Citation: Kan, Chui-Chui Flora (1992). Modelling the behaviour of a process control operator under stress. (Unpublished Doctoral thesis, City University London)

Modelling The Behaviour Of A Process Control Operator Under Stress

A Thesis by Chui-Chui Flora Kan

For the Degree of Doctor of Philosophy

Submitted to

City University
Northamptom Square
London EC1V 0HB

Department of Electrical,
Electronic and Information
Engineering

March 1992

Declaration

The author grants power of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to her. This permission covers only single copies made for study purpose, subject to normal conditons of acknowledgement.

Table of Contents

List of Illustrations

Chapter 2

Chapter 4

Chapter 5

Chapter 7

Acknowledgements

Many thanks to AEA Technology, Winfrith Technology Centre and the Health and Safety Directorate for funding the research work described in this thesis.

Thanks to Ian Smith, Nick Wall, Julian Lockett and other members of the Dynamics and Control Group, AEA Winfrith for the support during the three years of my stay in Winfrith. In particular, I would like to thank Peter Townsely, formerly of the Dynamics and Control Group, AEA Winfrith for his work in constructing the interface to the control room simulator, and Frank Kingdon for this patience in explaining the simulator to me. Also I would like to thank Ian Hill and Robert Gale, formerly of the Dynamics and Control Group, AEA Winfrith for their support in the research work.

A special thanks to Professors John Senders and James Reason for their valuable time, and to my own supervisor, Professor Peter Roberts, for the support he gave generously throughout the past three years. Thank you to FRIL Systems Limited for their generosity, in the form of prompt technical assistance and the complimentary copy of FRIL, without which the work described here would have taken a lot more time.

Of all the people who have helped me, I am most indebted to Geoff Crowther, who has provided unfailingly during the past three years, the vital but missing link between civilisation and the wilderness - from Winfrith to the railway station to London. Due to his kindness, my sanity is preserved and this work has reached a conclusion of a sort.

Abstract

In this study, a framework to model the effects of stress on a process control operator is proposed. There exists many cognitive models, each of which attempts to model a specific class of human behaviour. One major effect of stress is the cause of errors, both physical and cognitive. In order to model the effects of stress, two cognitive models, a cognitive model of human errors and a cognitive model of process control operators are examined in detail.

In this thesis, the basic functions of the human cognitive system, its organization and a cognitive model of error commission are first examined. The behaviour of a process control operator and a cognitive model of the behaviour of the operator are then discussed.

The known effects of stress on the process control operator's behaviour are described and a framework for modelling the behaviour of process control operators under stress is proposed. The inadequacies associated with existing cognitive models for process control operators are explained and a modified cognitive model is proposed, which takes into account the cognitive model of error.

Finally, an architecture design for the implementation of the cognitive model is provided and suggestions for the next step forward are proposed.

List of Abbreviations and Symbols Used

Abbreviations

CC = Control Centre

HRA = Human Reliability Approach

HEP = Human Error Probability

IF = Importance Factor

KB = Knowledge-based Behaviour Mode

SF = Spatial Factor

SR-B = Skill/Rule-based Behaviour Mode

PSF = Probability Shaping Factors

VF = Visual Dominance Factor

WM = Working Memory

Symbols

P_i^n = Probability for indicator i to be registered by the operator at any given time, when the operator is operating in normal mode

P_i^a = Probability for indicator i to be registered by the operator at any given time, when the operator is operating in abnormal mode

P_i^r = Probability for indicator i to be registered by the operator at any given time, in random monitoring mode

$P_i^{d^1}$ = Probability for indicator i to be registered by the operator at any given time in directed subconscious monitoring mode

$P_i^{d^2}$ = Probability for indicator i to be registered by the operator at any given time in directed conscious monitoring mode

V_i = Visual dominance of indicator i

S_i = Spatial factor of indicator i

I_i^1 = Basic importance factor of indicator i

I_i^2 = Basic importance factor of indicator i

α = Probability that the monitoring performed by the operator is in random monitoring mode

β = Probability that the monitoring performed by the operator is in directed subconscious mode

Γ = Probability that the monitoring performed by the operator is in directed conscious mode

δ = Probability that the monitoring performed by the operator is in normal mode

1. Introduction

Human beings have become important components in many complex technical systems and in many cases the failure of such human components will lead to disastrous effects. The consequences of such human unreliability are well demonstrated and in accidents such as Three Mile Islands (1,5), Chernobyl (2) and the Zeebrugge ferry disasters. Therefore, unsurprisingly, the need to consider the human-system interaction has increased with the adoption of automatic systems in which the operator is assigned the role of a supervisor.

As human beings are animals of emotions, their behaviour can be very different when they are under stress. In a system in which a human operator is an important component, it is often desirable to know the behaviour of the operator under stressful operating conditions. Although component reliability and hence system reliability can be assessed with a fair degree of accuracy, human components were generally considered to be fault free and this is obviously not the case. When the human operator does make a mistake, the result is very often spectacular because these faults were not taken into consideration when the system was designed.

1.1 Review of Approaches to Human Reliability Assessment

Human Reliability is very difficult to quantify simply because the nature of the failure modes is not quantifiable. There are mainly two ways for human errors to occur. They are errors in judgement or "decision"" errors and the misrepresentation of intention or "action" errors. Various methods for assessing component reliability techniques have been applied to the human reliability assessment (3,4,49,50) with varying degrees of success (8). This approach is generally referred to as the engineering approach.

The other major approach is the cognitive approach. This approach stems from the belief that an accurate description of the behaviour of a human operator based on cognitive modelling is required to achieve a good understanding of human errors.

1.1.1 The Engineering Approach

In the engineering approach the human operator is treated as a component with limited capacities in a number of well-defined categories, such as attention span, response time and accuracy etc. Erroneous actions are attributed to the inherent variability of human performance, fluctuations in performance capacity, information overflows and stress. The major drawback of the engineering approach is that the internal cognitive functions which determine behaviour, such as reasoning and planning are not considered, although these functions play an important role in "decision" errors. The importance of such decision errors is best summed up in the famous Kemeny report on the Three Mile Island accident (5):

> "The operating staff made wrong decisions, intervened incorrectly and failed to perform the required operations, all of which caused a minor operating fault to become the TMI-2 accident".

1.1.2 The Cognitive Approach

Failure modes of the human component are not well behaved and therefore cannot always be foreseen. In order to capture the "unpredictability", some sort of cognitive model should be included in the assessment of the total system reliability. The reliability of a system in which human beings play an important part can then be studied by representing the operator with a computer program, constructed according to the known cognitive models developed.

The cognitive approach to human reliability is based on the explicit use of models or theories of the cognitive functions that constitute the substratum of human behaviour. It is believed that cognitive models can be used:

> "to produce the same data about human error rates that were sought by the engineering models but are in addition applicable for other purposes, such as predictions, dynamic simulation, sensitivity analyses, performance monitoring, design guidance, specification of error reduction strategies etc." (6).

See Note

The cognitive approach adopts a theory or model of the operator's cognitive functions. By its definition, a model is only a simplistic representation of the real system and cannot possibly include all parts of the system. The choice of the items to be modelled depends on the intended use of the model and even then the system can still be modelled from different viewpoints, with different emphasis. The validity of any cognitive model and its usefulness will be determined by the purpose of the modelling.

Engineers, due to the nature of their training and work, are more used to accepting models on "face values", whereas psychologists spent much of their training learning how to test the validity of what looks

Note: I must stress that engineering models have been successfully used in many system engineering for all the purposes described above.

superficially like convincing explanations of human behaviour (7). Therefore a satisfactory cognitive model in the eyes of an engineer may not necessarily satisfy a psychologist's criteria for a successful model. Various cognitive models already exist and the emphasis of different models is very often different. A Cognitive model which sought to explain human errors or absent mindedness mistakes was proposed by James Reason (10,15). This model is concerned with ordinary human mistakes. A cognitive model used to explain the behaviour of process control operators was proposed by J. Rasmussen(9). Although both models can be used to model human behaviour, they are essentially separate models and there is at present no single cognitive model which seeks to explain human errors within the context of process control operations.

Attempts at computer simulations of the behaviour of a process control operator by Cacciabue et al have produced mixed results (16,17). On the one hand, Cacciabue's group has been able to model the normal behaviour of the operator. The cognitive model used by Cacciabue et al is Reason's model of generic human errors, which is geared towards the simulation of general human errors. Reason's model did not explicitly model the three different modes of behaviour, Skill-based, Rule-based and Knowledge-based behaviours, particular to process control operators.

The limitation of the above model is its inability to model the cognitive errors observed in different modes of behaviour of a process control operator. Although the model used by Cacciabue can simulate some human errors, it is felt that the model's inability to model the three observed modes of process control operator behaviour casts doubts on the accuracies of the simulation's predictions. It is felt that a cognitive model which can account for the three modes of behaviour observed in the operators should be used and then the model should be extended to include the modelling of generic human errors.

1.1.3 The Effects of Stress on Human Reliability

There are many factors that can affect the reliability of an operator. One of these factors is stress. Since an erroneous plan of actions devised by the operator will affect the system most, the effects of stress on the cognitive processes of the operators are of particular interest.

The literature on the psychological effects of stress on the operators of a process control plant is mainly qualitative and attempts to produce quantitative results have not been wholly successful (3,8,49,50). The qualitative findings can only provide broad answers to the question of how stress affects the operator, but in themselves are not sufficient to produce situation specific recommendations.

1.2 Objectives and Approach of This Work

The main motivation behind this work is the need to produce a formal framework for the evaluation of the reliability of the system when the operator is affected by stress. A computer simulation, built to cognitive specifications can then be constructed and it can be used as a tool when interfaced with a plant simulator, to study the overall system reliability.

There are many problems associated with such an attempt to develop a simulation of the behaviour of the operator. First of all, a cognitive model that can account for the different modes of behaviour, including human errors, must be developed. Second there is a need to define the modes of interaction between stress and the operator's cognitive abilities. Third, the effects of stress on the cognitive processes of the operator must be identified. Fourth, the representation and modelling of errors, both action and cognitive errors and the effects of stress must be addressed and finally, a computer program architecture which will support the above modelling methodology needs to be developed. Here the dominant cognitive approach, which classifies the behaviour and actions of a process control plant operator into three categories: skill-based, rule-based and knowledge-based behaviour (18) is adopted.

This work aims to model the cognitive behaviour of the operator and no attempt was made to address the issues of scheduling or planning within the artificial intelligence context. The scheduling and planning methods used should be those described by the operators and the efficiency of these methods are not important for the purpose of this study.

In this study, the effects of stress on the behaviour of the process control operator were identified. A framework for modelling these effects and human errors, both Action and Cognitive was developed. The modelling methodology developed is relatively independent of the cognitive models used. A new cognitive model capable of modelling the three modes of operator behaviour is proposed and is later extended to enable human errors to be modelled. The model implicitly supports the modelling of the effects of stress and methods for this extension are described.

A computer architecture that supports the implementation of the new cognitive model is proposed. The architecture is based on the blackboard architecture (27,42) but differs from the architecture used by Cacciabue et al. The proposed architecture closely reflects the cognitive processing functions performed by the process control operator and this architecture also reflects the current understanding of the mechanisms behind various cognitive functions. The architecture proposed did not take into account the processing efficiency of the implemented model.

The rating of stress levels experienced by the operator is a problematic one since the feeling of stress is a subjective one. A psychologically based rating method (29) is identified to be useful in the assignment of stress levels to different plant states.

1.3 Overview of the Thesis

In chapter 2, the cognitive system is described and the nature of the operator behaviour is examined in detail.

In chapter 3, the major causes of human errors and their relations to the cognitive system as described by Reason and Mycielska (10) will be discussed. In chapter 4, the effects of stress on the operator and in particular his cognitive processing abilities will be described and examined closely. A framework for modelling the effects of stress and errors is also proposed in chapter 4.

This proposed modelling method is at this stage independent of the cognitive models used. The underlying assumption taken by the model is that the ultimate function of the cognitive system/operator is to make decisions. This assumption treats various cognitive activities eg. diagnosis and planning as decision making.

In chapter 5, an existing cognitive model and its implementation (16,17) is examined and the inadequacies of this model are discussed. A new cognitive model is proposed also in this chapter and its advantages over the other model are discussed.

In chapter 6, the modelling framework proposed in chapter 4 is applied to the new cognitive model. In chapter 7, a software architecture design, together with a software package which supports the implementation of the two proposed recognition methods is given and explained in detail. Finally, chapter 8 provides a conclusion to this work and suggestions for the next step forward are made.

In appendix A, a psychological rating method which can be used to obtain the stress levels associated with different plant operation states is described in detail. In appendix B, the code of the software package which allows the fast translation of the knowledge involved in recognition into computer code is listed. The manual of using this package is also included here for completeness.

