

https://dspace.org/

Fault recovery in process control

Horn, Timothy Andrew

2015-02-05
http://hdl.handle.net/10539/16894

Downloaded from a DSpace repository. https://dspace.org/

Hazard and Operability (HAZOP), [1, 7], starts from the process state variable and considers their deviations from normal behaviour in the presence of faults.

Cause-consequence analysis , [1], involves what sequence of events lead to a failure, and also the conditions under which these events take place. This can easily be changed to a fault tree.

The most important advance in implementing fault diagnosis can be achieved by using what is termed an " Expert System ".An expert system is a computer program designed to simulate the behaviour of a human expert in a given field, [7]. The basic structure of an expert system is shown in Fig 2.5 .

The knowledge base is made up of knowledge of two types, [28]. The first type is the facts about the process or plant, ie. the physical behaviour of the plant, and an associated data base concerning this behaviour, (the same as the modelling systems as in section 2.1) . The second type of knowledge is the heuristic knowledge, or what is termed good practice and good judgement in the field. This is basically experience gained by known experts in the particular field, and applied systematically in the expert system.

In addition to knowledge, an inference procedure is required, ie. the method of reasoning in the expert system.

Other essential parts of the expert system are the knowledge acquisition facilities whereby the knowledge can be inputted into the knowledge base, and the input/output facility whereby the data and relevant operations of the plant can be included.

APEX 3, [7], is a general expert system shell for fault diagnosis, where the " inference engine " has been developed without assuming any particular type of knowledge base, where the goals defined are determinable faults, as in fault trees.

With particular reference to APEX 3, [7], the following types of inference control mechanisms are available :-

(1) Forward chaining - from the top event down, similar to HAZOP studies, [1] ;

(2) Backward chaining - looks at the effect of a goal, or fault, similar to FMEA study, [1] ;

(3) deciding whether to forward or backward chain - backward from most likely goal, (fault), else forward ;

(4) pre-condition handling, ie. implies a goal depends on another event with a measurable certainty, where probabilities of certain events or faults can be input by the user interactively ; and

(5) user override facilities - enables user to enforce forward/backward chaining always, and allows user to determine what particular faults should be tested for and override certain forward or backward chaining sequences wherever necessary.

The APEX3 system operates interactively with the user to supply external information to complete the knowledge base in the investigations of the implemented fault tree.

2.4 Fault Diagnostics Evaluation

The three stages of fault diagnostics, viz. modelling, fault detection and fault analysis have been discussed.

With regard to modelling, the simpler the model, and the less interconnected the sub-systems are, the easier the model is to develop and diagnose for faults. Even a exceptionally complex model cannot be trusted, as the model is only a model, and usually all equations describing the systems are simplifications of the actual operating principles. Also, the availability and accuracy of data and input parameters is limited to the data acquisition equipment, be it temperature or level transducers, or even more complex or intelligent measuring equipment.

Different classes of models describe the viewpoint of the fault diagnostician or analyst, and pinpoint the critical faults from this viewpoint only.

Fault detection systems involving state and parameter estimation and pattern recognition schemes were discussed. The state and parameter estimation is good for the model development and the refining of control and fault recovery strategies, whereas the pattern recognition technique is better suited to the fault detection and classification or diagnosis of the faults.

In most practical cases, the complexity of the system hardware and software design increases as the number of variables to be measured increases. Also, some of the variables measured are only for indication and/or management information and thus do not require alarms or fault detection applied to them. In a comprehensive fault detection, diagnosis and recovery scheme all measured variables and inputs have to be handled in the fault handling scheme.

For systems implemented using small, dedicated microprocessor controllers, there is a limit on the program space and the number of inputs and outputs available. This means that only the critical fault conditions should be considered. This is discussed in more detail in the case studies in section 5.

An alarm system must take into account corrective actions performed and the time it takes to do these. the alarm system must not alarm until it is certain that the corrective action being performed has not succeeded in correcting the fault.

Fault analysis involves the the systematic analysis of faults in a process. Details of the fault-tree or flowchart approach were given, along with other approaches such as Failure Mode and Effect Analysis, (FMEA), and Hazard and Operability, (HAZOP).

The application of the computer in the fault analysis then approaches the " Expert System " approach to fault analysis. The major problem which occurs in expert systems is where the inference procedure and the knowledge base are not sufficient for the expert system to correctly simulate a human expert, and thus should be continuously refined as new experience or heuristic knowledge is gained.

