

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/

Different check phases can cause different recovery procedures, and can depend on the phase or sequence step currently being performed, and the mode of operation, (eg. manual or automatic). A self-stabilising program is used to implement these check phases, [17]. Self-checking programs are ones in which any legal input produces the correct output or a message indicating that the output may be incorrect. These programs are also capable of recognising irregular conditions in their state space, ie. states of all devices, and reporting it. The self-stabilising concept is one in which recovery is implemented as well or instead of reporting the possibility of a fault.

4.3 System Implementation

Selecting a computer control system depends on a number of factors, and a number of traits in the selecting of control systems become apparent, [2], (turnkey systems in particular) :-

(1) Selection of a particular computer control system is based on hardware considerations, since most software functions, eg. PID control, speed ramping and logical functions, etc. , are available from many different vendors. Thus, a system is chosen if it has presumedly sufficient memory and input and output (I/O) capabilities, and any extra features.

(2) Selection of a system is biased towards systems proven in the field, especially if used in the particular process type being designed.

(3) The extent and availability of service and project management is considered, as is the software support for modifications during and after installation. The budget and the number of staff put in to solve problems during and after installation affect productivity and ensure the customer is satisfied with the service provided.

(4) Software would only be considered from the point of view of user-friendliness, ie. can the user change the software, and is the system easily understood.

Software complexity is repeatedly underestimated, especially when the operating principles are unique, and where the process is new to the computer system vendor. As software is " invisible " to a proposed user of a computer system, it becomes difficult to estimate software costs, ie. man-hours and facilities required to fully program and test the system software.

The cost and complexity of software is related to the proposed process control architecture used, ie. centralised versus distributed architecture, [39]. The cost and complexity of application or control and input/output (I/O) software both rise exponentially with increased utilisation of a central processing architecture, as shown in Figure 4.3 . For a distributed architecture, the application or control software costs and complexity falls exponentially with increased distribution, as shown in Figure 4.4 . The I/O software still increases exponentially with increased distribution because the I/O software becomes more complex as more processing elements are added to the bus or network.

Software costs are also dependent on whether the software is modular or not. Software development is expensive , and rises exponentially with complexity , [2], and quality and reliability are inversely proportional to complexity. Packaged systems in the form of standard software modules give consistent quality and good documentation.

To implement the level system defined in the hardware considerations, all sections of the whole plant should be automated to the same level, [3]. This assures that some sections are not left in manual control, while other sections are highly automated. The results of this are :-

- better operator training, as they will know how systems work at lower levels as higher levels are implemented.

- there is no need to interface and frig between automated and manual sections of the plant, especially interlocks required in the automated sections.

- the operating experience gained at the lower levels of the automating process allows sophistication of higher levels as the engineers and operators know what they want of the next level/s in the system. This allows faster realising of these aims, especially in the management levels.

In the same manner as fault trees are defined from the top event, (eg. general fault), the level system requires that the system be defined from level 9 downwards to level 1, [3]. As has just been shown, the system should be designed from level 1 upwards to level 9, and above.

The compatability of computer control equipment used at all levels must be ensured for effective communications and fault detection and recovery. This must be considered very carefully when different hardware suppliers are used.

Also, any errors and/or ommissions in the definition of the control and fault handling systems can cause these to be propogated into the final designed system. This will then create conditions in which no fault recovery strategy has been designed, ie. in both hardware and software.

Fault detection and diagnosis systems can be implemented using models, as discussed in section 2. The majority of the models, [1], can be combined into a complete model. For example, in the ISABELLE accelerator control system, [36], the system includes phenomenological modelling for both predictive and historical models. This was included for cases where the operators and systems engineers wanted to try a new operating procedure. The model would accept all required setpoints and parameters, and then foresee the outcome of these actions. This could cause the model to predict a dangerous operation, and forewarn the implementation of a dangerous set of conditions. Also, operational experience would refine the database of the predictive and historical modelling, and the model could also be refined in software. The predictive model would generally set certain conditions which would be considered fault producing or dangerous, and this can be modified to represent the fault tree verification of a system.