2. The Nature of the Behaviour of A Process Control Operator

2.1 Introduction

Nowadays, the control of a process plant is highly automated. Under normal operating conditions, the role of the operator is mainly supervisory and at most will only carry out instructions as dictated in the operating manual. These required procedures are well rehearsed and practised. As a result, this type of routine action is almost automatic and requires little or no active thinking by the operator.

Under normal circumstances, the operator of any technical system will most likely perform the required actions as laid down in an instruction manual. Assuming that the manual is well written and correct, there may still be a finite chance that the operator will still commit an error. This may be due to the omission of a certain action as requested in the manual or an erroneous action was performed instead of the required one. These "Action" errors (as oppose to "Decision" errors mentioned in chapter 1) appear to be random in nature and yet no satisfactory way exists for the prediction of this type of error. In addition, the consequences of this type of error are dependent on the precise context in which they occurred.

In an abnormal operating situation, the main task of the operator is to change the system from an undesirable state to a desirable one. The operator will need to identify the current state and decide on a desired state to which the system should be changed. In order to achieve this desired state, the operator must devise a series of actions that will cause the system to reach the desired state.

Before this final state is reached, a series of intermediate states may be traversed on the way. The precise path of this transition may be crucial because the operator must ensure that no unsafe state is traversed on the path and that all operating constraints are observed.

It must be clear that many possibilities for error exist both in normal and abnormal operating conditions. In order to understand the nature and causes of these errors, an appreciation of how the cognitive system is organized and its basic functions is needed; in addition an understanding of the different types of errors that can be committed by human beings is required. This is particularly true for "Action" errors which have a common occurrence in daily life.

2.2 The Cognitive System

The human mind is a highly organized system. It can be separated into four sub-systems according to their relative functions (10)(Figure 2.1):

- Intention system
- Memory system
- Various Pandemoniums (abode of demons)
- Action system

2.2.1 Intention and Action Systems

The Intention system is where intentions are formed. It is not very clear how this is achieved but since this study is more concerned about what intentions are formed and how they are carried out, the Intention system will not be considered in detail here (10). The Action system facilitates the movement of the body and will carry out the actions intended (Figure 2.2).

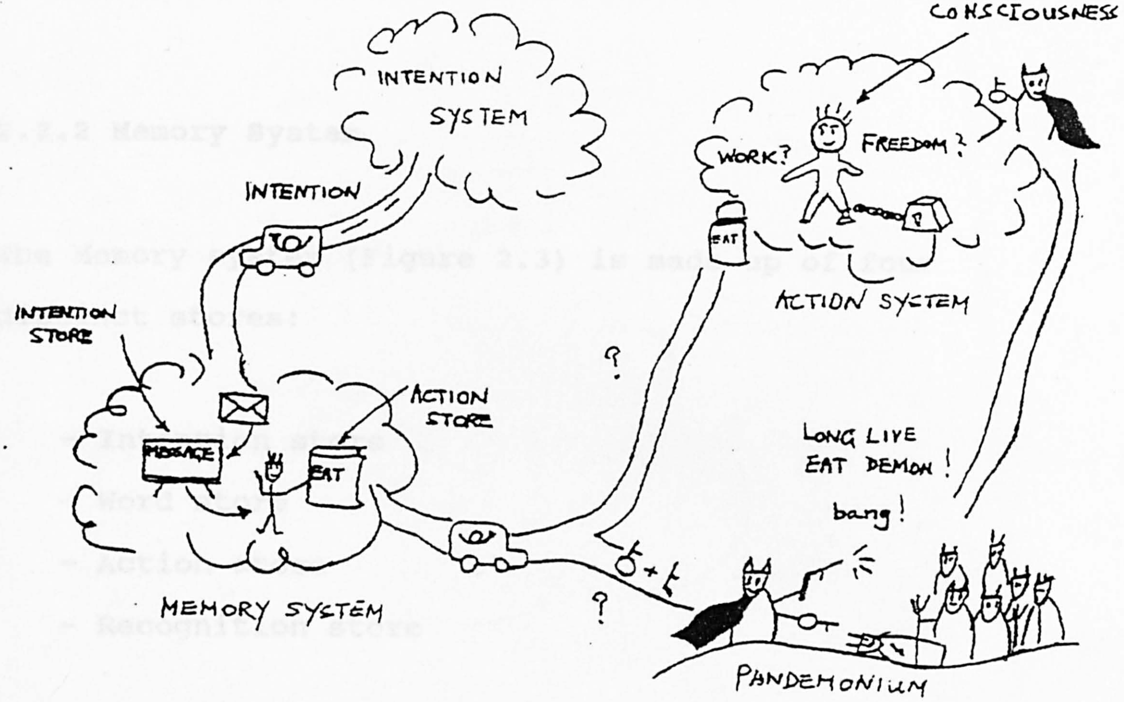


Figure 2.1 The Cognitive System

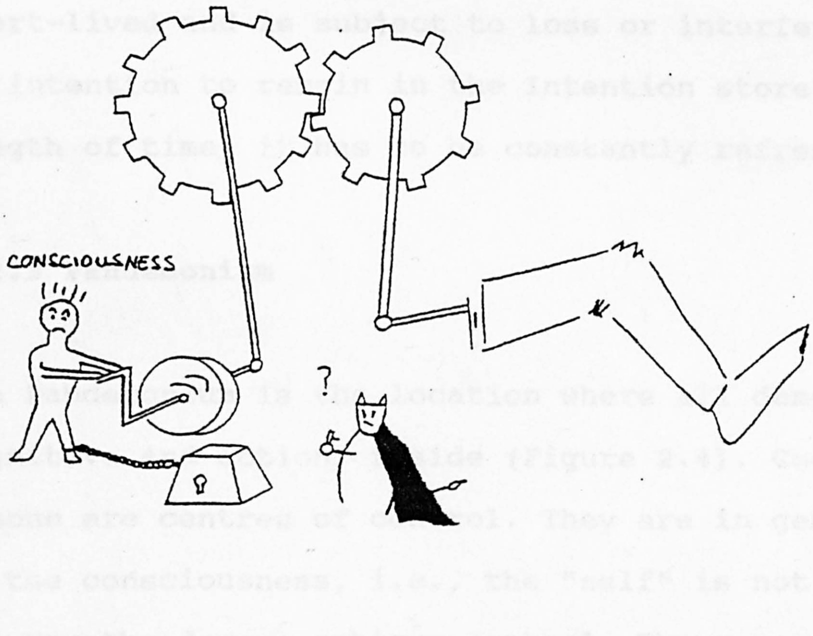


Figure 2.2 The Action System

2.2.2 Memory System

The Memory system (Figure 2.3) is made up of four distinct stores:

- Intention store
- Word store
- Action store
- Recognition store

The Intention store is a temporal extension of the Intention system. When an intention is produced in the Intention system, this intention is passed into the Intention store. The content of the Intention store is short-lived and is subject to loss or interference. For an intention to remain in the Intention store for any length of time, it has to be constantly refreshed.

2.2.3 Pandemonium

The Pandemonium is the location where all demons, both cognitive and action, reside (Figure 2.4). Cognitive demons are centres of control. They are in general opaque to the consciousness, i.e., the "self" is not aware of the way the demons achieve control. There are various types of cognitive demons, each controls a particular

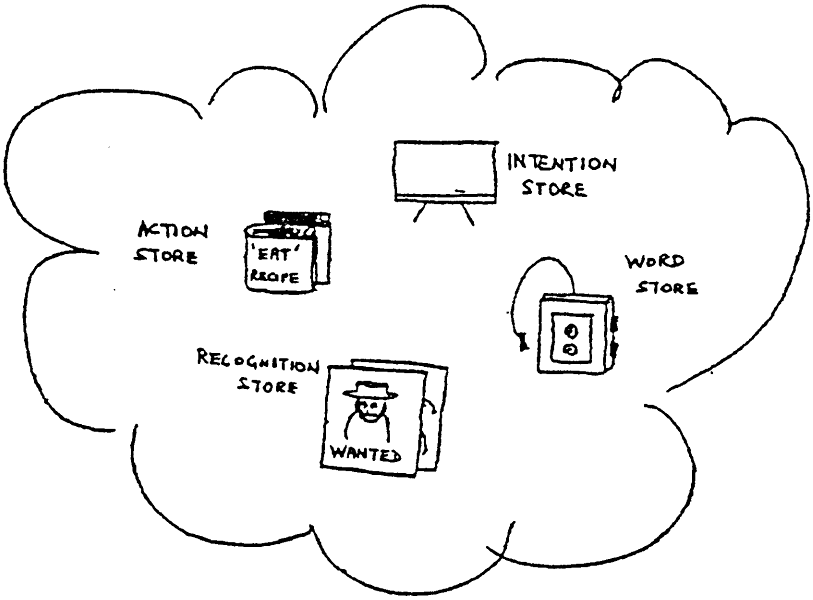


Figure 2.3 The Memory System

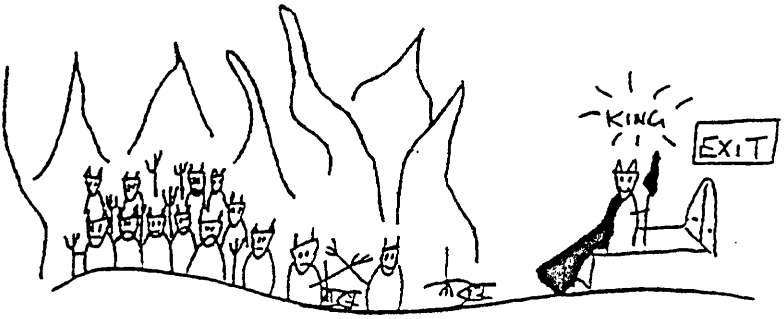


Figure 2.4 The Pandemonium

function. They can be grouped according to the functions for which they are responsible. These are Action demons, Recognition demons and Word demons etc. The Pandemonium in which all demons reside is an unruly place, where all the demons are competing with each other for control. In particular, within the Pandemonium there exists no "godfather" demon that can exert control over other demons. The Pandemonium is therefore a place with "lawless violence and uproar, which is constantly in a state of confusion" (10).

As their name suggests, Action demons control the effector mechanism of the body; Recognition demons are input specialists, each of which is tuned to respond to a particular feature of the sensory input and Word demons control the usage of words. The following examples will illustrate the importance of demons' role in the control of our everyday actions.

Learning new skills requires great concentration. It can be felt when one is learning to swim or playing snooker for the first time. Every move required has to be consciously performed. However, after practising these moves many times over, the concentration required in the performance of these actions will be lessened to the extent of being semi-automatic in nature. One extremely good example is walking.

When a child is learning to walk, he has to devote all his attention to the coordination of his limbs and body. After many tumbles and falls, he finally learns to walk and then he will no longer be conscious of the walking process. The required set of actions for walking will be stored and a "walk" demon is born. Once a demon is constructed, the shrieks from the Intention system messenger - "I want to walk" - will activate the "walk" demon and it will then assume control of the body (Figure 2.2).

The benefits of this type of devolution of labour are obvious. If we have to devote all our concentration every time we want to walk, nothing substantial will ever be achieved. We will not be able to notice the dresses displayed in the shop windows while walking down Sloane Street to the underground station or talk while we walk. One cannot fail to appreciate how utterly boring such an existence would be. Demons can thus be regarded as dedicated machines and once they are activated they will happily continue to perform their tasks without any supervision. However, it must be also clear that this type of devolution of labour has its drawback because the less the consciousness participates in the effecting of actions, the less detail control it has over the outcome of the actions. Demons can be misfired and "Action" errors are mostly caused by the misfiring of demons.

2.2.4 Activation of Demons

It is convenient to regard cognitive demons as being made up of TOTE (Test-Operate-Test-Exit) (10), nesting one within another like a Russian doll. It is therefore clear that there are two types of demons

- Testing and
- Operating

Action and Word demons are activated by shrieks of the message from the Intention system and Recognition demons receive their triggers from the sensory input. Each demon is activated by a set of conditions which may be external or internal. It is possible for a demon to misfire and this is most likely when the schema of the on-going activities resembles closely another schema, into which the misfired demon is incorporated. The relationships between the activations of demons and errors are discussed further in chapter 3. The cognitive system described above is basically true for all cognitive systems. Very specialized mental functions contain extra characteristics themselves, which are specific to the cognitive functions required. The special cognitive characteristics of a process control operator is described below.

2.3 Different Modes of Process Control Operator Behaviour

In highly automated process control plants, the major task of the operator is monitoring. If the operator detects an abnormal situation, he will act according to the training he received. The operator has to perform three separate functions, namely perception, thought and intervention.

Perception:	The reception and interpretation of data as supplied by the visual and acoustic indicating devices of the system, which provide information about the state of the system.
Thought :	The evaluation of the state of the system and the determination of the operation to be performed.

Intervention: The operation of control elements (switches, handles, etc.). The operator generally receives a signal of the result of the intervention insome form or other.