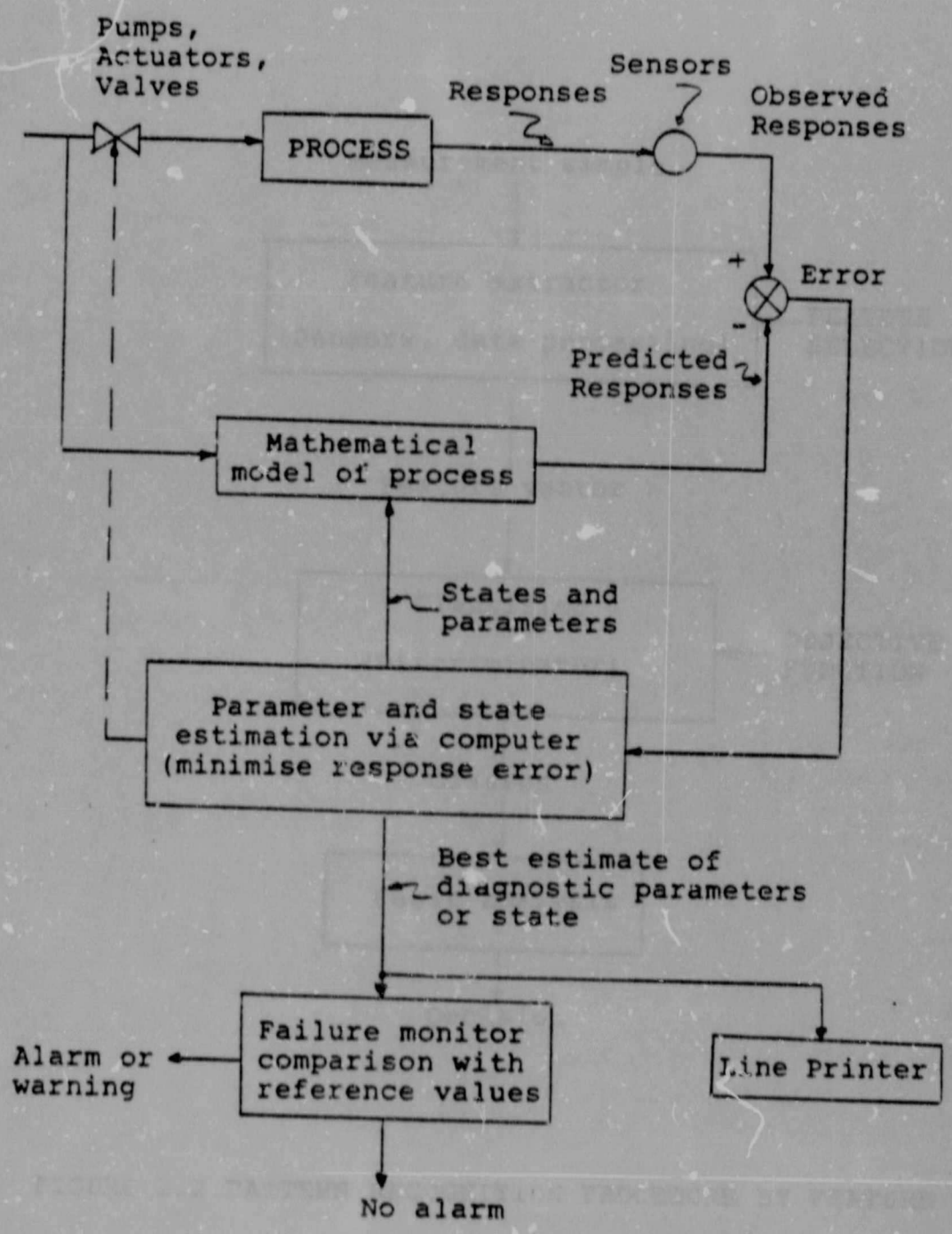

FIGURE 2.1 INFORMATION FLOW FOR STATE AND PARAMETER ESTIMATION

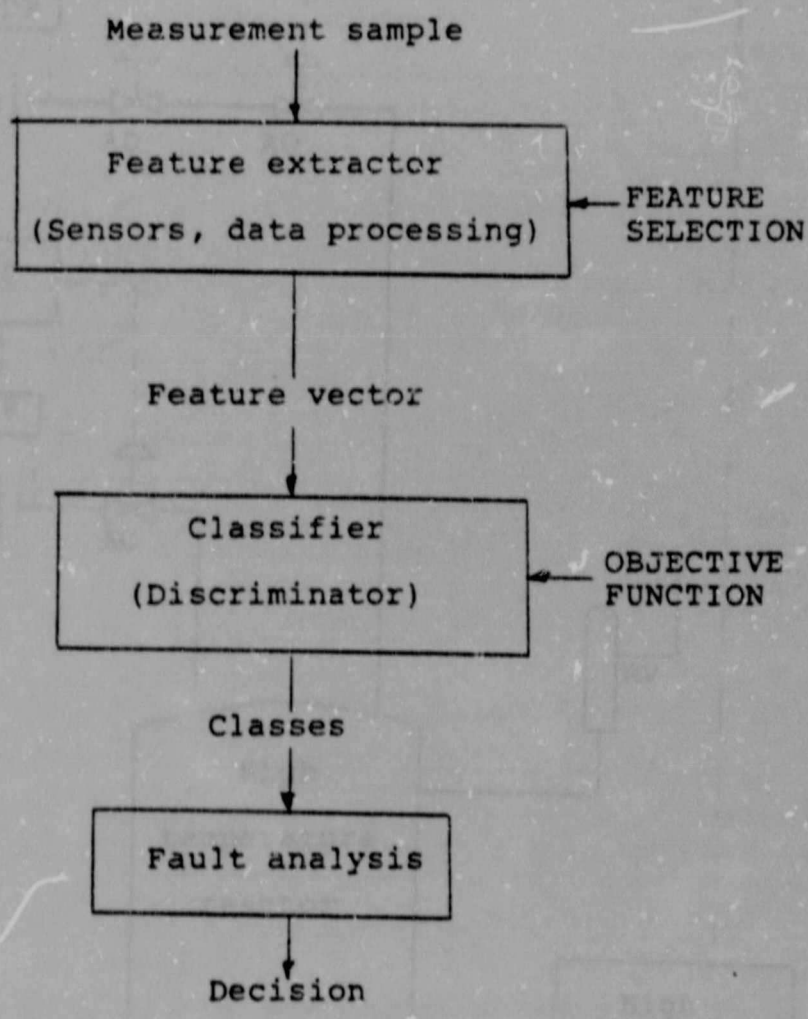

FIGURE 2.2 PATTERN RECOGNITION