequences. However, it would be important for the operator to have ultimate control of the system, so that it could be directed to diagnose a chosen set of alarms.

In addition, the real-time scheduler could optimise the order in which the alarms were conjugated. For example, if four or five related alarms were detected shortly after each other, the diagnosis execution time could be minimised by conjugating the alarms sharing the greatest number of common failure modes first. This would not only reduce the number of alarm explanations generated at any point in the diagnosis, but it would also minimise the number of cutsets that were disregarded to achieve a reasonable processing time.

9.3 Synthesising The Diagnostic Rules Using FAULTFINDER

Without reiterating the modelling discussion included in Section 7.3, it is fair to say that some problems were encountered when using FAULTFINDER to model both the reactor charging system and the batch distillation column. However, the majority of these were due to simple programming limitations, or slight differences in the objectives of the methodology, rather than a weakness in the methodology itself. The few problems that were related to the basic modelling approach (the use of signed functional equations) were unavoidable, yet acceptable, given the benefits in terms of modelling simplicity.
Furthermore, many of the deficiencies in the alarm fault trees synthesised by FAULTFINDER were genuinely unavoidable, given that the requirements of the alarm diagnosis system were unclear until it had been prototyped. The main differences between the design type fault trees currently generated, and those required by the alarm diagnosis system are as follows:

1 Most design type fault trees involve the development of the top event to all its root causes. Conversely, the alarm fault trees generally only need to relate the top event alarm to the nearest alarmed process variable deviations. The ability to insert break points into the system configuration would overcome this limitation.

2 In order to detect logical inconsistencies, the alarm diagnosis system requires to know when certain key items of process equipment are assumed to be working, if their failure modes are considered in other alarm fault trees. Notable examples are control loops, which can propagate a disturbance without significantly effecting their controlled variable, and in certain situations, sensing equipment. Again relatively little effort would be required to implement these modifications.

3 Boundary condition information is sometimes very important to ensure the logical consistency of the alarm explanations. These extra conditions are in the form of other variable states, either expected or NOT expected as consequence a fault. Unfortunately, this information would be more difficult to synthesise within FAULTFINDER because it would involve propagating the effects of a disturbance, rather than tracing the causes. It is estimated that implementing this feature would require more effort.

The modelling procedure would also benefit from an improved interface between FAULTFINDER and the rulebase compiler. The two systems modelled as part of the current research were relatively small, therefore the task of converting the alarm fault trees into the compiler input format was trivial. This may not be the case for larger systems.
The design of the interface would obviously depend on whether the FAULTFINDER program was customised to generate alarm fault trees. This is the principle reason why the interface software was not developed. However, if the output from FAULTFINDER could be automatically converted into diagnosis rules, the results would still require to be manually scrutinised, to ensure their integrity.

9.4 System Architecture And Implementation Tools

The KBS described in this thesis was never directly interfaced to live process data because of the difficulties of implementation. Despite the lack of practical experience, the issue of integrating a fault diagnosis system within a computer based process control environment still warrants the following discussion:

Most modern computerised process control systems are structured in hierarchies. At the lowest level the distributed control system (DCS) contains the individual flow, level, pressure and temperature controllers. These either operate as single loops or may be cascaded together to improve the control stability.

The level above the regulatory controllers is usually termed advanced control. Typically these control schemes attempt to improve the economics of the process by saving energy, maximising or minimising certain process variables up to constraints or reducing the fluctuations in process variables. These control applications may be configured within the DCS system, or a separate computer interfaced to the DCS.

In some cases there may be a third tier in the hierarchy which is optimising the process, taking into consideration the current operational constraints and the prevailing economic situation.

The requirements for a fault diagnosis system are that its should have easy access to the process variable and alarm information, at least the current values and preferably their short term histories. This dictates that the host computer needs to be quite closely, coupled
to the DCS, rather than being interfaced to a management information system database, which may only be updated with process data once every few minutes. However, the ability to access more general information stored within such a database could prove useful.

Many DCS vendors have developed their own process control computers which can rapidly access the DCS information. These are ideal locations to implement advanced control software. However, in most cases they are unlikely to support the type of AI shells or toolkits usually required to implement a fault diagnosis system. As a consequence, the most probable environment for a fault diagnosis system would be a VAX, SUN or IBM PC workstation for example. Many DCS systems have the ability to interface to at least one of the aforementioned computers.