When the plant is operating normally, little or no intervention is required from the operator, i.e., the operator thought, from the signals he received from the plant, that the plant is in a desirable state and decided that no intervention is required.

During an abnormal situation, a typical task of the operator is described below (Figure 2.5) (12):

At time T1, the system is in a desired state S1 and plant condition C1. At time T1, an unexpected event E1, such as a valve failure took place. E1 causes the system to change to state S2 and plant condition C2, which may be undesirable. The operator will have to identify the plant state at T2 based on the data, i.e., C2, he received in the control room. He will have to diagnose the cause of this change of state and make a prognosis according to his diagnosis. The operator will need to decide on a course of action which will shift the system from the undesirable state S2 to S3 (which may be identical to S1).

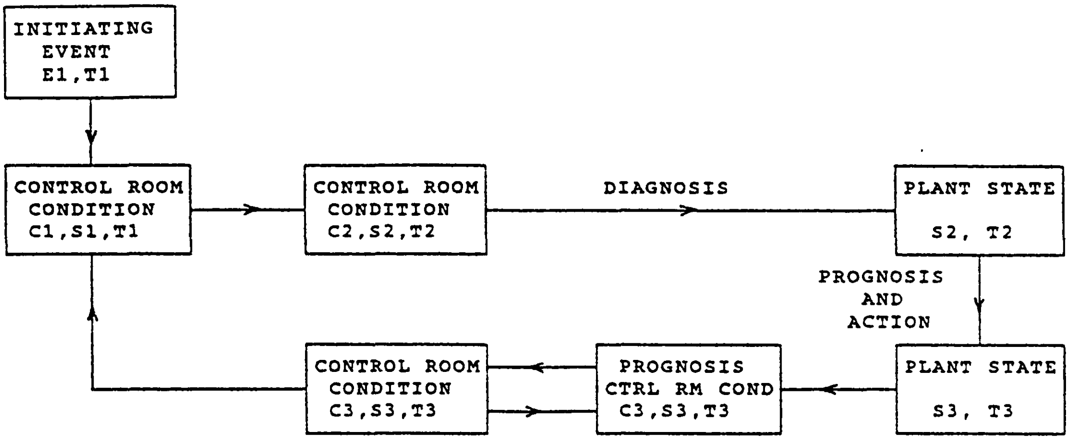


Figure 2.5 Steps Required During Operation Under Abnormal Plant Condition

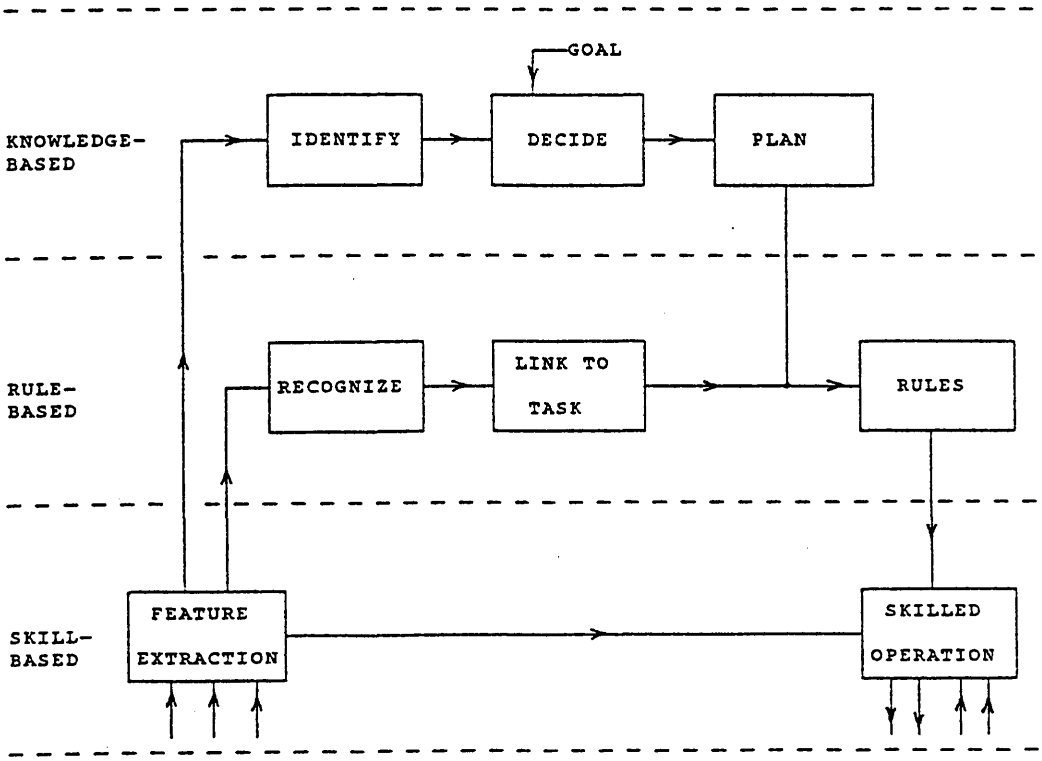


Figure 2.6 The Three Categories of Operator Activity

In most cases, the system state is recognized and the relevant tasks to be performed as laid down in the operating manual can be followed. The required course of action is nevertheless expected and as such no elaborate thinking is required of the operator. Under some very unexpected situations, the state of the system may not be immediately recognizable, nor may it be detailed in the operating manual. In order to decide on a goal (state S3), the operator will have to rely on his own judgement in order to select a course of actions, formulate a plan according to the safety and operational requirements, and carry out his plan. The operator's judgement will be based on his knowledge of the design, the operation of the plant and physics.

Rasmussen classified the behaviour and actions of a process control plant operator into three categories: Skill-based, Rule-based and Knowledge-based levels of behaviour (9). Evidence to support the existence of these separate categories has been confirmed in the observations of the operator in action (11,12) (Figure 2.6) (9).

2.3.1 Skill-based Behaviour

Skill-based behaviour refers to behaviour which involves well-learnt and mostly automatic reactions. The executions of these reactions are automatic and do not require attention. These skill-based actions involve mainly one step actions which constitute the basic building blocks of human activity. As can be readily appreciated, skill-based activities are closely associated with demons (section 2.2.3 and 2.2.4) and errors within this category of actions share the same mechanism with the errors associated with the activation of demons (chapter 3).

2.3.2 Rule-based Behaviour

During training, the operator practises dealing with the more likely faults. Provisions related to these possible faults are also included in the operating instructions for dealing with likely operating faults. Hence, when such a fault occurs, the actions required from the operator are as follows:

- Recognition of the operating fault based on the plant parameters.

- Assignment of the task to be performed in the known way.

- Execution in accordance with the instruction.

Rule-based behaviour refers to behaviour where the operator knows that he has the information on what to do, either from memory or from some manuals. The actions required in this mode involve mostly multiple steps actions. These are generally linked together by some testing, such as if A is true then do B. Rule-based behaviour requires attention during the execution of the actions but attention may not be required continuously because rule-based actions may themselves be made up of many one step skill-based actions and so no attention will be required to perform them.

2.3.3 Knowledge-based Behaviour

Knowledge-based behaviour refers to the situation when the operator does not know what to do or when he cannot relate to the situation. He will then need to use his general knowledge and common sense to try to devise a solution to the problem and to construct a plan and strategy. This mode of behaviour requires attention constantly because the operator has to be on constant alert in order to monitor the behaviour of the system.

Knowledge-based behaviour occurs in particular when the operator fails to identify the state of the system and therefore he cannot predict the subsequent behaviour of the plant. As a result, he will have to be particularly vigilant over the monitoring of the plant signals and he has to make sure that no unexpected change in plant parameters escapes his notice. This sort of extra demand for sustained attention from the operator is sufficient to produce a stress response (13).

2.4 Summary

In this chapter, a generic cognitive model which can account for different human errors was examined. In addition, a specific cognitive model describing the nature of the behaviour of process control operators was examined. In the next chapter, major causes of human errors will be described in detail and their relations to the cognitive system will be discussed.

3. Causes of Errors

3.1 Introduction

The main differences between human activities and machine operation are the greater variability and seeming unpredictability associated with human actions. If the operator has to perform a certain operation many times over, the characteristics associated with the execution of the operation, such as the speed of execution or strength, will vary greatly as compared to an equivalent machine operation.

In general, an error occurs when a planned action fails to achieve its desired outcome, and when such a failure cannot be attributed to the intervention of some chance occurrence. Errors can be committed at two different levels: the perception-intervention level (Action errors) and mental errors (Decision errors) (Figure 2.6).

Action errors rarely have serious consequences in a modern nuclear power plant. Most of the time, these errors will be corrected by the operator themselves or at worst, the protective system will be activated and the safety of the plant is thus ensured. Decision errors however are more important and may have more significant consequences.

The forms of Action and Decision errors can take are closely linked with the organization and functioning of the cognitive systems. The relationships between errors and the organization of the cognitive system will be discussed in section 3.3.

3.2 Factors Affecting Operator Performance

There are many factors which can affect the performance of the operators and they can be broadly separated into two categories, external factors and internal factors. In addition, there are the effects of stress. The performance of the operator can be affected in two ways, in terms of susceptibility to Decision errors and Action errors

The external factors are mainly concerned with the design of the plant, its management and ergonomic aspects. The internal factors are mainly concerned with the individual characteristics of the operating personnel, such as operator qualification, experience and intelligence, etc. Stress generally occurs when the external and internal factors are in conflict, e.g., if the work imposes too heavy a burden on the operator.

For an existing plant, there is little one can change in the area of design once the plant has been built. The same arguments can be applied to the ergonomic aspect of plant control. Of the many external factors, the management and the underlying working philosophy of plant operation may be the least difficult to change; though this does not mean that no resistance, union or otherwise will be encountered.

The internal factors can be dealt with with less difficulties, such as by requiring prospective operators to undergo psychological and IQ tests. The intelligence and personality of the prospective operator can be evaluated and therefore operators with all or nearly all the characteristics required of a nuclear power plant operator can be selected, that is assuming that such a profile can be defined in the first place.

The influence of stress on human behaviour has been a topic for inquiry for many years. Interest in this area is fuelled largely by two factors. It is believed that a better and profound understanding of human abilities may be gained from the study of individuals' reactions to extreme conditions and the second reason is a practical one. It is increasingly important for many different utilities, notably the nuclear electric industry, to understand the reactions of their personnel under stressful operating conditions.

A detailed examination of the effects of stress will be taken in chapter 4.

3.2.1 External Factors

There are many external factors which can affect the performance of the operator and a list of these factors is listed in table 3.1 (26). The conditions of the working environment will affect the comfort of the operators which may in turn affect operator performance. A detail examination of these factors is beyond the scope of this study but in general, the management of the operation of the plant, the management of the operating personnel, design emphasis of the plant and the ergonomic design of the controls will all affect the performance of the operator. A brief discussion of the effects of the environmental factors will be given below as an example since environmental discomfort can very often cause stress.

External Factors Affecting Operator Performance:

- Building Construction — What's this?
- Quality of Environment:
  - Temperature
  - Humidity
  - Air purity
  - Noise
  - Vibration etc.
- Management Factors:
  - Shift pattern
  - Responsibility assignment
  - Bonuses
  - Recognition
  - Benefits
- Working Philosophy:
  - Degree of reliance on verbal and written instructions
  - Communication between team members
  - Preventive measures
  - Operation guide-lines
  - Established practice
- Design Concepts:
  - Perception requirements
  - Motor requirements (speed, power, accuracy etc)
  - Relationship between control elements and indicators
  - Interpretation requirements
  - Information load
  - Frequency of repetition of tasks
  - Requirements on transient or long span memory
  - Criticality of tasks
  - Calculation requirements
  - Indications of the results of operations
  - Team structure requirements
- Ergonomic Considerations:
  - Design of basic equipments
  - Positioning of indicators etc. on the control panel
  - Man-machine relationship

Table 3.1 External Factors Affecting Operator Performance

Internal Factors Affecting Operator Performance:

Individual Characteristics:
- Qualifications
- Experience acquired in previous job
- Experience acquired in present job
- Personality
- Intelligence
- Motivation and attitude
- Assumed knowledge of instructions and specifications
- Physical conditions (mental and physical fitness)
- Outside influence (e.g., family or political persuasion)

Table 3.2 Internal Factors Affecting Operator Performance

Causes of Stress:

Psychological:
- Suddenness of event
- Time constraint
- Speed of execution of task
- Information load
- Consequences of failure of task
- Fear (of failure, of loss of job, recrimination)
- Monotony of work (boredom)
- Length of long and uneventful duty period
- Unstimulating environment
- Conflicts of work and personal belief
- Distraction (noise, movement etc)

Physiological:
- Long period of psychological stress
- Tiredness
- Pain or discomfort
- Hunger or thirst
- Temperature
- Freedom of movement
- Degree of physical exercise

Table 3.3 Causes of Stress (Szabo) Action errors.