PROCEDURE BY FEATURE EXTRACTION AND CLASSIFICATION

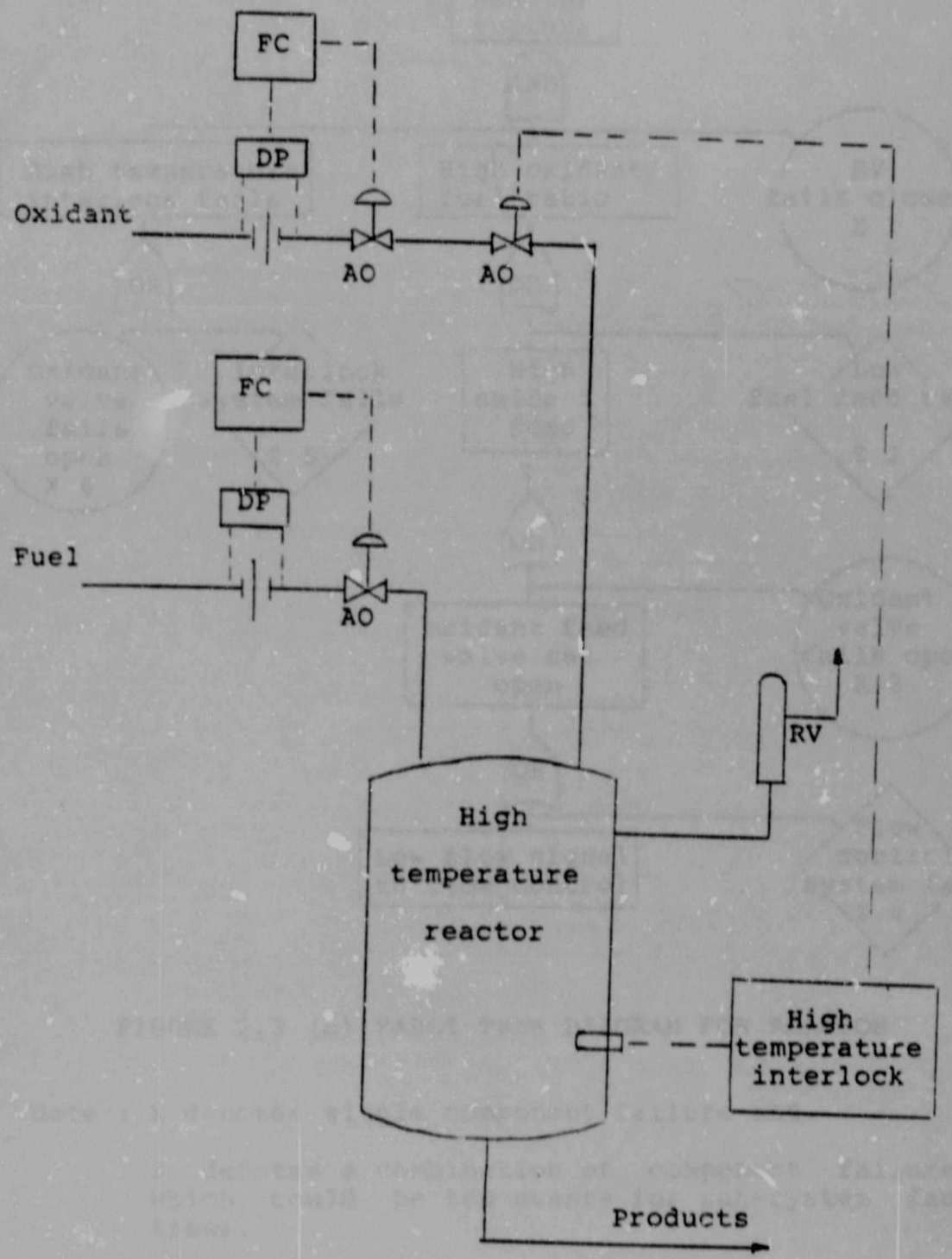

FIGURE 2.3 (a). REACTOR AND CONTROLS

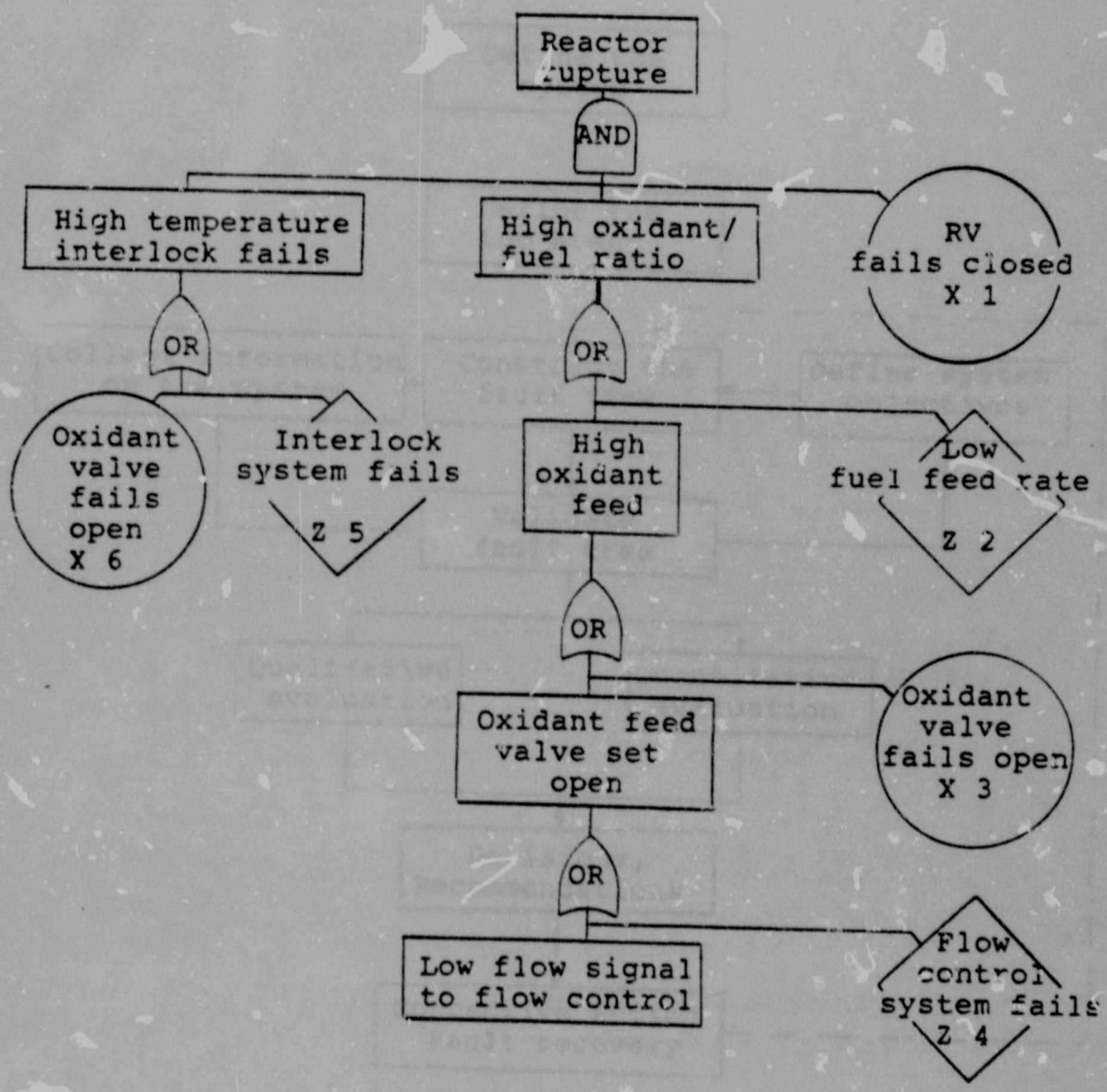

FIGURE 2.3 (b) FAULT TREE DIAGRAM FOR REACTOR

Note : X denotes single component failure and

Z denotes a combination of component failures, which could be top events for sub-system fault trees.