The situation, however, may change in the relatively near future. At present one DCS vendor (The YOKOGAWA Electric Company) is already marketing a frame based KBS which is directly interfaced to the DCS system. The software may lack the functionality of more mature packages such as KEE, MUSE or G2, but this is likely to change with time. The current problems of integrating AI technology within a process control environment will probably become much less of an issue in the relatively near future.

9.4.1 The POPLOG Language Environment

A detailed discussion of the advantages and disadvantages of the PROLOG language has already been included in Section 6.2.1. However, to summarise it is thought that PROLOG, in conjunction with POP11, was an excellent choice for the research project. The inherent characteristics of the language, such as its goal driven searching procedure, the list processing facilities and ease with which natural language operators could be defined, were invaluable assets. As a result, the tasks of conjugating alarm explanations, checking them for logical consistency and reducing them into minimum cutsets were so much easier to implement.
On the negative side, PROLOG was not designed as a real-time tool and therefore many of its earlier implementations (including POPLOG) exhibit undesirable features, such as stopping to perform garbage collections. By careful programming these drawbacks can be minimised, but they still illustrate the weakness of the language in this domain.

Outside of the research environment, the usefulness of PROLOG in the development of a commercial scale alarm diagnosis system would clearly depend on many factors. More mature implementations of the language are now available. However, these versions may still be unacceptable on the grounds of robustness, or that they do not integrate with existing software. More conventional languages, such as 'C' or ADA, might be preferential in these respects, but would require considerably more programming effort.

At the other end of the extreme, it is unlikely that most off-the-shelf KBS shells would have the necessary flexibility to implement the methodology described herein. However, real-time AI tool-kits such as MUSE [65] could offer attractive solutions.

9.5 Representing The Behaviour Of Process Plant

It is not difficult to observe from the literature that there are a great variety of approaches to the problem of fault diagnosis, and particularly alarm diagnosis. Despite their differences they all have at least one common feature, namely that they require some model of how the process behaves in response to fault conditions. As discussed in Chapter 2, these models range in complexity from simple definitions of the relationships between process alarms, to more quantitative descriptions of process behaviour.

The more complex approaches involving pseudo steady state models of the process, such as those reported by Dhurjati, Lamb and Chester [29,30] and Kramer [62], seem to perform well from a technical standpoint. Furthermore, because the process model is more fundamental in nature, it is better equipped to diagnose unforeseen problems by reasoning from the "deeper" knowledge of system behaviour.
Conversely, non-qualitative cause and effect methods, such as those based on fault trees and cause consequence diagrams, are not highly sensitive detectors of violations of process heat or mass balances. However, they can be efficiently used to relate process alarms to each other and to prime cause failures.

From a purely academic standpoint, the closer the fault propagation model reflects the reality of the process, the more accurate the resulting diagnoses may be. Unfortunately, as the model becomes increasingly more complex, so the time and effort required to initially generate the process model, and to maintain it throughout the lifetime of the plant, will increase. For example, the cost of developing the FALCON system and applying it to the adipic acid plant was estimated as 1 million $US [29]. Given the technology involved, this estimate may have been rather optimistic.

Whilst a large proportion of the project cost was undoubtedly incurred in developing the diagnosis environment, the cost of generating and proving the semi-qualitative knowledge base for the process must have been significant. For this investment the system could uniquely identify any one of 100 possible failure modes.

If any fault diagnosis approach is to achieve sustained commercial exploitation, the economics for implementing such a system will have to be favourable. More specifically the potential savings in plant shutdown time, reductions in accidental damage and compensation claims and possible reductions in manpower will have to outweigh the cost of installing and maintaining the system.

The FALCON project has served to illustrate both the benefits and costs of one approach to fault diagnosis. However, in order to determine the most cost effective solution it is necessary to investigate the merits of a number of fault diagnosis methodologies, based on a variety of representations of process behaviour. Two of the key objectives of this project have therefore been to determine the quality of the diagnoses it is possible to produce from the fault trees generated by FAULTFINDER and to evaluate the difficulty of the modelling task.
The issues of diagnostic quality and modelling effort are discussed elsewhere. However, it is perhaps useful to briefly discuss the possible costs involved in developing a marketable fault diagnosis system, based on the methodology described within this thesis, and to estimate the probable modelling effort required.

Providing that any development team had the correct blend of experience, the right software and hardware tools and assuming that they were not too ambitious, it is estimated that they could develop a fault diagnosis system with between 0.5 - 1.5 man years of effort. At present rates this could translate to a cost in the region of £ 50K to £ 150K. If they chose to significantly enhance the methodology the cost might double or quadruple. In addition, the cost of marketing any product might be approximately the same as the software development.