3.2.1.1 Example - Sound

The operators are required to be vigilant and studies in noise effects have produced evidence that vigilant performance is indeed affected by noise level, though no systematic pattern emerged (13). In general, performance is degraded by a high level of white noise (above 90 db) when processing demands are high. Performance level remains unchanged when processing demands are low whether the white noise level is above or below 90 db. It also appeared that performance during low task demand is enhanced by the presence of low-level varied noise.

Generally, a low level of sound is considered a low stress condition and presents little adverse effect on operator performance. Therefore, noise can be considered unidirectional in effect.

3.2.1.2 Example - Temperature

The other usual cause of discomfort is changes in temperature. The effects are bidirectional because both high and low temperature are stressful. Research has shown that an increase in body temperature speeds the apparent duration perceived by the operator, whereas a decrease in body temperature slows it down (13). It has been suggested that this change in duration perception by the operator is one of the fundamental causes of performance variation on a number of tasks that require sustained attention.

3.2.2 Internal Factors

Operating a process control plant requires certain characteristics. The personality of the operator may affect his performance and this is particularly true during a crisis situation. A list of some of the internal factors which can affect operator performance can be found in table 3.2. (26). A brief discussion about qualification of operators and experience is given below.

3.2.2.1 Example - Qualifications

Qualification will have some bearing on the performance of the operator. When the plant is operating normally, the major task of the operator is monitoring. In the case of an operating fault, however, the highest qualifications are essential to good performance. If

> "the operator is highly qualified, the routine monitoring tasks will not be demanding enough for him. Consequently the operator may become bored and will not be well motivated. Boredom is also a frequent contributor to stress. However, if the operator is not very well qualified, he

> may not have the knowledge nor skill to cope with an operating fault; particularly if the recovery of the plant requires independent thinking." (14).

3.2.2.2 Example - Experience

It has been suggested (4) that experience will increase the operator's resilience to stress. An experienced operator will possess some empirical knowledge which may not be documented. Experienced operators may also have intuitions which cannot be easily explained. They may also be able to judge the relative success likelihood of different alternatives more accurately. Extra experience enables the operator to feel that he is in control, which serves to reduce the degree of stress felt by the operator.

3.2.3 Stress

Stress arises because an individual feels unable to cope with the demand of the situation and is mainly dependent on the perception of the individual in question. Although stress does produce physiological symptoms in the individual and these are relatively difficult to fake, these do not represent the true extent of stress the subject is experiencing. A better measure of the level of stress experienced by an individual is the subjective feeling of the individual in a given situation.

Stress is often viewed as one of the fundamental reasons of performance degradation. However, when a person describes himself as stressed, he is merely describing the physical and mental state he feels he is in and as such does not provide any indication to the cause of this state. Stress can be caused by many factors and a list of these factors can be found in table 3.3 (26).

It must be emphasised that there are many factors which can cause stress and these can be external, e.g., temperature and humidity, or internal, e.g., fear. Stress is closely coupled with the external and internal factors discussed in 3.2.1 and 3.2.2. It is not yet clear how these factors affect the subjective stress level of an individual because the precise interactions between different stressors are not understood. The interactions between the stressors could be additive, synergistic or antagonistic. Sleep loss and noise, for example, have been found to have antagonistic effects (13).

3.3 Mechanisms of Error Commission

The difficulties associated with the study of errors are best described by Reason and Mycielska (10):

> "Error, ... is not an easy notion to pin down. If we look for its meaning in the dictionary, we are sent on a semantic circular tour through other terms such as mistake, fault, defect, and back to error again. The fact that dictionaries yield synonyms rather than a definition suggests that the notion of error is something fundamental and irreducible."

Reason (15) classified errors into three categories, Slips, Lapses and Mistakes and the definition for error as used by Reason will again be adopted here . A slip is having the correct intention but the actions carried out were wrong. A lapse is when an intention is forgotten and a mistake is when the intention itself is wrong. Reason associated slips with absent-mindedness (10) and are closely related to the activation of demons. The central argument being:

> "the belief that systematic forms of human error have their origins in fundamentally useful processes. However, the centralised operation of high risk, complex, and incompletely understood process systems can, on occasion, transform these normally adaptive human processes into dangerous liabilities."

3.3.1 Action Errors and the Activation of Demons

As mentioned in sections 2.2.3 and 2.2.4, demons are dedicated machines and are in general specialists. Once a demon is activated, it will happily continue to perform its designated task without further supervision from the "consciousness". Since "consciousness" participates little in the action of the demon, it will have little direct control over actions performed by the demon. Demons can be misfired and the misfiring of demons are one of the major causes of " Action" errors.

Action errors are closely coupled with slips. Slips occur during largely automatic execution of well established routine sequences of actions in which demands upon continuous attention for control are small. Slips are associated with distraction or preoccupation. They are most likely to occur when limited attentional resources are allocated to some external or internal matters which are totally unrelated to the on-going environments, where there are few departures from the expected and, therefore, require little outward vigilance. There are indications that the liability to minor cognitive failures has some relationship to an individual's general vulnerability to stress (14)

3.3.1.1 Misfiring of Demons

The role played by the misfire of demons in error is best illustrated by an example.

A commuter travels to waterloo station every morning. He then walks to the bus stop in order to wait for bus 171 to the City University. He has trained himself to look out for the bus while crossing the bridge. If he sees bus 171 approaching, he will run across the bridge so that he will not miss the bus. One Saturday morning, he had arranged to meet his friend at Charing Cross station and decided to walk there.

He arrived at Waterloo station and while he was crossing the same bridge he crosses everyday, he noticed that bus 171 was approaching. He only realised his mistake when the bus was pulling out of the bus stop with him in it. The similarity between the on-going situation and the schema of going to work is the basic cause of this substitution of actions.

Schemata are organised memory units. They are active organisations of past reactions to past experiences. Schemata are always present in any well-adapted organic response. Determination by schemata is the most

definition!

fundamental of all the ways in which human beings can be influenced by reactions and experiences. Schemata control is continuously switching between the Intention store and Action store. The sequence of control is as follows:

When the commuter arrives at Waterloo station, the Intention system is in control. He "knows" that he is going to meet his friend. Then the control is passed to the "walk" demon and the "intention" to meet his friend has gone into hibernation. However, on crossing the bridge, the on-going situation resembled the schema of "going to City University", and when the bus arrived, the schema is completed and this prompted the "run" demon to fire. While he was crossing the bridge, the control was hijacked from the "walk" demon by the "going to City University" demon. Once the man was sitting in the bus, the control was returned to the Intention system and then he realised his mistake.

3.3.1.2 Attention Capture

As described in section 2.2.1, intentions need to be constantly refreshed in the Intention store. However, when some external events, which cause other intentions to be triggered occur, the attention of the person can be captured. The intentions triggered by these external events can sometimes take precedence over the original intention, which should then be put into hibernation. At the completion of the tasks required by the overriding intention, the original intention should then emerge from hibernation. This system of first in last out Intention store is analogous to a stack machine. However, in this stack storage system, the first-in intention can be lost because of the finite life span of the intentions stored in the Intention store.

Although the control is eventually returned to the Intention store, the original intention may be lost. Under these circumstances, the demon in the schema of which the on-going situation resembles most will take control. Most errors associated with the forgetting of intentions are of this type.

The memory of the Intention store is refreshed constantly. In framing an intention to do anything, a state of tension just sufficient to remind us of the stored intention is created. The more important that intention is, the greater the stress generated. As a result, though performed, some residual feeling may remain and may cause an important intention to fire again.

3.3.1.3 Over Attention

This type of error is most readily appreciated by the highly skilled professionals such as top class musicians and snooker players. This syndrome is also known as the "nosey adviser syndrome". Basically, over attention can cause some well practised actions to falter because control of this type of action is mostly controlled by the respective Action demons. Demons are created in order to take over control of certain actions and as such demons can be considered to be a finely tuned but dedicated machine. This situation is analogous to the relationship between the apprentice and the supervisor. For example, having learnt how to solder from the supervisor, the apprentice had been delegated the sole job of soldering. Since the apprentice has more practice at soldering than the supervisor, who may be more knowledgeable at the general theory and techniques of soldering, the apprentice becomes more proficient at the actual soldering than the supervisor. Over attention is equivalent to the supervisor suddenly deciding, after twenty years of not having done any soldering, to do his own work. It can be guarantied that his performance will be inferior to that of the apprentice.

3.3.1.4 Program Counter Failures

For the successful completion of a plan, many largely automatic actions have to be carried out sequentially. Each plan of action will involve a unique sequence of actions to be performed and therefore an automatic device, analogous to a counter, is needed to keep track of the action sequence.

This cognitive program counter can fail in its operation mainly by miscounting actions. This can happen when the counter miscounts a thought as an action and thus proceeds to the next required action in line. Program counter failures are generally the cause of errors of omissions and lapses.

3.3.1.5 Anticipating Leaps

When intentions are framed, a state of tension is created with it. This tension serves as a reminder to the Intention store. If this intention involves a primal action demon which has a high level of tension associated with it, this primal demon may jump the queue of demons. Tension is not a pleasant feeling and the person will want to remove this tension as soon as possible.

Basically, the higher the tension associated with the intention, the more the person needs to remove it. As a result, the on-going state of need for the person is the removal of the tension, which implies the completion of that intention.

3.3.1.6 Habit Intrusion

Man is an animal of habit. Habit is a mixed blessing indeed and it can divert our actions, but most important of all, it can in some cases impose restriction upon our thinking. Habits of action are easily detectable but habits of thought are not. As a result, habits of thought can be self perpetuating and seem to emerge when they are most unwanted. In general, the stronger the habit, the easier it is for it to emerge.

The causes of habit intrusion are manifold. Habit intrusion is most likely to occur when there are strong established linkages between demons within the action store. They can also happen as a result of external circumstances prevailing at the time.

Errors due to habit intrusion are characterised by inertia, rigidity or reservation. They are most likely to occur when a change of goal necessitates a departure from routine or a change of circumstance demands modification of a preestablished action pattern. If habit intrusion occurs in thought then decision errors will result.

3.3.2 Decision Errors and Cognitive Demons

False impression or hypothesis is especially likely to occur and be accepted during a period of high stress and high anxiety (10). It is also likely to occur immediately after a successful recovery from an emergency because there is a natural tendency to relax one's vigilance. The successful management of an emergency creates a euphoric feeling which can last much longer than it actually warrants. When a person is trapped in a false sense of euphoria, his guard will be lowered and under these circumstances, a false hypothesis which fits in with his needs and expectations has a strong chance of being accepted. Once a hypothesis is accepted, the Intention system shows a marked resistance to abandon it. This sentiment is echoed by the great theoretical physicist Einstein:

> "If facts don't fit the theory, then the facts are wrong."

There are various factors which influence the occurrence and acceptance of wrong impression or hypothesis (10).

3.3.2.1 Frequency

It is found that the more frequent a cognitive demon is called, the easier this demon will spring into action of its own accord even when the situation is entirely inappropriate.

3.3.2.2 Incongruity

Four different types of reactions are possible when a person is faced with absurdity and inconsistencies. These are dominance, compromise, disruption and recognition. In the dominance mode, he will deny the incongruous elements; in the compromise mode, he will attempt to resolve the conflict; in disruption mode, he will fail to resolve the conflict and will respond with bizarre actions and, finally, in the recognition mode, he will realise that a false hypothesis is used and this is normally preceded by a vague sense of wrongness.

3.3.2.3 Context

The human mind processes information mainly in two ways, namely top-down and bottom-up. In the top-down mode, higher level knowledge determines the way inputs from specific feature demons are perceived. In the bottom-up mode, minor demons identify specific features at the lowest level of processing which are then combined by higher level demons into patterns. With the aid of context, less information is needed to hypothesize. Similarity in context (schemata reasoning) could cause inappropriate cognitive demons to fire and the creation of a wrong hypothesis.

3.3.2.4 Need

Human sometimes perceive that they want to perceive. For example, when an abstract picture is shown to a person who has been deprived of food, he is very likely to perceive the image on the picture to an image of food.

3.4 Summary

In this chapter, the major causes of human errors were examined and their relationships to various cognitive models were discussed. In the next chapter, the effects of stress on the operator and in particular, his cognitive processing abilities will be described. Also, in the next chapter, a framework for modelling the effects of stress and errors will be proposed.

4. Framework for Modelling the Effects of Stress

4.1 Introduction

Stress has two major effects on a process control operator. First it can cause an increase in Action errors and second the cognitive processing abilities of the operator will also be impaired. This impairment will in turn cause Decision errors to occur. Of the two types of errors, Decision errors are of more significance. The findings of stress researches are mainly qualitative in nature but the researches seem to be unanimous on the effects of stress on the cognitive processing functions of the operators. However, the findings concerning the effects on Action errors are less conclusive and, in fact, it is still an area of much debate (6,8).