The implementation of a heirarchical DCCS for reliability and availability along the lines of the level system has been discussed. Maintaining the DCCS requires a means to repair hardware, process I/O, and software errors, while maintaining effective control of the process or plant. It also requires the ability to adapt under fault conditions, and adaption and functional enhancement, ie. expansion, to complete the fault-tolerance of the DCCS, [4].

Trends in industrial control systems have not been geared to fault-tolerance. Multiple redundant computer systems for reliability purposes have been strenously resisted by industrial application engineers. This has been due to several factors, [29] :-

(1) An instinctive dislike for "wasting" the computing capability of the second machine while it sits there idling waiting for the first to fail.

(2) The price of the second computer system in the past was an obvious point for cost saving even though an analogue backup system was even more expensive.

(3) A reluctance to severe completely from the analogue ' umbilical cord ' or familiarity and consequent trust of analogue systems.

Either total or partial recovery is implementable, [17].

Total recovery techniques can be employed when time is not critical, which is not true of real-time systems and general process control requirements. This is usually implemented when the requirement of a correct output is a first priority, and not successful backward recovery or state restoration.

The time for the software checking strategies to give a correct output is usually limiting in real-time processes, and so partial recovery is implemented. This is typically true when the computer systems are limited in recovery capabilities, or simple mechanisms are implemented, eg. complete shutdown if any fault occurs.

The prediction is that there will be an ever increasing trend to multiple redundant backup systems replacing the present analogue backup, [29]. Also, standby redundant machines will not be given secondary tasks to do while in standby states. This is because of :-

(1) The software complexity involved and the lack of trained personnel to carry out the work required.

(2) The fear of interaction between the primary and secondary tasks at the moment of switching and subsequent failure of the standby system to take over the primary task.

Another factor influencing the swing towards redundancy and standby systems for both plant equipment and computer hardware and software, especially critical systems, is the costs accru[illegible] if [illegible] is not done, [23].

The cost of repairs, switching or replacing, manual operation, maintenance, and the lost production must be considered, and the benefits in the long term far outway the original investment of standby equipment.

There is an increase of research into fault tolerance and automated fault analysis that should eventually eliminate much of the psychological dislike for the 'waste' of completely redundant backup systems as expressed above.

A general computer control system comprises two elements, [16], viz. a collection of dedicated instrumentation, PLC's and possibly up to mini-computers distributed in the plant, and a central control and management room with operators who monitor and control overall operations.

The reliability of any control system, and it's fault-tolerance depends on the reliability of the hardware, process I/O, software, and the operator, [4]. The operator is a vital factor in most control systems today. Human reliability, [22], in process control operations has been studied. It has been found that an operator is usually required to take flexible fault action when a manual control system is enforced, and their presence generally increases the reliability of the plant. In ideal conditions, human reliability is 1 in 1000 errors, and under extreme stress is 1 in 10 errors.

To make the operator function efficiently, the system must incorporate operator aids. These generally take the form of mimics of plant item statii, fascia panels with all identifiable faults and states, and with PLC's and micro-computer systems there can even be colour graphic video display units (VDU's), and printers for all state changes and alarm conditions to be reported, [8, 9, 10].

The allocation of functions between controllers and operators is roughly split between computer controllers doing the control and monitoring functions, while the operators do the emergency handling, ie. fault recovery, [26]. Consider under what conditions will the operator be able to undertake the right control action(s) at the right time(s) in order to prevent an undesired state. This becomes virtually impossible as the control task gets bigger and the number of faults that can occur increases.

The concept of human error is where the operator would pursue the wrong goal in trying to rectify the fault and acting incorrectly to achieve the required goal, [26]. Also, the time of intervention would in most cases be too late. The incapability of operators to perform satisfactorily under fault conditions thus requires automated fault detection, diagnosis and recovery.

The operator is an active member in the man-machine system, especially in large DCCS with a central control room with many operator CRT's. From this control facility, the operator must possess many capabilities, [34] :-

(1) The ability to start up and shut down the system

(2) The ability to control and manage the whole system to the objective and subjective constraints of production and management schedules.

(3) The ability to manipulate any emergency and failure or any combination of malfunctions in the system.

(4) The ability to assure that products are up to standards, ie. both quality and quantity.

Many computer control systems incorporate manual override facilities allowing the operator to assume control of parts and sometimes the whole of the system control. In the beginning of the computer control era, many controllers were controlled through manual override facilities, which undermines the effectiveness and necessity of implementing the computer controllers in the first place.