The cost of applying a fault tree based diagnosis system to commercial scale plant is somewhat more difficult to estimate. As a rough guide, it should be possible to model the causes of each alarm using between 2 - 6 hours of the system modeller's time and 1 - 4 hours of the process experts time (checking the synthesised knowledge, performing any required plant tests etc.). The unit cost per alarm could, however, vary more significantly depending on the nature of the plant and the level of instrumentation. With more instruments each alarm should represent fewer process faults. The unit cost per alarm would decrease, but the overall modelling effort for all the alarms would then increase.

In order to properly assess the benefits of installing a fault diagnosis system, the operation of the process or similar processes would need to be scrutinised in some detail. Historical information regarding the frequency of plant trips and the cost of re-commissioning the process would provide the main basis of the estimate. Taking into account the possibility of preventing major plant upsets or catastrophes would involve quite detailed analysis and even then the results would be relatively approximate.
9.6 Future Developments

The overall aim of any alarm diagnosis system is to help the operators, in the event of a plant upset, to understand why the process performance has degraded. This is so that they can either take compensating action, or shut down the plant with the minimum risk and cost. Many fault diagnosis methodologies have been developed to assist the operators in the first stage of this task, namely identifying the potential alarm causes. Clearly, it would also be desirable to predict the future plant state and to help the operators steer the process to a safe working or shutdown state.

Both these issues have been considered before. The two disturbance analysis projects, reported in references [9 - 17], aimed to predict the future propagation of disturbances, and to recognise the significance of these events. Nelson and Jenkins [66] described the response tree system. This was designed to guide operators of nuclear power plant from a hazardous situation to a safe state.

With qualitative descriptions of the process behaviour, such as the fault tree based approach described in this thesis, it would be possible to predict the future plant state in qualitative terms. However, the absence of process gains and temporal information would severely restrict the accuracy of the disturbance predications, both in terms of their severity and expected times. These limitations could only be overcome with the aid of quantitative information.

Similarly, the information required to guide the operator from an unsafe process state to a safe state, could not be synthesised from any fault propagation models. Instead the experience of process operators and engineers would need to be elicited. Perhaps the economics of the process would also need to be taken into consideration. This would be a major undertaking in its own right, and therefore could only be justified in relatively few situations.
In the longer term, it is quite possible that advanced process control applications will incorporate fault diagnosis functionality as a standard feature. This is because both technologies could benefit from being integrated with each other.

One of the developing areas in process control is the field of model based control. This involves identifying the dynamic and steady state relationships between the process input and output variables, and building this into some form of model. With this information it is then possible to predict the future plant state and calculate control moves to steer the process to the correct state. For example, a technique called dynamic matrix control has been used by SHELL and the DMC Corporation to control complex interacting processes, such as fluidised bed catalytic crackers, with good effect.

Whilst such model based control techniques are very powerful, they generally cannot update their process models automatically. The notable exceptions are single variable self tuning controllers. As a result, if the process dynamics or gains change significantly, the model based control applications can worsen rather than improve the control of the process. The only method of avoiding this problem is to de-tune the controllers so that they cope with a wider range of process conditions. This then results in good but non-optimal control.

The difficulty with dynamically updating the process model information is ensuring that the changes in the identified process dynamics are sensible and not due to factors such as process equipment failures or exceptional operational circumstances. Clearly this problem has an overlap with the more general task of fault diagnosis.

Similarly, there is an overlap in the information requirements for fault diagnosis and model based process control, namely the process model. One theoretical approach to fault diagnosis is to simulate the process behaviour with a mathematical model and to compare the simulated variables with the observed process measurements. Any discrepancies between the simulation and the observed plant state can then be used to infer equipment or instrumentation faults. This concept is not new, however, to date the
biggest problem has been justifying the cost of developing the process simulation model for the diagnosis system alone. This would no longer be a major limitation if the control applications could identify the process dynamics themselves.

9.6.1 The Design Of The Process Model

In any future development of an integrated process control and fault diagnosis system, the two tasks would be best served by a model which was based on the process flowsheet information, rather than just a linearised approximation of the relationships between the input and output variables. For example, if the model included knowledge of the process connectivity, and the types of process units included in the system, the relationships between the process variables could be modified if the plant state was known to change.