There are many causes of stress and some of these are not as obvious as otherwise expected. Fear, boredom and shift work can all cause changes to the level of stress experienced or perceived by the operator. In general, stress arises because the operator feels unable to cope with the demands of the situation and here is the definition of stress as used by Baker and Marshall (14) -

> "Stress is a subjective state, having negative connotations, which arises in response to a stressor and entails a feeling of inability to cope."

A detailed survey of the research on stress can be found in (14) as well as a full discussion on the effects of stress on the cognitive abilities of the process control plant operator. In addition, observations of the operators' behaviour during a crisis situation (11,12) appear to corroborate with the findings of (14). Research into the effects of stress on the Action error commission rates has produced some useful statistics (4) and are, up to a certain extent, supported by empirical results (8). A full discussion on these effects is beyond the scope of this work and only a summary of these effects is provided here.

4.2 The Effects of Stress

4.2.1 Cognitive Aspects

One of the major effects of stress is the degradation of the reasoning ability of the operator in the following ways:

1. Increasing rigidity in problem solving
2. Narrowing of the attention and perceptual field
3. Mindset
4. Reversion to Skill-based behaviour

When an operator is stressed, he will experience increasing rigidity in problem-solving. This means that his choice of solution will be influenced by the frequency a solution has been used and how regular the solution is, while the appropriateness of the solution will not necessary be foremost in his selection criteria.

At the same time the operator's attention and his perceptual field will be narrowed temporally and spatially, which can lead to important stimuli being ignored. In addition, the operator will be increasingly reluctant in admitting that wrong decisions have been made and this syndrome is known as mindset. Mindset is said to have occurred when the operator identifies the plant state incorrectly but persists in interpreting the data to fit the perceived state or hypothesis. As a result of mindset, the operator will tolerate a larger deviation from the expected symptoms to occur before other alternatives are explored.

Finally, when the operator is stressed, he will tend towards Skill-based behaviour and this reversion has been confirmed in the observation of operators performing during a crisis situation (12). Skill-based behaviour, as discussed in section 2.3.1, consists mainly of automatic responses. The degree of control exercised by the operator over his actions is lower than in Rule- or Knowledge-based mode and therefore there is more room for errors. When Skill-based behaviour is favoured over Rule- and Knowledge-based behaviour, the general level of control exercised by the operator is effectively less than when no such bias is present. When the operator reverts to Skill-based behaviour, the methods used to achieve certain goals will be biased towards those of which the operator has more practise. On the other hand, if the operator is operating normally in Rule- or Knowledge-based mode, the selection of methods will be guided by the appropriateness of the methods in the given situation.

4.2.2 Statistical Treatment of Action Errors

When an operator is placed under stress, he will become more prone to Action errors. As discussed in section 3.3.1, Action errors are closely associated with the activation of demons which in turn characterizes Skill-based behaviour.

The mechanisms of Action errors are essentially those associated with demon activations as discussed in section 3.3.1. It can be readily seen that Action errors are not random and are in fact highly context sensitive. This can be best illustrated using one of Reason's examples concerning year change (38). When a new year has just arrived, it can be predicted with a very high degree of certainty that one will miswrite the previous year instead of the new year sometime during the first month. However, in the middle of a year, this error becomes extremely rare.

4.2.2.1 Human reliability Approach

It is clear that a purely statistical treatment of Action errors will not sufficiently capture the essence of this type of error. The mechanisms by which these errors can be made further illustrate this deficiency. The statement, "the probability of miswriting the old year instead of the year is y", where y is a constant, is clearly incorrect because we know that the probability is dependent on the time of year the action is performed.

Nevertheless, Swain and Guttmann's probabilistic treatment of actions, known as the Human reliability Approach (HRA) (4) remains one of the best methods available for assessing Action errors. It is known that even top class operators can commit errors under perfect operating conditions and these errors cannot be eliminated, though the probability of error commission can be modified. This treatment is consistent with the fact that Action errors are inherent in every human activity. The error commission rates for this type of errors are proposed by Swain and Guttmann. The frequency and probability of Action errors are referred to not as a unit of time but as the number of operations performed and is known as the Human Error Probability distribution (HEP).

Action errors can be classified as follows (4) and the HEP for each type of action error was proposed for specific plant systems:

- Omission: failure to perform a given action
- Faculty execution: performance of the activity not according to instructions.
- Inappropriate action: action which should have been performed
- Sequential error: the order of actions is mixed up
- Time errors: intervention took place at the wrong time, either too late or too early

The Human Reliability Approach assumed that the probability of error can be calculated by means of a careful consideration of all the factors which can affect the performance quality of the operator. These are known as Performance Shaping Factors (PSF).

The real error probability can be obtained by modifying the base HEP with the Performance Shaping Factors associated with the effects present. The PSFs are applied to the base probability by multiplication.

The empirical results from the study by Beare et al (8) neither fully proved the validity of the proposed HEPs nor did they demonstrate the validity of the proposed HEPs there is no evidence to support the PSF proposed for experienced operators. The main objections to the statistical approach is that the underlying mechanisms for Action errors were not covered by this modelling method, and the assumption that the various factors affecting operator performance is multiplicative, and the effects of each factor do affect each other (14).

4.2.2.2 Stress and Other Factors

Although support for increase in errors in the high stress faults was observed only in the case of the Boiling Water reactor and not the Pressure Water Reactor, Beare et al (8) concluded that "despite the anomalous results observed in this study, it would be prudent to use the Handbook (Swain and Guttmann) modifiers for stress when performing human reliability estimates." The major problem associated with this conclusion is that other factors for increasing error rates have not been wholly eliminated.

There are many other performance shaping factors and the interactions between these factors are not well understood. For example, excess heat can cause discomfort which can in turn be interpreted as causing physiological stress. It is not certain whether temperature discomfort should be treated as a single stressor only, or as a separate environmental factor and contributes toward the degree of stress experienced by the operator. Furthermore, there is an optimum level of stress below which the efficiency and reliability of the operator is degraded (Figure 4.1) (12). This could be viewed in terms of boredom. In this study, stress is treated as the only source of error, which is in addition to the basic non-eliminable errors. The treatment of the bidirectional effects is discussed later in chapter 6.

4.3 Experience and Stress

The Human Reliability Approach proposes that novice operators will commit four times as much errors as

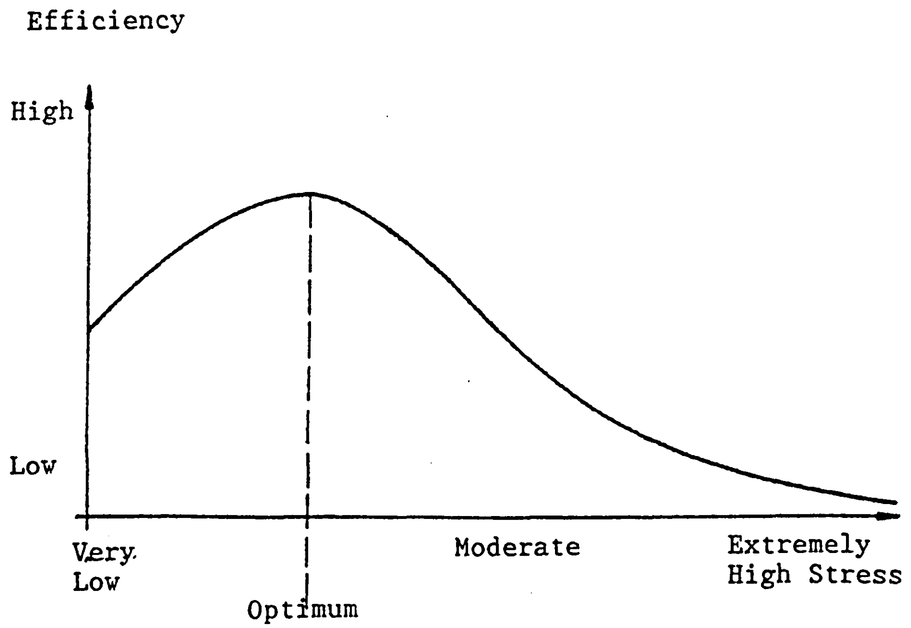


Figure 4.1 The Effect of Stress on The Efficiency of the Operator

experience operators though this prediction has not been supported by experimental evidence (8). There are however indications that experience does indeed lower the error commission rate slightly. Experience may act as a psychological buffer against stress in the more experienced operator but on the other hand, the experience of the operator may act as an additional source of knowledge which is not available to the novice operator.

Experience thus has two effects on the reliability of the operator. First, it affects the operator on the cognitive level and makes available a pool of knowledge. Second, it can reduce the level of stress experienced by the operator and in doing so reduces the rate of Action error commission. The precise role of experience is not clear and although its interaction with other stressors is not understood, it is reasonable to assume that experience is a significant factor in the coping of stress and plays a significant role in the cognitive processes of the human mind.

4.4 Modelling Methodology

Stress degrades the performance of the operator. This degradation manifests itself as physical and psychological changes in the operator. Unfortunately, these changes cannot be easily represented within a computerised model of the operator, so a different but necessary correct description of the degradation of performance must be adopted instead. In addition, since experience increases the tolerance of operators to stress, the interactions between stress, experience and the cognitive model representing reasoning processes of the operator must be clearly modelled.

Experience increases the body of knowledge available to a person, perhaps in the form of rules of thumb, direct correlations between symptoms and causes etc. At the same time, experience can sometimes enable the person to utilise better existing available knowledge to generate new knowledge. The generation of new knowledge from old knowledge is otherwise known as creativity and the modelling of which is beyond present day know-hows. The knowledge about the utilisation of knowledge is also known in Artificial Intelligence jargon as metaknowledge.

During the development of the error modelling framework, it was decided that the model will be separated into two submodels, cognitive and statistical (Figure 4.2). The cognitive submodel essentially models the reasoning processes and Decision errors. The statistical model

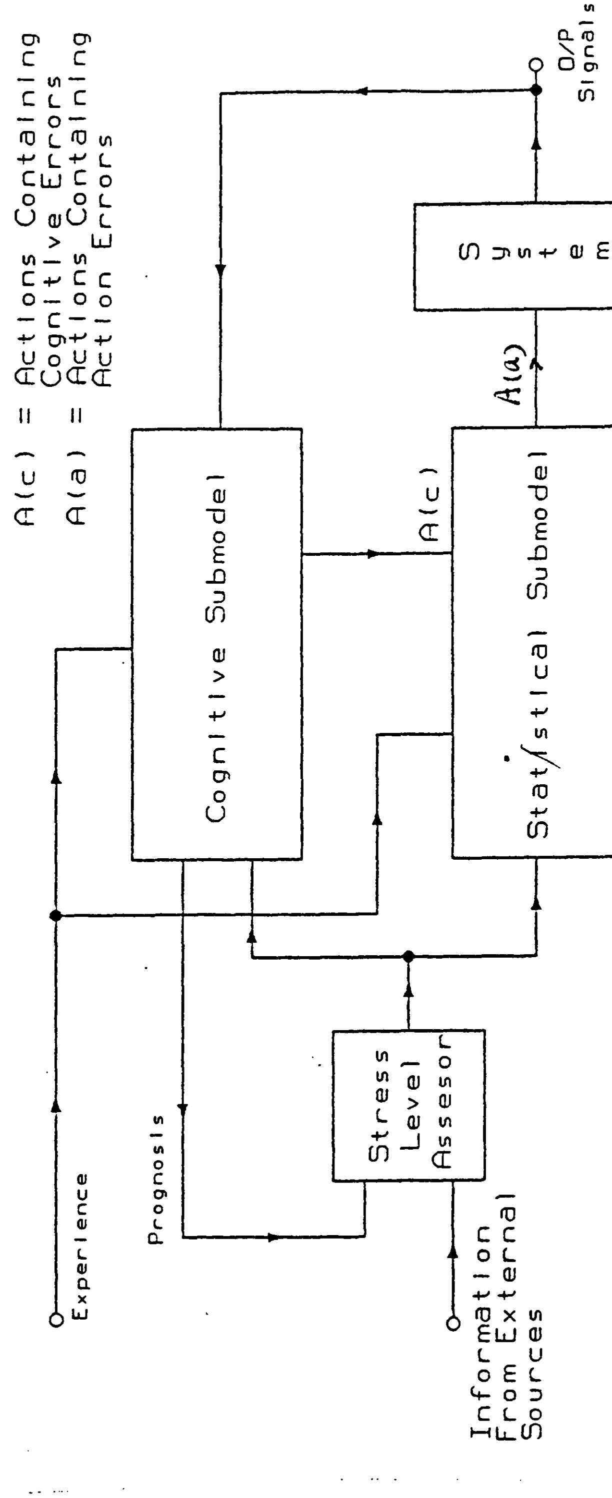


Figure 4.2 Interaction Between the Cognitive Submodel and the Statistical Model

models the generation of Action errors. Experience affects both of the submodels but within the confine of the cognitive model, experience is modelled as the addition of extra knowledge.