Thus it becomes necessary for companies to prohibit the use of manual facilities without consulting the systems engineers. An effective way of doing this is to have a lockable key-switch or a password entry scheme to place any parts of the control system into manual mode. Another more effective way is to train the operators to accept the computer control systems, [34]. This can be done with a well-designed operator training programme, which incorporates process operating details, the computer implementation and it's uses and abuses, and effective fault and emergency handling.

4.4 System Design Concept Evaluation

To make a computer control system fault-tolerant, the four basic functions must be performed, [21], ie. error or fault detection, fault diagnosis, fault recovery and isolation, and repair or restoration. Fault detection requires hardware and software detection mechanisms in real-time, with backup software tests at regular intervals.

The level system discussed in the hardware considerations lends itself to multi-processor and distributed computer control systems. The heirarchical structure of the level system can be developed to a modular, expandable, reliable and maintainable control system.

However, the level systems approach has some limitations, [3]. One is that the level system was originally developed with the characteristics of one class of process, viz. batch processes. This has been adapted to a more general fault recovery and control strategy as detailed above.

An important restriction is that more engineering, be it hardware design and evaluation, or software development, is generally required to make a system more fault-tolerant. This far exceeds what is necessary for successful control (only).

Fault recovery dealt with in the software considerations cover rollback functions, and backward and forward state restoration. Repair and restoration of the failed items is then done while the system is operational. The effect of these functions is to make the computer control system fault handling transparent to the operation of the process and plant.

The system implementation concerns the overall design concepts. An undesignable factor in the system is the operator, and this was discussed in section 4.3 in terms of reliability of function, the functions the operator can perform, and their role in the fault-recovery scheme in DCCS systems.