The task of fault diagnosis would then involve adapting the process model until the observed measurements matched the simulated data. The modifications needed to achieve the match could then be used to infer the cause(s) of the discrepancy. Unfortunately, the process model would still need to be augmented with qualitative information in order to relate a model discrepancy to a process fault or human error. This causal knowledge could probably be synthesised using a similar approach to that employed within FAULTFINDER. However, rather than relating process disturbances to prime cause failures, as is the case in this thesis, the knowledge would need to explain discrepancies between the simulation and the process.

The advanced control applications could utilise the modified process model to predict the future plant state, and then determine a set of control moves to maintain the process stability. If the control system predicted that the disturbance was compensatable, the operator could be advised of the situation. However, if the disturbance was determined to be uncontrollable, the operator could be advised of the consequences of the fault so that he could take remedial action or decommission the process.

294
9.7 Conclusions

A method has been developed to identify the causal relationships between process alarms and to diagnose their root causes in terms of basic equipment failures. One of the key aspects of the resulting technique is that the diagnostic knowledge is based on systematically created alarm fault trees, synthesised using the FAULTFINDER suite of programs.

The identified strengths and weaknesses of the developed alarm diagnosis methodology are summarised in Sections 9.7.1 and 9.7.2.

9.7.1 The Strengths Of The Methodology

1 The operator's attention is drawn to the significance of each new alarm, as it is diagnosed, in terms of whether it is another manifestation of an already suspected fault, or a fresh indication of a new problem. In the former case, the potentially related alarms are considered in conjunction, so that the common causes can be identified. However, their independence is not overlooked if a plausible explanation (a minimum cutset) exists.

2 The methodology can discriminate between process faults, providing that there are sufficient process measurements. The quantitative severity of the faults cannot be determined because of the qualitative models used within FAULTFINDER. However, the operators should be able to assess the magnitude of a problem from the process indications.

3 The alarm diagnoses are not restricted to single process equipment failures. Instead, the individual explanations are retained or rejected on the basis of their likelihood, calculated from their 'a priori' failure frequencies. The technique is therefore both robust and flexible in its interpretation of fault conditions. Furthermore, it does not need to assume that the integrity of the process instrumentation is unusually high.
The vast majority of the alarm diagnosis information can be systematically created using the FAULTFINDER suite of programs. Because of this modelling route, the alarm diagnosis methodology can only reason about a process fault using qualitative information, rather than identifying faults in more quantitative terms. However, it is estimated that the process modelling phase, in any implementation of an alarm diagnosis system, will incur a significant proportional of the overall project costs. There is clearly an advantage in minimising the modelling effort, if it is still possible to achieve acceptable diagnoses.

9.7.2 The Weaknesses Of The Methodology

Because of the limited scope of the research project, many of the real-time aspects of the alarm diagnosis problem have not been addressed. For instance, the diagnosis engine is operator driven, rather than under the control of a resource scheduler. As a consequence, the handling of alarms prior to their diagnosis is crude. For example, the potential causal relationships between process alarms are not identified automatically, but only when they are diagnosed. This limits the sophistication of the causal searching strategy which attempts to group the alarms into sets of related symptoms.

For example, the method described in this thesis can handle one alarm failing inactive in a causal chain of events, but cannot take account of two or more neighbouring alarms failing inactive. Whilst this latter scenario is extremely unlikely in normal circumstances, common mode failures can undermine the security of this assumption.

The exclusion of temporal information from the fault propagation models (within FAULTFINDER) restricts the ability of the diagnosis methodology to reason with time. The task of matching observed symptoms to potential causes requires little temporal information. However, time can be an important aspect when the diagnoses are being refined. For this reason, it has been necessary to apply the
alarm coexistence heuristic. This appears to work well for lasting faults (those that continue to exhibit their symptoms until repaired), but less reliably for intermittent faults.

It should be possible to identify those alarm explanations which cannot be reliably rejected with the alarm coexistence heuristic. However, this would involve extending the fault basic diagnosis methodology to include more quantitative information.

3 The method used to derive the causes of a group of related alarms is quite complex, and therefore difficult to explain to the operators. Clearly the explanation facility used within the KBS could be improved with a better user interface. However, to win the confidence of the operational staff, the diagnosis procedure needs to be more interactive. For example, the methodology should be able to take more account of information volunteered by the operators. Similarly, the system should be able to justify the presence or absence of a particular fault scenario with greater clarity.

4 The alarm diagnosis methodology is combinatorial by design, which means that it is sometimes necessary to reject alarm diagnoses, in order to achieve an acceptable execution time. Whilst the method employed does make certain checks to minimise the loss of useful information, there is room for improvement, given the difficulty of regenerating the lost information.