The two submodels are kept as self-contained as possible and the necessary interactions between them explicitly limited. The modular nature of the framework should enable a system to be gradually built up and the refinements made to a submodel will not necessitate major structural changes in the system. Since Action errors are found to have relatively less important effects, and due to the complexity involved, the statistical submodel will be excluded in the first instant and be introduced once the cognitive submodel has been refined.

Finally, the computer implementation of the cognitive submodel assumes the basic construct of an expert system. Strictly speaking, the cognitive submodel is not an expert system because the implementation does not require the most efficient processing and representation techniques, such as those used in an expert system. The implementation instead should closely mimic the known cognitive processing methods and these criteria will cause the implementation to become an inefficient expert system. What may be a cognitively efficient processing technique may not be an efficient computer implementation. In addition, the modelling of the effects of stress and the treatment of uncertainty will produce fundamental differences between the implementation of the cognitive model and an expert system.

Nevertheless, the cognitive model will be considered as an expert system whose function is to perform decision making according to some underlying rules. These decisions may be very high level and abstract, such as "identify the state of the plant" and "decide on a course of action" etc.

4.5 Definitions and Approximations Adopted

This computer model adopts the following definitions and approximations:

1. A perfect operator working under ideal operating condition is approximated to be entirely free of Decision errors.
2. The only observable effect of stress is defined as the introduction of errors, both Action and Decision errors.
3. The only observable effect of stress on the cognitive processes of the operator is defined as the

introduction of Decision errors.

4. The coupling between the cognitive model and Action errors is assumed to be weak and is approximated to be independent of each other.
5. Action errors are taken to be probabilistic and are not affected by the context in which they occur. The Human Reliability Approach is adopted here.
6. Experience will only be allowed to affect the size of knowledge available to the operator.
7. All changes in the cognitive processing mechanisms and the errors produced as a result of such changes are considered to be due to stress only.

Approximations (1) to (3) are necessary due to the nature and constraints of modelling using computers. Approximation (4) implies that the operator will not be aware of any Action errors and therefore no recovery action is possible in this model. Although it is known that recovery does indeed happen and the effects of the decoupling approximation are not clear, this approximation is considered necessary at this stage in order to reduce the complexity to a manageable level.

Although it is certain that Action errors are not entirely context free due to the mechanisms of Action error commission (see section 3.3.1), approximation (5) is considered to be reasonable at this stage of model development. Otherwise, all actions required to be performed by the operator will have to be scrutinised, taking the position of the action within a sequence, the physical demand of the action, the spatial distribution of the control elements and the appearance of the control panel. Later models can be refined by taking the context of actions into account.

Assumption (6) appears at first sight to be invalid because it is certain that experience will have some influence over the utilization of knowledge and information available to the operator and not simply increase the knowledge available. The restriction placed on the interactions between experience and the cognitive processing mechanism by this assumption is an artificial one. This assumption is needed for the ease of conceptualization of the operator model and its implementation, which will become clear in chapter 5.

The additional knowledge, which does not affect the cognitive processing mechanisms of the operator model structurally, can be in the form of constraints e.g., physical laws which must not be violated, and empirical knowledge serving as short cuts. This restriction merely implies that constraint checking is built into the basic cognitive processing mechanisms, although there may not be any constraints to satisfy. In the case of an experienced operator, constraints are present in the knowledge base and will have to be adhered to while the novice will not be aware of any constraints nor short cuts etc.

4.6 Cognitive Submodel

As stated in approximation 3 in section 4.5, the only observable effect on the cognitive processes of the operator is defined as the introduction of Decision errors. There is as yet no successful mathematical model available for the representation of various cognitive processes, but cognitive models based on psychology are available (9,16,17,18,19,20,21,22). The representation of Decision errors should therefore compliment the available cognitive models and should be possible to be incorporated in a computer implementation of such a model.

4.6.1 Representation of Decision Errors

During normal operation (when the operator is not stressed), a "rule" will fire only when all preconditions are satisfied and hence a decision, which is always correct, will be taken. When the operator is under stress, a wrong decision can be chosen although not all the required conditions are met. The basic definition of the effects of stress on the cognitive system used in the computer implementation is now extended to become:

> The only observable effect of stress on the cognitive process is the introduction of decision errors. Unstressed decision making is analogous to a one to one mapping of conditions and decision (which can be translated into actions) whereas stressed decision making is analogous to a one to many mapping (Figure 4.3 and 4.5)

In other words, when the operator is not stressed, each unique set of conditions will produce only one set of response. For example, if y is the response of the operator and is dependent on x, the condition, then y = f(x) is a single value function. Under a normal situation, it is assumed that the operator will always

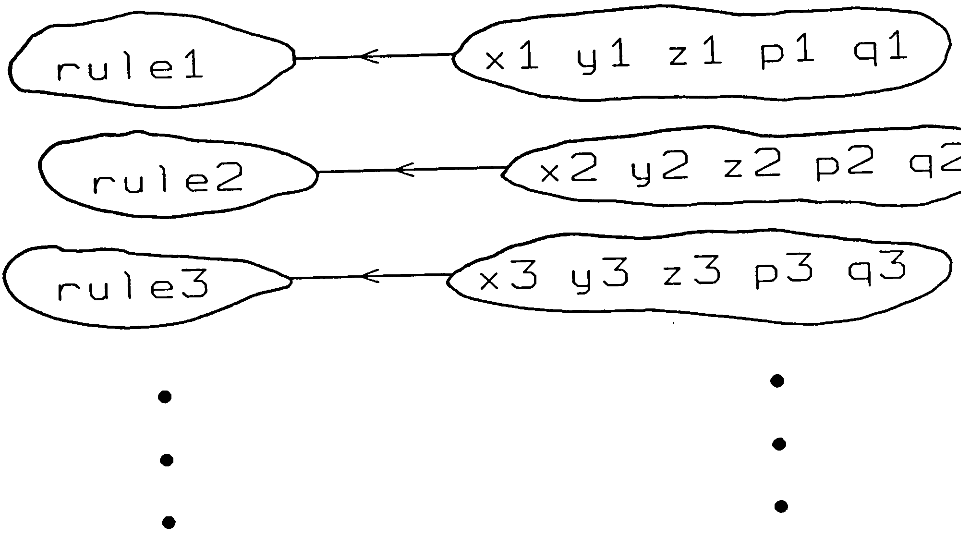


Figure 4.3 One-to-one representation of Unstressed Decision Making

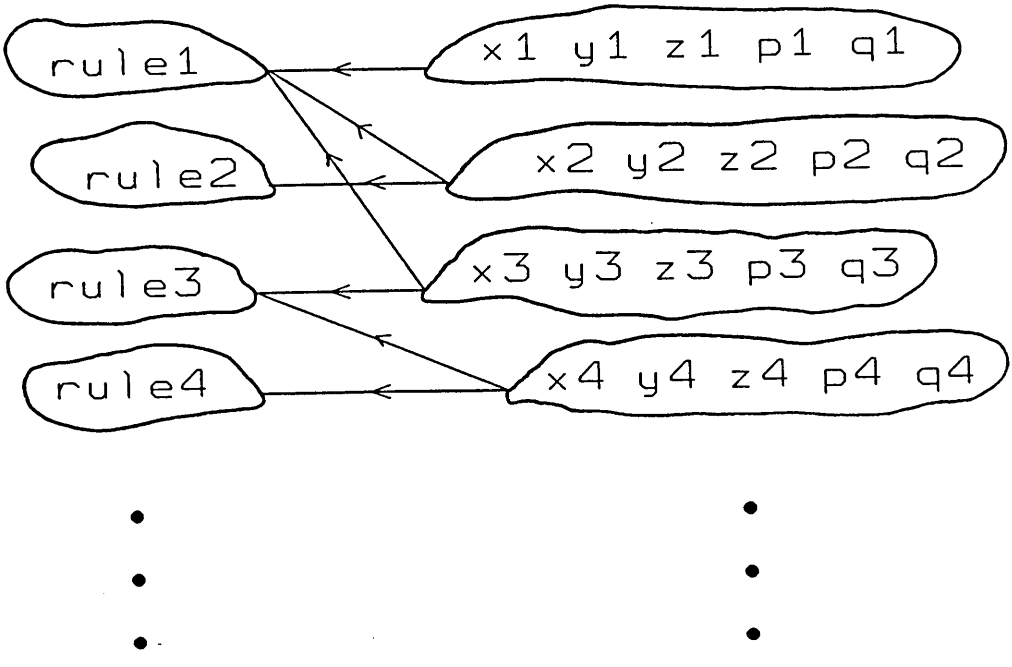
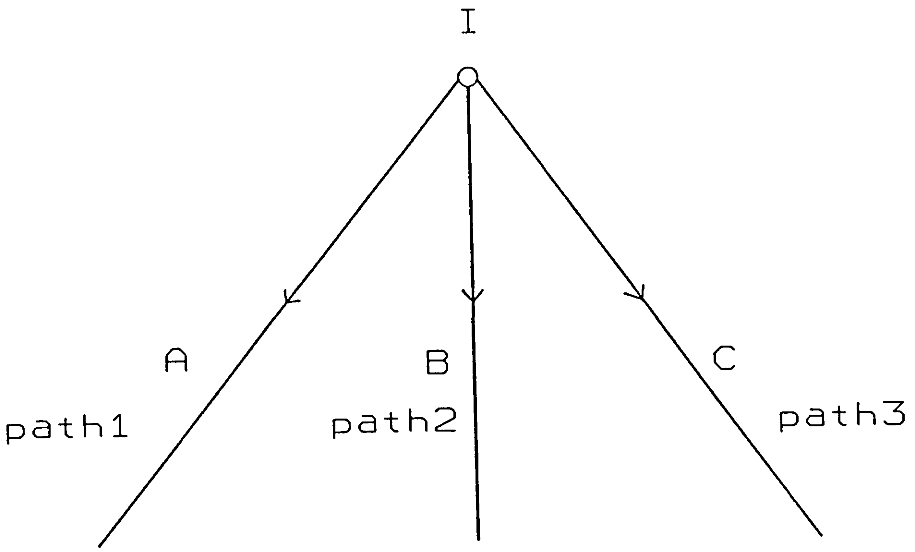


Figure 4.4 Many-to-one Representation of Stressed Decision Making



Node I, where a decision is required

Figure 4.5 Unstressed Decision Making

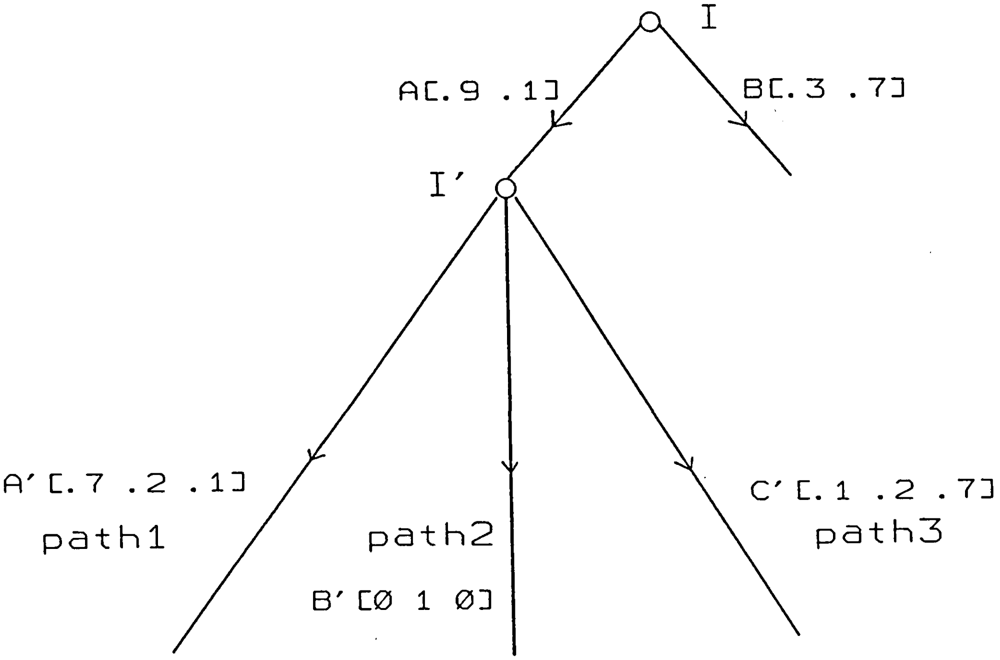


Figure 4.6 Fuzzification of Decision Paths Due to Stress

select a certain path if the prerequisite conditions are met, e.g., if condition A is true, path 1 will always be selected (Figure 4.5).

Node I represents the point where a decision is required, which can be translated into a response. This decision will be made according to some predefined conditions of the plant and the operator will seek information concerning the plant. Therefore, the input to the decision node is plant information and the output is a decision.