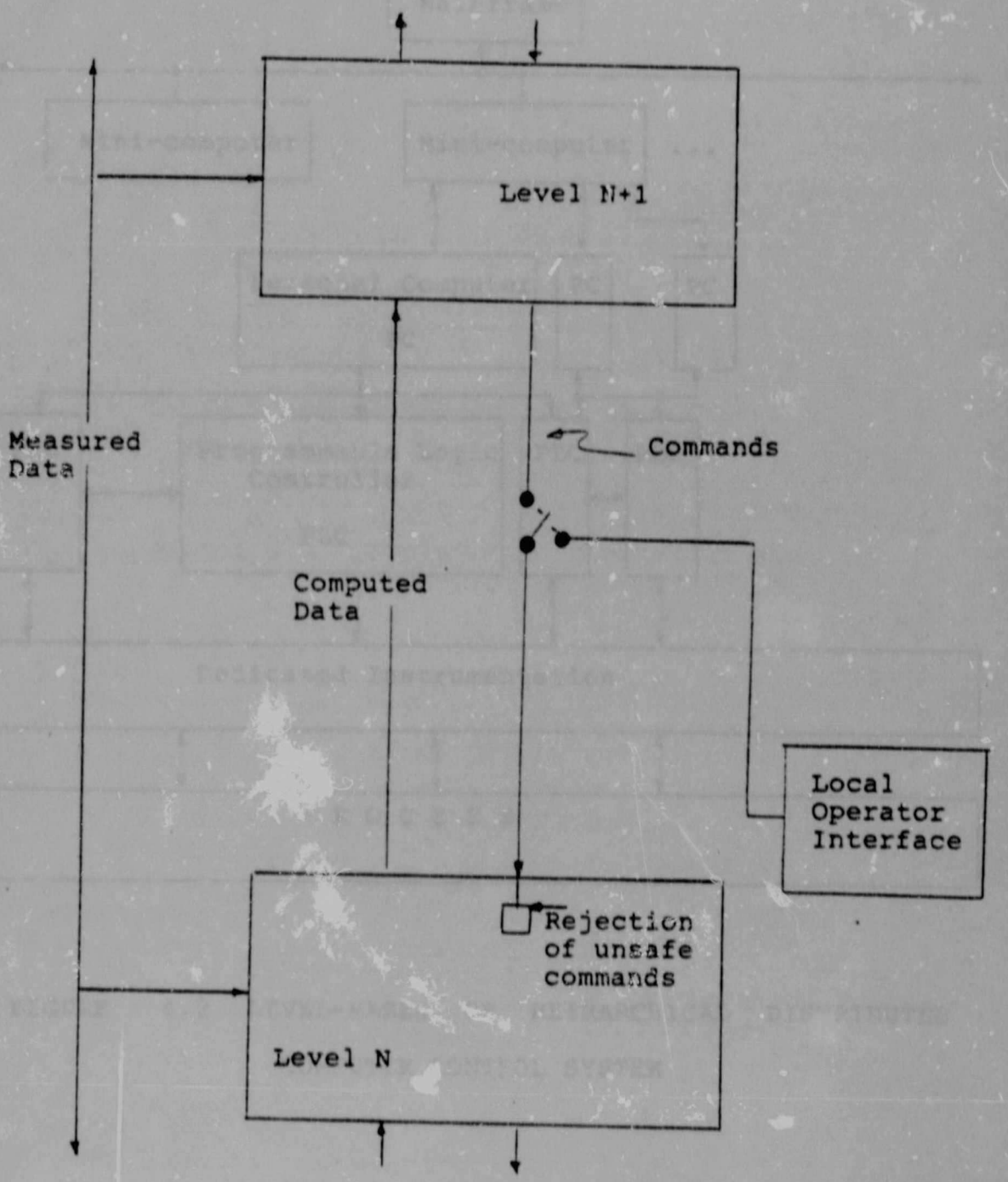


FIGURE 4.1 LEVELS OF CONTROL

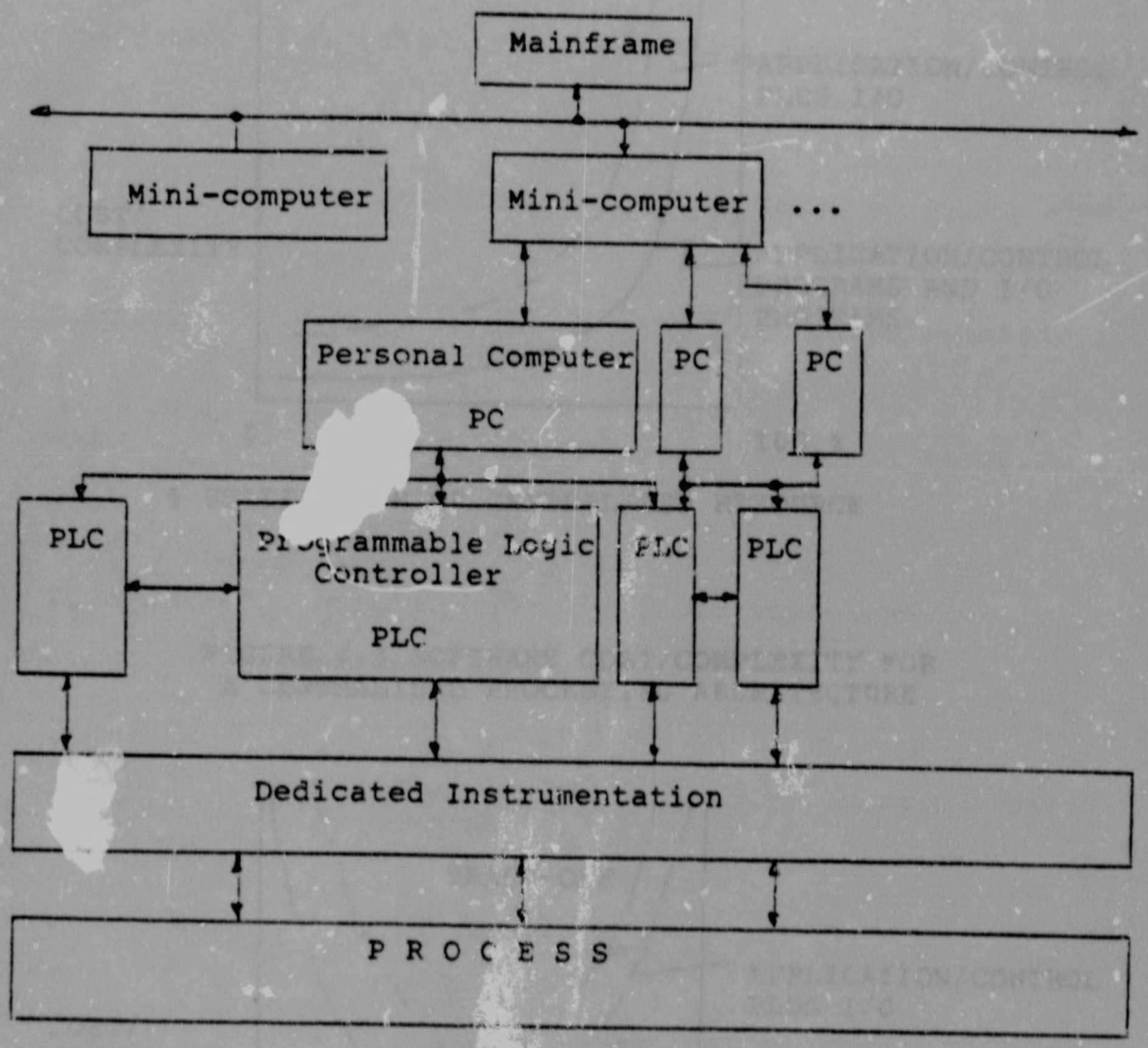


FIGURE 4.2 LEVEL-BASED OR HEIRARCHICAL DISTRIBUTED COMPUTER CONTROL SYSTEM

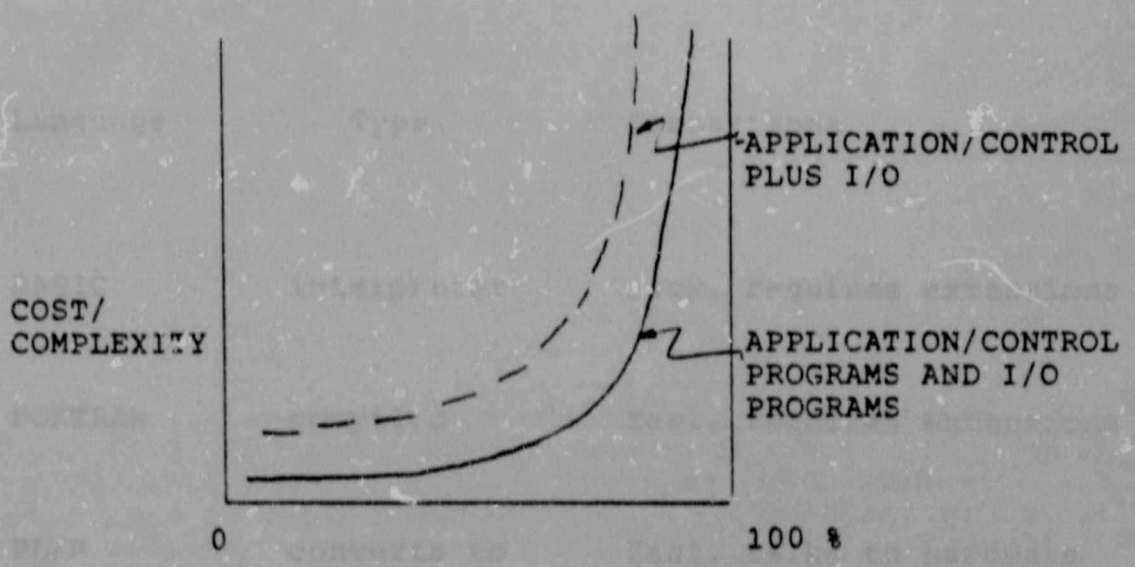


FIGURE 4.3 SOFTWARE COST/COMPLEXITY FOR
A CENTRALISED PROCESSING ARCHITECTURE

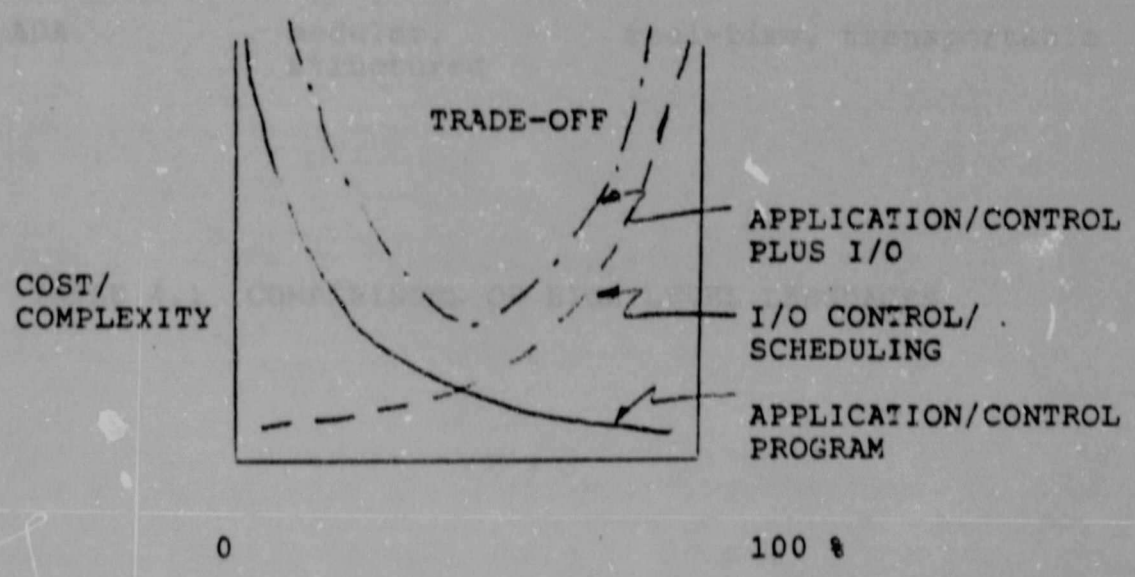


FIGURE 4.4 SOFTWARE COST/COMPLEXITY AND TRADE-OFF
FOR DISTRIBUTED PROCESSING ARCHITECTURE

Language	Type	Comparisons
BASIC	interpreter	slow, requires extensions
FORTRAN	compiled	fast, requires extensions
PL/M	converts to assembler	fast, talks to hardware and I/O
POLYFORTH	multi-tasking, compact code	fast, real-time, transportable
ADA	modular, structured	real-time, transportable

TABLE 4.1 COMPARISONS OF HIGH LEVEL LANGUAGES

5. CASE STUDIES.

5.0 Introduction

Two case studies will be examined in this section with the view to analysing the techniques used for fault recovery in the two control systems. The implementation of fault recovery techniques will be examined in three ways, viz :-

- fault detection

- fault recovery

- system implementation