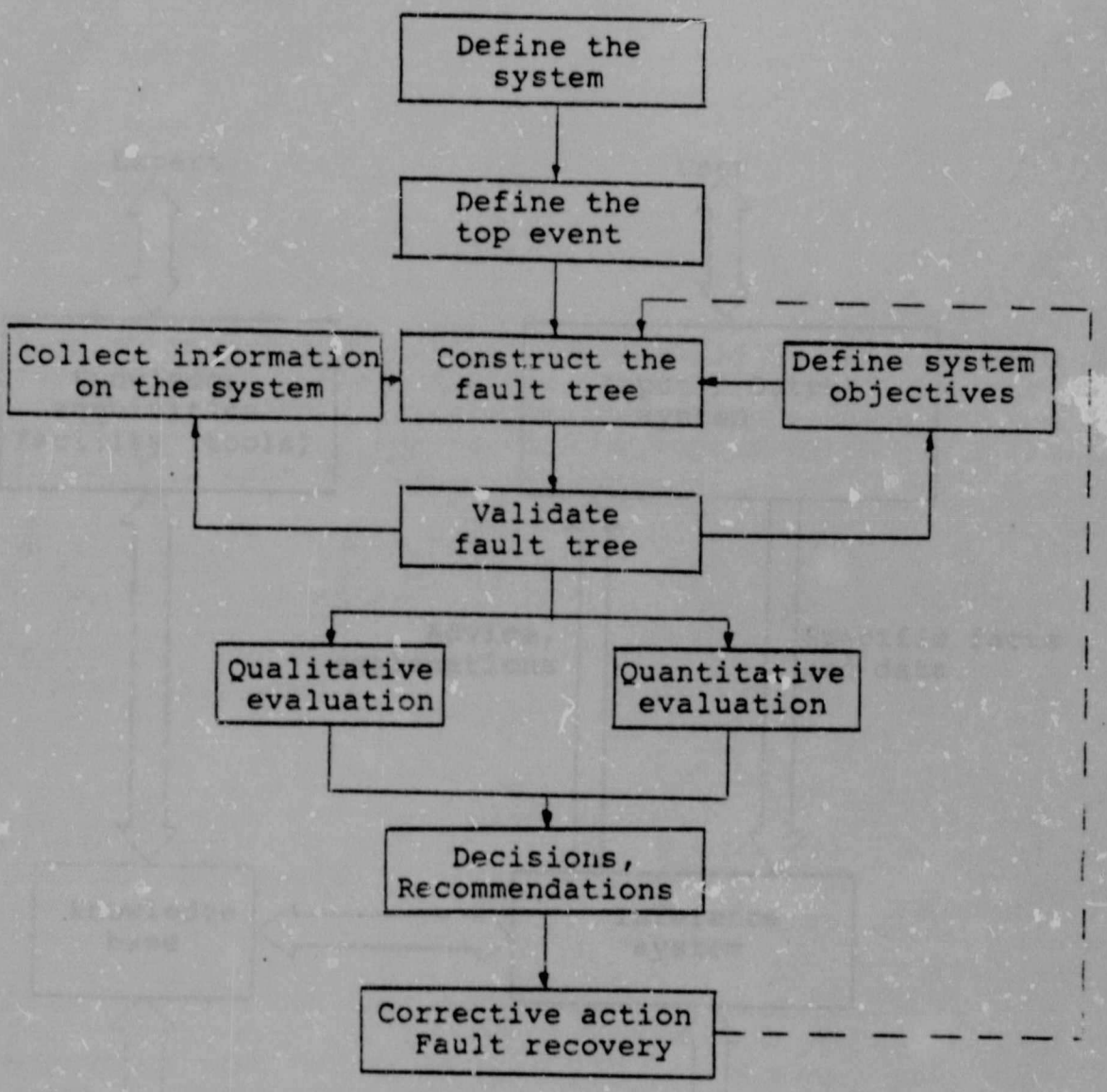

FIGURE 2.4 STEPS IN THE FAULT TREE ANALYSIS

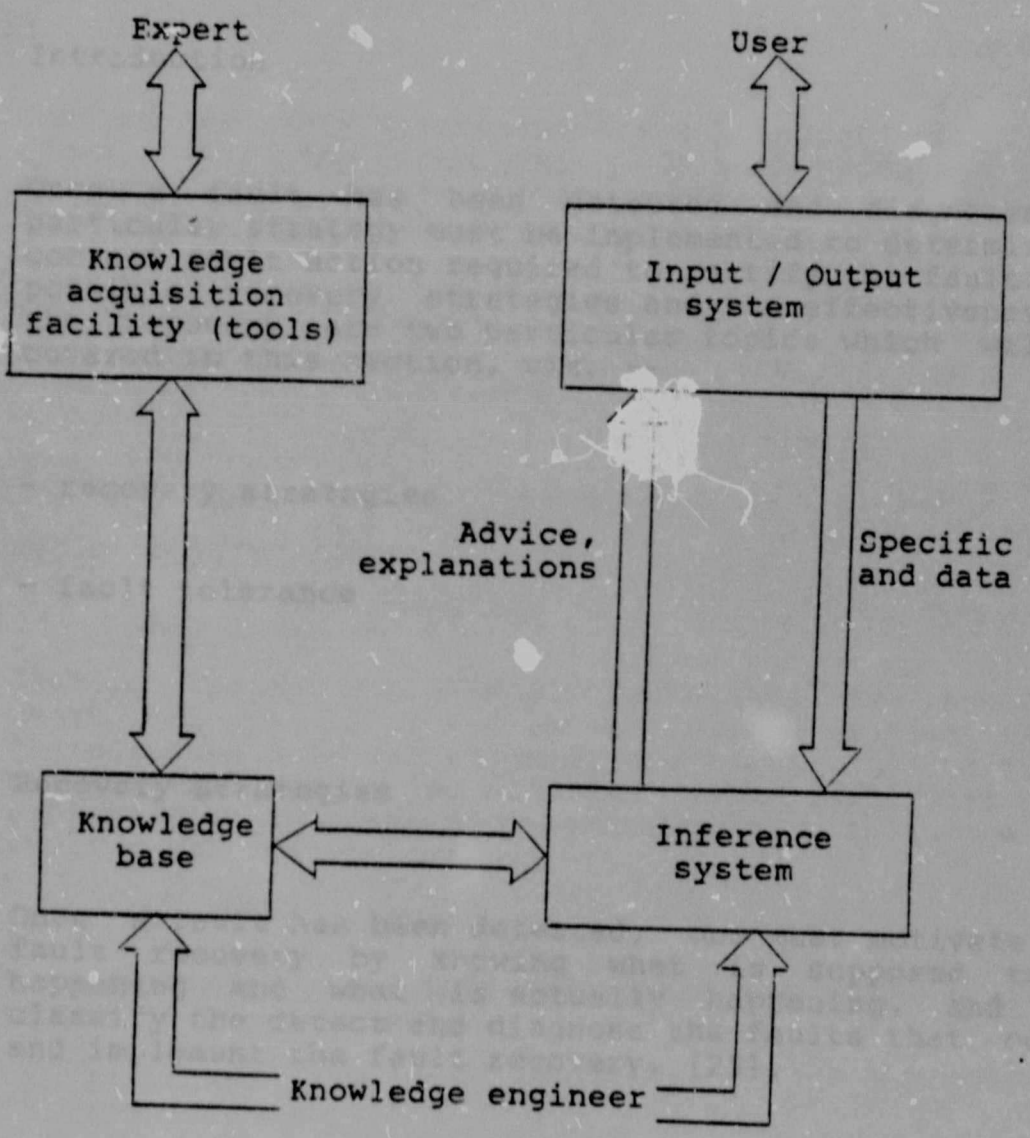

FIGURE 2.5 BASIC STRUCTURE OF AN EXPERT SYSTEM

3. FAULT RECOVERY.

3.0 Introduction

Once a fault has been detected and diagnosed, a particular strategy must be implemented to determine the correct fault action required to rectify the fault. Some possible recovery strategies and the effectiveness of the recovery are two particular topics which will be covered in this section, viz. :-

- recovery strategies

- fault tolerance

3.1 Recovery Strategies

Once a fault has been detected, one must motivate the fault recovery by knowing what is supposed to be happening and what is actually happening, and then classify the detect and diagnose the faults that occur, and implement the fault recovery, [25].

Recovery strategies can be grouped into two different categories, viz. simple and complex recovery strategies or techniques. These are shown in Table 3.1 .

Simple recovery involves techniques to readjust some control, ie. adaptive control, [1], and basically consists of two types, viz. digital and analogue recovery techniques, (see Appendix C.).

For digital control systems, digital fault recovery is simply the switching ON and OFF on pumps and motors, or the opening and closing of valves, etc. , eg. while filling a tank, when the tank full limit switch is activated, close the inlet valve.