When the operator is stressed, he may make a wrong decision and this is reflected in the operator taking a wrong path. In this case, even if condition A is true and condition B is not, there will still be a finite chance that path 2 will be selected and this is analogous to a one to many mapping (Figure 4.4 and 4.6).

For example (Figure 4.6), condition A is true at node I. In this case, although condition A is definitely true, it is not the path to be taken that is determined but the probability for each path available for selection at node I is defined. This probability distribution is dependent upon the condition prevailing at the time. This means that although it is not known which path will be taken, there is 90% chance that path 1, which is the correct path, will be taken but at the same time there is a 0.1 probability that the wrong path, path 2 will be chosen. The precise path selected will be determined only at run time, using a weighted random number generator.

Conversely, if condition B is true then it is known that there is 70% chance for path 2 to be selected and 30% chance for the wrong path, path 1 to be taken. In this example, it can be readily appreciated that the chance for an error to occur is higher when condition B is true.

4.6.2 Determination of Selection Likelihood

As mentioned in section 4.6.1, the probability distribution for an individual path is dependent upon the condition prevailing at the time. In ordinary expert or knowledge-based systems, many methods have been used to enable an unambiguous decision to be selected based on uncertain or fuzzy real world data. In most of these implementations, such an ambiguous choice is made on the degree of support each received. In effect, a filtering mechanism is used for path selection. Figure 4.6 can be reduced to Figure 4.5 by selecting the path with the highest support.

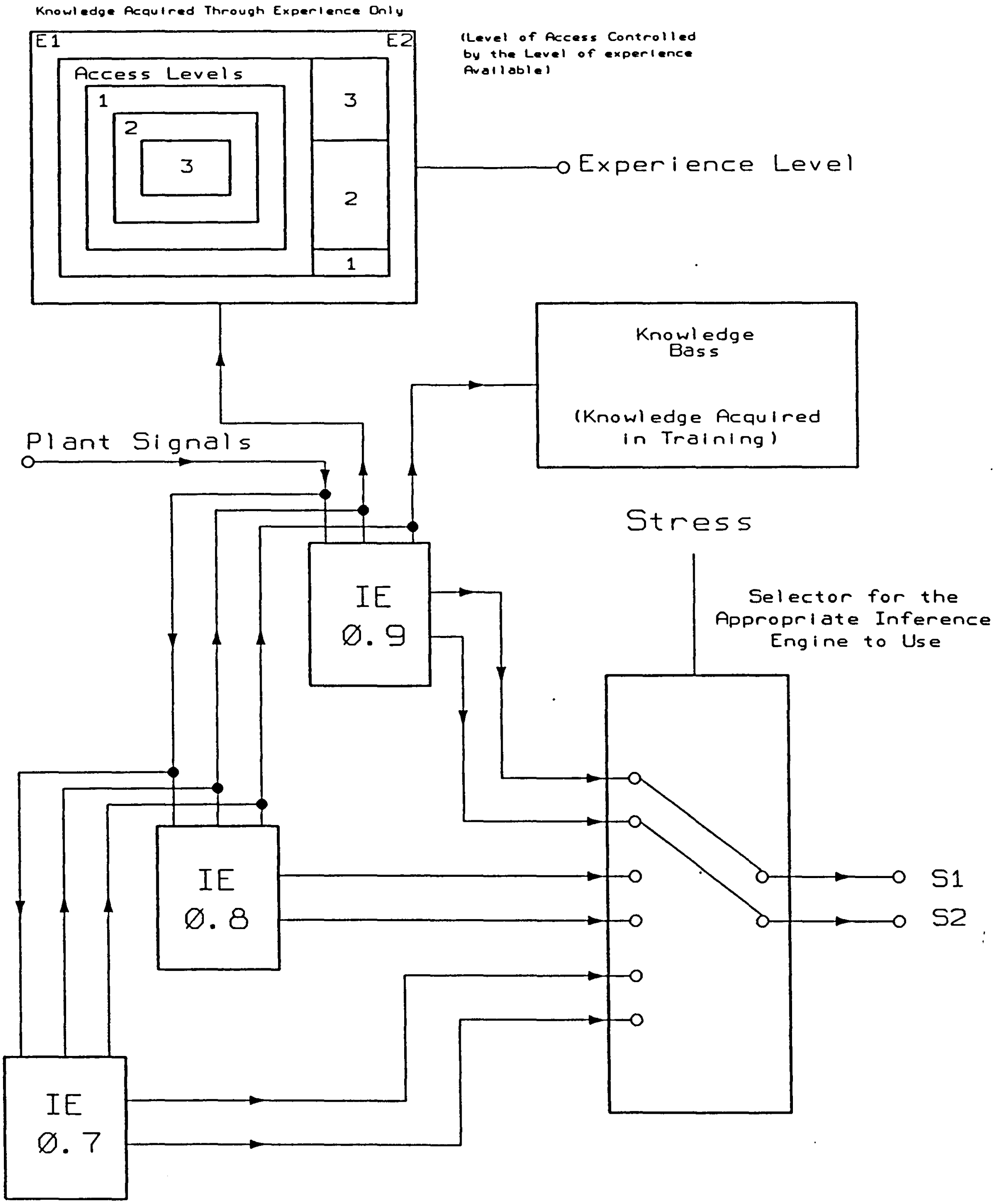
There are two methods by which the error likelihood can be obtained. The most obvious of which is by direct consultation with the operators. There will obviously be variations on the perceived probability for each path for each condition, so the amount of data which has to be acquired will be very large. The second approach, which is the preferred method here, is to utilize the uncertainty associated with human knowledge.

The knowledge acquired from the operator will invariably involve some linguistic descriptors such as fairly high, rising rapidly etc. Real world data are mostly numerical data and these are often translated into natural language form. In this proposed representation of errors, it is proposed that the uncertainties due to the vagueness of the data should be retained and instead of a "depth first, search until succeed" strategy, a parallel search mechanism should be employed. The result of the parallel search will provide a set of supports for all the possible paths for selection at the decision node. These supports can be used as the inherent probability distribution for decision errors and will be used for path selection probabilistically. The result is that at each decision node, the path selected will not be deterministic but probabilistic. This implies that the decision may not be the same given the same set of inputs, which is precisely the nature of errors.

4.6.3 Representation of the Degradation of Inference Ability

Stress degrades the reasoning abilities of the operator and, since the inference engine of an expert system is responsible for reasoning, it seems reasonable to model this degradation as the production of an imperfect inference engine. In this model, different inference engines, each with a different degree of accuracy, will be used for different stress levels. For the ease of conceptualization, inference engines with different accuracy are considered to be represented as separate entities (Figure 4.7). In practice, difference in accuracy of the reasoning ability may or may not require big structural changes to the inference engine. In some cases, the changes may be very minor such as the lowering of the tolerance threshold. The set of inference engines will need to be carefully designed to ensure that no unwanted discontinuity results.

The effects of stress, in terms of problem solving rigidity, narrowing of the attention and perceptual field, mindset and the reversion to skill-based behaviour



E1 = Rules of Thumbs and Knowledge Acquired Through Experience only
E2 = Constraints Checking Levels and Constraints Used
S1 = Plans of actions, Input to the Statistical Submodel
S2 = Prognosis of the system
IEx = Inference Engine with x Accuracy

Figure 4.7 Cognitive Submodel

can be represented as changes in the inference strategies. These changes will certainly have effects on the performance of the inference engine and therefore changes in inference engine accuracy will be facilitated by introducing changes to the inference strategies.

4.6.3.1 Problem Solving Rigidity

When the operator is stressed, he tends to avoid the use of novel problem solving methods. He may select a conventional response such as a well tried method instead of pursuing more suitable solutions (11,12,14). In other words, there appears to be a restriction on his behaviour in terms of his readiness to accept novel solutions to problems. This rigidity can be represented by placing restrictions on the possible plans generated by the inference machine. Since a plan will invariably consist of bits of well known and tried sub-plans, a "conventional" metric can be used to evaluate the conventionality of the plan. When the metric exceeds the acceptable threshold, a different plan must be generated. Increasing problem solving rigidity will imply an increase in the acceptance threshold. The metric will be based on the total sum of the conventional rating associated with each integral sub-plan.

4.6.3.2 The Narrowing of the Attention and Perceptual Field

The narrowing of the perceptual field can lead to important stimuli being ignored. These stimuli can be in the form of flashing indicators, the fast rising of a dial needle or even actual changes in the reading of a gauge. There is a distinction between information and readings specifically requested by the operator and those only monitored by the operator. Generally, those readings requested consciously will always be registered and therefore, the narrowing of the attention and perceptual field is closely related to the general monitoring of data. This narrowing of perception will be modelled as the ignoring (see chapter 5) of changes of readings not specifically requested by the operator. Since monitoring can be modelled statistically, this effect is considered to be largely statistical in nature (chapter 5).

4.6.3.3 Mindset

In the cognitive model used by Bersini et al (16), the recognition of a plant state is based on similarity matching of the stored state frames. These plant states provide a state description in the form of a state label, which is a set of attribute values associated with the diagnostic signs of a particular state. The selection of the frame to be instantiated (i.e., "recognized" by the operator) involves a detail parallel exploration of a set of possible frames. The attribute values which characterize a specific state are often in vague linguistic terms. Consequently, it is not expected that a perfect match will result. The best partially matched frame, which in most cases will be the frame with the highest support, will be selected and a plan of actions appropriate to the plant state associated with the instantiated frame will be devised and carried out.

When the marginal support between the frames is very small, and/or the absolute supports for all the frames are low due to uncertainties in the data, the uncertainty concerning the correctness of the instantiated frame is high. When this happens, the operator will normally seek confirmation from the subsequent behaviour of the system. If, however, the behaviour of the plant becomes different from the expectation of the operator due to his initial diagnosis, the operator will conclude that a wrong frame has been instantiated and alternatives will be sought and actions more appropriate to the situation will be taken. When Mindset occurs, the operator will be very reluctant to admit that a wrong hypothesis or frame has been used even when the symptoms indicate otherwise.

Mindset can therefore be modelled by a simple increase in the deviation tolerance threshold and a change in the instantiating mechanism. In addition, the monitoring process will also be affected. The searching strategies employed in the selection of frames can be modified. For example, the hybrid searching mechanism, "best-probables/breadth-first" search (i.e., breadth-first search is performed on the group of paths which are most suitable) can be changed to "best-probables/depth-first" search.

4.6.3.4 Reversion to Skill-based Behaviour

When an operator is stressed and faced with an unfamiliar situation, he is very likely to revert to first learnt behaviour. This means that he may be inclined to take a course of actions which he has practised many times over, though it is not quite appropriate to the on-going situation. He will prefer to do that than to devise a course of action specifically for that situation. The well practised actions are mostly actions for a well known situation or a group of integral actions which are strongly linked to each other. Therefore, this behaviour can be represented by the substitution of the well documented plan associated with the frame most similar to the on-going situation. Furthermore, some of the actions scheduled to be performed can be substituted with a different group of actions with strong associations with each other.

The modelling of the reversion to skill-based behaviour places certain constraints on the underlying cognitive model used. It is implicit that the cognitive model must be able to model separately skill-based, rule-based and knowledge-based behaviour which are the characteristics exhibited by process control operators. If skill-based, rule-based and knowledge-based behaviour cannot be modelled separately by the cognitive model, it will not be able to facilitate a reversion of behaviour. If the mechanisms used in the modelling are not clearly defined, the boundary between the three separate behaviours cannot be distinguished. In view of the above requirements made on the underlying cognitive model, a new cognitive model which will satisfies all the necessary criteria is proposed in chapter 5.

4.7 Statistical Submodel

All Actions errors can be classified into the following types:

Error of omission (omit step or entire task)
Error of commission:
Selection Error (select wrong control, misposition
  control, issue wrong command)
Sequence Error (action carried out in wrong order)
Time Error (too early or too late)
Qualitative Error (too little/too much)

Human Error Probabilities for the occurrence of each of the above types of errors are suggested by Swain and Guttmann (4). Stress and experience serve to modify these underlying probabilities.