5.1 CASE I - A Vacuum Pressure Impregnation Plant

The Vacuum Pressure Impregnation process, (VPI), is one which improves the quality and performance of large industrial machines, implemented at GEC Machines, Benoni, South Africa. This is done by insulating the large machine coil windings with glass-fibre cloth type insulation, and bonding with resin which is forced into the insulating cloth to remove all airgaps, bubbles, etc. to form a uniform insulation layer. This plant is currently being used for all sizes of motors from very low voltages up to 11 kV industrial machines. A simplified extract of the functional specification is given in Appendix A, and in [37] .

The stator and rotor to be impregnated is put into a vacuum pressure vessel called the impregnator. All air and residual vapours are withdrawn from the unit, ie. the stator or rotor being impregnated, and the vessel with a high vacuum and heating. Resin is then transferred to the vessel at the required temperature to ensure good transfusion into the glass-fibre insulating cloth. The vacuum and heating is held for a time to reduce the resin vapours due to heating and momentary catalytic reactions, and then the vessel is put under high pressure. This forces the resin into the glass-fibre insulating cloth, creating the uniform layer of insulation. The resin not forced into the insulation is transferred back to storage after cooling for reuse. The unit impregnated is then placed in a hot oven to cure the resin fully.

5.1.1 Fault Detection

General fault detection involved the digital and the analogue control and process parameters. All faults detected and/or the fault recovery performed were reported on the printer, and the alarm sounded. This was necessary so that any critical fault is automatically corrected by software, but reported for later identification, maintenance and repair.

As given in Appendix A, section A.3.1, the general fault messages for the digital I/O are of the following form :-

For pumps

(-item no-) NOT RUNNING
(-item no-) NOT STOPPED

For valves, safety and indicator limit switches

(-item no-) NOT OPEN
(-item no-) NOT CLOSED

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.