For analogue control systems, fault recovery is regarded as true adaptive control, eg. controlling a temperature loop according to a set point and the actual value measured, by performing PID (Proportional, Differential and Derivative) control.

Complex recovery strategies are subdivided into three types, viz. the combination of both types of simple recovery types, back-up systems or in-built redundancy techniques, and state restoration techniques.

The first type, ie. the combination of the two simple recovery types is seldom considered complex, as this is literally all that it means.

3.1.1 Redundancy Techniques

Back-up systems and in-built redundancy are important where sub-systems of a plant are deemed critical to the correct and safe operation of the plant. All critical sub-systems should be provided with duplicated sub-systems, which can be operated when faults on the working sub-systems fail.

A concept of " protective redundancy " , [5], involves two types of redundancy, viz. :-

- masking
- dynamic

Masking protective redundancy hides or masks the effects of faults, ie. as long as redundancy is effective, faults remain totally invisible to the system, (hardware and software), or to the operator.

An example of masking protective redundancy is TMR (Triplicated Modular Redundancy). To correct a fault on this system,the faulty component is forced to read fault-free data and continue from there, (ie. state restoration , section 3.1.2). If errors are persistent, then the system fails to a simplex/duplex mode, with only the two healthy systems operational. This replication of systems is termed static masked protective redundancy.

For analogue control systems, fault recovery is regarded as true adaptive control, eg. controlling a temperature loop according to a set point and the actual value measured, by performing PID (Proportional, Differential and Derivative) control.

Complex recovery strategies are subdivided into three types, viz. the combination of both types of simple recovery types, back-up systems or in-built redundancy techniques, and state restoration techniques.

The first type, ie. the combination of the two simple recovery types is seldom considered complex, as this is literally all that it means.

3.1.1 Redundancy Techniques

Back-up systems and in-built redundancy are important where sub-systems of a plant are deemed critical to the correct and safe operation of the plant. All critical sub-systems should be provided with duplicated sub-systems, which can be operated when faults on the working sub-systems fail.

A concept of " protective redundancy " , [5], involves two types of redundancy, viz. :-

- masking
- dynamic

Masking protective redundancy hides or masks the effects of faults, ie. as long as redundancy is effective, faults remain totally invisible to the system, (hardware and software), or to the operator.