9.8 Future Work

The work described in this thesis demonstrates the basic viability of the fault diagnosis methodology. However, the scope of the research project was naturally constrained by time and resources, consequently many aspects of the diagnosis technique have room for future development. The purpose of this section is to discuss the key areas where future research would prove most beneficial and interesting.
9.8.1 Applications Of The Diagnosis System

Clearly the intention of the research project has been to develop a generic alarm diagnosis methodology, which is applicable to as many process systems as possible. Unfortunately, some of the specific characteristics of the two modelled systems have undoubtedly influenced the content of the work. Unless the methodology is applied to a broader range of process situations, these system specific features will be difficult to identify.

Ideally, any future trials of the methodology should include the integration of the diagnosis system with the process monitoring computer. This would enable the functionality of the package to be properly investigated within a real-time environment. Furthermore, it would also provide an excellent platform to assess the ability, or willingness, of the process operators to accept a fault diagnosis system, to consider the strengths and weaknesses of the user interface and to determine the user requirements for the alarm explanation facility.

9.8.2 The Use Of Quantitative Diagnostic Models

The alarm diagnosis methodology described in this thesis uses qualitative models of fault propagation behaviour. As a result, the KBS cannot diagnose the causes of a process alarm with more quantitative information, such as that used within the FALCON system [29,30], because it has not been designed to do so. Even if the reverse were true, a delicate balance would still need to be achieved so that the benefits of deriving the bulk of the causal knowledge usingFAULTFINDER, were not undermined by the necessity to model significant areas of the process in quantitative detail.

Despite the fact that extending the fault diagnosis methodology to include such extra information could prove difficult, the diagnostic accuracy would probably be enhanced in certain situations. A cautious investigation of the benefits of extending the methodology
in this direction would therefore prove interesting, and might result in noticeable improvements in the robustness of the diagnosis technique, for relatively little modelling effort.

For example, by examining whether or not a mass or heat balance held over a particular section of process plant, it would be possible to enhance or reduce the likelihood of certain classes of alarm explanations. The applicability of this technique would obviously depend on the level of process instrumentation, because in order to calculate a mass balance over a section of plant, the inventory changes in mass would need to be measured. However, if this were the case, any mass imbalance which could not be explained in terms of random instrument error would either indicate a gross measurement error, or leakage of material to or from the system within the mass balance envelope. In this situation, the likelihood of any faults which caused a mass imbalance could be enhanced. If the reverse were true, their likelihood could be reduced. This type of information could be accommodated within the boundary conditions of each alarm explanation.

Another potentially useful source of information would be the first derivatives of the process variable values with time. These could be calculated within the process control system and then discretised into fuzzy membership functions like any other process variables. Unusually high or low rates of change of the process variables could then be linked to certain fault conditions. For example, a high rate of change might be indicative of a leakage, a low rate of change could be indicative of a blockage type fault.

From the standpoint of deriving the fault propagation knowledge, it would be useful if any weaknesses in the qualitative models (which cause incorrect alarm explanations) could be automatically identified at the modelling stage. These special areas could then be targeted with extra quantitative information.

Similarly, it would be interesting to investigate the possibility of enhancing the fault tree synthesis package, so that it could highlight areas where simple mass and heat balance envelopes could be
constructed. The violation of these balances could then be automatically related to the modelled process faults. In the same way, the effects of the process faults, in terms of the rate of change of the process variables, could also be included within the fault tree synthesis models.

9.8.3 Target Applications Of Diagnosis Systems

Whilst the academic literature describes many different approaches to the problem of fault diagnosis, there appears to be little discussion about the relative merits of applying the technique to different process situations. Will the potential users of such technology invest in systems that aim to diagnose faults across a wide section of process plant, or is there greater interest in systems that can identify more specific faults in certain key unit operations?

The first application will require a great deal of process specific knowledge, but the improvement in the operator’s diagnostic skills may only be relatively marginal. However, a diagnosis system which is solely concerned with the behaviour of a key unit operation, such as a catalytic cracker within the oil refining sector, should yield significantly better diagnoses in the narrow domain of application. Furthermore, the unit operation specific knowledge should be more portable between similar processes.

The answers to these questions, gained through more collaborative research between academia and industry, could significantly influence the direction of current research and therefore shape the form of the technology in the future.
References


64 Andow, P.K., Private communication, 1987.