An example of masking protective redundancy is TMR (Triplicated Modular Redundancy). To correct a fault on this system,the faulty component is forced to read fault-free data and continue from there, (ie. state restoration , section 3.1.2). If errors are persistent, then the system fails to a simplex/duplex mode, with only the two healthy systems operational. This replication of systems is termed static masked protective redundancy.

A system may be configured as the above TMR system, but with spare modules to bring system back to original if any one fails, by isolating the faulty component and reconfiguring a spare. This is termed hybrid masked protective redundancy, and a number of systems have been developed using this technique, egs. SIFT, [15]; REBUS, [16]; PLURIBUS, [18].

Dynamic protective redundancy, [5], detects and identifies a fault, then take corrective action. In this, the faults are not concealed. Forward and backward recovery, (see section 3.1.2), are used to reconfigure hardware and/or plant items for state restoration, (see Appendix C.).

Automatic redundancy can be achieved with built-in software for decisions, eg. recover in stages, ie. full recovery, graceful degradation or safe shutdown.

Manually controlled redundancy would require an operator to decide on recovery, and perform the required reconfiguration manually.

3.1.2 State Restoration Techniques

State restoration techniques are used when a failure occurs and the plant state is only recoverable by either going backwards or forwards in the operating sequence to a known safe state. State restoration is defined as the restoration of the internal state of a component such that it is consistent with it's environment, [4]. This is particularly concerned with the internal states represented in the microprocessor control system, be it single, multiple or even distributed control systems. This will be considered in the context of the process control system, and not in terms of the microprocessor control system itself, (see Appendix C.).

Backward recovery means to restore the state of the control system to a previous known good state by modifying state and environment to be consistent, ie. set up control data and outputs to the plant such that it is the same as it was previously, and continue from there.

In sequential processes it is a simple matter to go back in a sequence and start from the beginning or any known safe intermediate point. This is done by storing the control data and outputs that occur at the steps of the sequence. Usually there are time delays involved in a process control sequence, and all steps backtracked must be reset so as to ensure a full backward recovery.

The difficulty in applying backward recovery in distributed systems is finding a consistent check point in all the different tasks in the system, and this causes what is known as the Domino effect, [5]. Each separate distributed controlling element will be able to store control data and the relevant outputs for the sequences that are being controlled, but these will very rarely coincide with the other controlling elements.

Industrial processes are generally real-time processes, and any controls that are done cannot be undone. This suggests that only forward recovery is permissable.

Forward recovery means finding a state consistent with the environment in the near future, [4], ie. set the control data and the outputs to the plant such that the state achieved is the same as it would have been if the controls had functioned correctly and the operating sequence had reached the new step "naturally".

In sequential processes, forward recovery is used to reset the sequence cycle to the beginning step, and then the sequence is continued while repairs are done.

To consider forward recovery in real-time processes, three classes of information are apparent, [4] :

Class I - state information enforced on the environment (outputs)

Class II - state information available from the environment (inputs)

Class III - neither Class I or II, (internal states)

Restart vectors are enforced on the environment to start in a known safe state, where a restart vector is the set of Class I and Class III information necessary to make the system look as if it is in the restart operation.

Class III information should be stored in redundant SRU's, (Smallest Replaceable Unit). The SRU is the single microprocessor board or memory module which is duplicated for redundancy and reconfiguration purposes. By storing the Class III information in SRU's, the Class III becomes part of Class II, as it can be recalled from the redundant store as new inputs.

Thus, forward recovery in real-time systems, and distributed real-time systems particularly, [4], involves retrieving Class II information (and Class III information), analysing this information and deciding on a restart vector. The restart vector must then be enforced and the control continues from this.

3.2 Fault-Tolerance

To make a system fault-tolerant, the system must try to detect erroneous behaviour and correct it dynamically, [4]. A brief outline of the steps involved in fault-tolerance realisation for distributed computer control systems is :-

(1) detect a fault has occurred
(2) diagnose the fault and locate Smallest Replaceable Unit, (SRU), or faulty device
(3) repair and/or reconfigure the system
(4) perform state recovery for the system
(5) repair the redundant units

Distributed computer control schemes, [4], lend themselves towards structured and modular fault-tolerance. Software can be designed by eliminating Class III information, (internal data, see section 3.1.2), by duplicating this information in redundant stores or

Author Horn Timothy Andrew
Name of thesis Fault Recovery In Process Control. 1985

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2013

LEGAL NOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed, or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page) for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.