When the cognitive submodel decides on a course of actions, a schedule for the actions to be performed on the system will be issued. If no error occurs, this action will be translated into a low level instruction to the system and such an action will be effectively "executed" (performed by software) by the operator. In the Statistical Submodel, the level of experience and stress will be used in conjunction with the fundamental Human Error Probabilities (HEP) to generate a HEP for the appropriate level of stress and experience. This probability distribution is then used to decide, probabilistically whether the action ordered by the cognitive submodel will be successfully completed (Figure

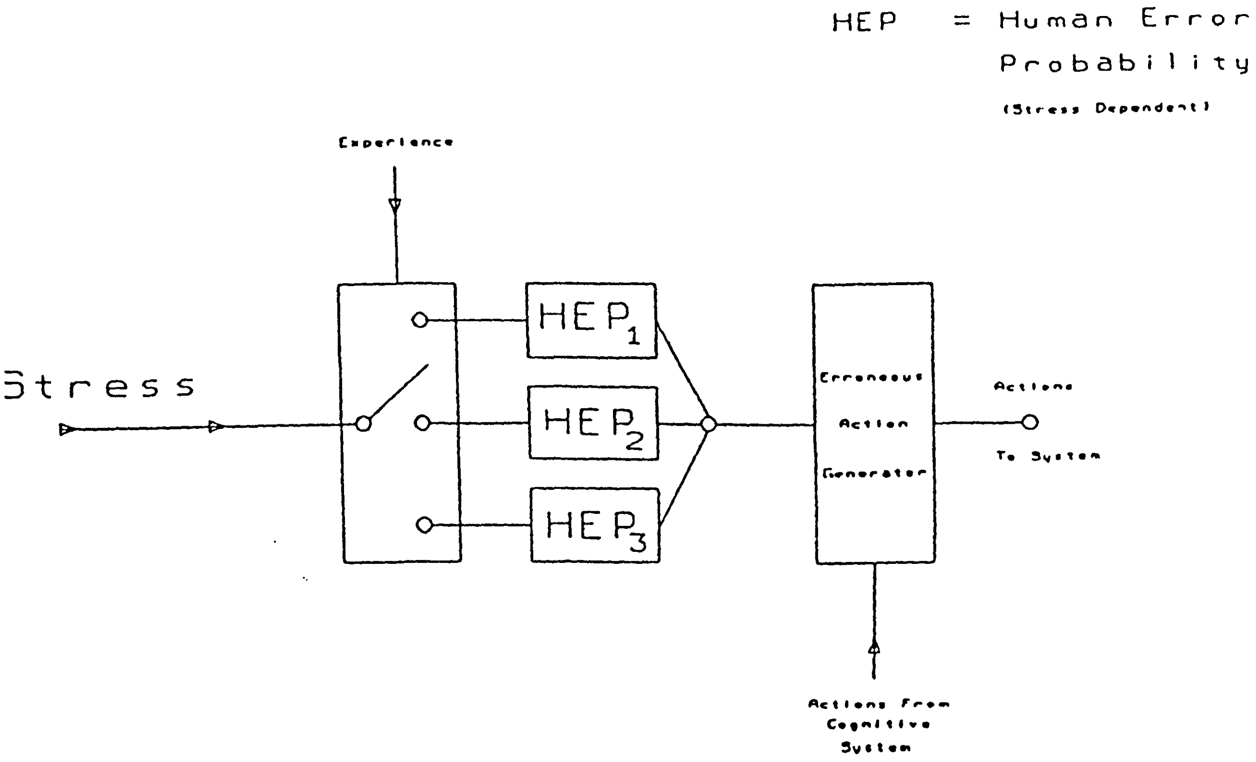


Figure 4.8 The Statistical Submodel

4.8). The error introduced will be of type selected statistically at run time.

For omission errors, the actions will simply be ignored. For Selection errors, a different control will be substituted depending on the actual configuration of the control panel being used. For Sequence errors, the next action will be performed before the original action required. For Time errors, the execution of the required action will be delayed or put forward and, finally, for Quantitative errors, the magnitude value associated with the required action will be scaled up or down.

4.8 Summary

In this chapter, a framework for modelling the effects of stress on the operator was proposed, and in the next chapter, a cognitive model suitable for use in the proposed modelling framework will be discussed.

5. The Cognitive Model

5.1 Introduction

In order to model the cognitive processes of a process control operator, two points must be borne in mind. First, this cognitive model must be able to model the basic human cognitive functions and their associated everyday errors. Second, it must be able to model the specific behaviour exhibited by the process control operators.

Existing works (16,17) on software implementation of the behaviour of a process control operator have concentrated on the basic human cognitive processes and have produced encouraging results in the modelling of skill- and rule-based behaviour. The implementation by Cacciabue et al has so far treated the behaviour of the operator as continuous and the obvious discontinuity (the switches between the three separate types of behaviour) has not been modelled explicitly.

The cognitive model used by Cacciabue et al reflected all the basic characteristics of the cognitive system described in section 2.2, though Rasmussen's (9,18) classification of the three different types of behaviour is not included. As it is, the model allows readily for different Action error mechanisms to be modelled, but the model does not offer any way forward for the modelling of the reversion from Knowledge-based behaviour to Skill-based behaviour. It is felt that in order for this reversion to first-learnt behaviour to be modelled at all, a different cognitive modelling approach is required.

The cognitive processes involved in skill- and rule-based behaviour appear to be relatively similar but a clear distinction seems to exist between the cognitive processes associated with the knowledge-based behaviour (KB) and skill-/rule-based behaviour (SR-B). This model attempts to model these two separate categories of cognitive processing mechanisms by varying the models used in the modelling of the subprocesses such as monitoring and recognition. The basic conceptual architecture of this cognitive model is similar to the one adopted by Cacciabue et al (16,17) but contains major differences in the modelling of monitoring and recognition.

5.2 Skill/Rule-based vs Knowledge-based Mode

During normal operation, the major task of the operator is monitoring. If something unusual happens, the operator first of all will realise as a result of monitoring that the plant is operating abnormally. He will try to determine the cause of the abnormality, in order to return the plant to its normal operating state.

If the operator can match the symptoms of the plant readily and quickly to the symptoms of some known (standard) faults, he will have successfully "recognised" the fault. Once the operator recognises the fault, he will be able to retrieve from his memory or from some manual the relevant actions he is required to perform and no elaborate thinking will be involved. The operator will then be working in SR-B mode. The sequence of events leading to the SR-B mode can be summarised as follows:

1) Monitoring in "normal mode"
2) Something triggers the operator into suspecting that something is wrong.
3) The operator tries to determine whether an abnormality has occurred. If an abnormality has occurred, the operator tries to match the symptoms to the symptoms of some known faults stored in his memory.
4) A match is obtained and the fault is "recognised".
5) The operator tries to retrieve from memory the actions associated with the fault. If a plan for that fault is stored in the memory, the operator will be in skill-based mode.

6) No plan is stored in the memory, the operator needs to consult the manual and follow the instructions laid down in the manual.
7) The operator is in rule-based mode.
8) Monitoring in "abnormal mode" and performing recovery actions.

It can be seen that SR-B mode is entered only after the abnormality is successfully recognised (matched). When the operator fails to recognise the fault, he will still have to decide on a course of actions. The operator will then engage in some elaborate thinking, and he will be operating in KB mode. It can be readily seen that the sequence of actions that leads to the KB mode is similar to that of the SR-B mode. The first four actions will still have to be performed by the operator but the result of action (4) will be different in that no "acceptable" match is made and the fault is not recognised. When this happens the operator will need to reevaluate the abnormality and select the most suitable course of actions. This immediate course of actions may belong to the associated plan of the most similar fault, though this fault is deemed not to be the on-going abnormality, or the actions will be selected according to some other criteria.

Due to the uncertain nature of the real world, there will always be slight variations on the strength of the symptoms associated with any particular fault. In fact, the symptoms of each fault are most likely to be described by some natural language descriptors that can in turn be modelled by using fuzzy sets. Therefore the result of the matching will not be a clear cut yes or no, instead it will be an indication of the degree of similarity between the two sets of symptoms. It can be readily appreciated that a threshold value will almost always be needed to define what constitutes an "acceptable match". Alternatively, frequency gambling, based on the frequencies of each state occurring in the past may be employed (16,17).

When no match (from now on, no acceptable match will be referred to as no match) can be made, the operator's memory will have to be searched again for a suitable candidate. It must be clear that the searching method used in step (4) will no longer be suitable and some modifications are necessary. One of these modifications could be the lowering of the threshold for a match. Although this may seem reasonable at first sight, it is not always very satisfactory because symptoms associated with the abnormality may produce several partial matches, each with very similar degree of match. Obviously, a different type of selection mechanism must be employed.

It is felt that there should be two different types of monitoring performed by the operator, one type for normal operation (normal mode monitoring) and a second type (abnormal mode monitoring) is when the operator is aware that the system is functioning abnormally. Since monitoring involves "looking" and hence eye movements, each indicator can be assigned a probability of whether it is looked at a specific time but cannot be said categorically whether it will be looked at or not. Monitoring is one of the major roles of the operator. The operator cannot become aware of a certain condition if he did not see it. Regardless of how many red lights are flashing, indicating something is not working, the operator will continue to assume that nothing is happening if he did not notice the flashing indicator.

Researches in psychology suggest that the methods used to categorise problems and the basis employed in similarity judgements by the problem solvers are appropriate characteristics for distinguishing between good and bad problem solvers. These same characteristics seem to be valid also in distinguishing between novice and experienced problem solvers (19). It is felt that the major differences between the methods used in modelling SR-B and KB behaviour would be in the theories of similarities used, methods of searching through the knowledge base, criteria for selection and finally the methods used for modelling monitoring. These major differences between SR-B and KB behaviour are discussed in detail and methods for modelling these processes are proposed in the following chapters.

5.3 Modelling the Monitoring Process

Modern industrial systems are both large and complex. The level of automation and computer control is also high and rising. In addition, many safety features are incorporated into the system and this is particularly true when high reliability is required such as in nuclear power plant. These systems are designed such that even when faults do occur, the built in safeguards will ensure the safe and most probably continuous operation of the plant, because total stoppage will invariably result in financial losses. Therefore automation has shifted the role of the human operator in modern systems from the system operator to that of the system monitor.

System monitoring demands a high level of alertness and vigilance, requirements that are not consistent with the long hours and periods of inactivity associated with working on a smoothly operating automated plant. Although automation does reduce the degree of control possessed by the operator and does provide some level of automatic monitoring, it cannot, should not and does not reduce the operators' ultimate responsibility for the plant. Indeed it might be argued that automation reduces further the stimulation presented to the operator. Under these circumstances the sole role of the operator is monitoring which involves periodically scanning the system parameters and checking that values of these parameters are normal. If the system is operating in steady state, the values of these parameters are not expected to change very much. So, the main component of this monitoring is noticing changes from steady state values and, for the rest of this paper, the term "normal mode monitoring" will be used to refer to this particular type of monitoring.

If a significant change in value is noticed, the operator may then have to decide whether the automatic controls will be able to cope, or it is something that requires actions from him. At this point, the operator will be actively seeking information on specific plant parameters and his monitoring behaviour will be different from the "normal mode" monitoring. The term "abnormal mode monitoring" will be used to refer to this type of monitoring.

5.3.1 Monitoring and "Looking"

Information concerning the state of the plant is displayed in the control room and, in addition, if something requires the attention of the operator, a warning will be given. These warnings normally take two forms, visual or acoustic and in this study only visual monitoring will be modelled.

The values of the system parameters are generally displayed as a dial or a digital gauge. Visual alarms generally take the form of charts and flashing lights etc and some of these may even be coupled with sound. In order to perform routine monitoring, the operator will need to periodically look at each parameter displayed in the control room. Since it is obvious that the operator cannot look at all these gauges simultaneously, due to the finite nature of his vision field, he must exercise a choice as to which gauge he will look at next. The operator, however, in normal mode monitoring will not be making a conscious choice. This lack of awareness in making a selection poses a real problem in the modelling of monitoring. Since there is a clear relationship between the physical action of looking and monitoring, an examination of the process of eye movements may produce some clues.

When there is no specific inclination to look at anything in particular, eye movement has been modelled successfully by employing statistical techniques (39 and 40). It is suggested here that normal mode monitoring, which is not heavily directed by the mind, can be modelled statistically.

Consider the case of a panel of identical indicators, evenly distributed on the control panel and each with equal importance. It is assumed here that associated with each indicator is a function which gives the probability of it being in the direct line of sight.

These probabilities are spatially dependent, i.e., the position of the indicator relative to the eye will determine this probability. It must be stressed that these probabilities do not give any indication of whether the indicator will be "seen", it merely indicates the likelihood of the indicator being "looked" at. This probability function will be called Spatial Factor (SF).

In addition, each indicator will have a different level of visual dominance, such as brightness, colour of indicator and frequency of flashing etc. The level of visual dominance will have some effect on the probability of the indicator being noticed if the indicator is in the direct line of sight. The probability distribution function due to visual dominance factor alone will be called Visual Factor (VF).

There are other effects that can influence this probability. For example, consider the case of a group of indicators positioned relatively close to each other and most of the indicators are lit up except one. When that indicator eventually comes on, there is a high probability that it may not be noticed. However, if none of the indicators among that group is lit up, when that same indicator comes on, it will have quite a high chance of it being registered due to its high visual dominance. This particular effect is difficult to model because of the large combination of indicator states and it implies that the probability distribution function VF will not remain static. This second order effect is not modelled in this study but its inclusion in future models is the next logical step forward.

Finally, there is the importance the operator associates with each indicator. The operator may have acquired this association during training or gained it during actual operation of the plant. This association will influence the amount of attention the operator places on these indicators. However, it must be stressed that this influence is assumed to operate at the sub-conscious level and will act only as a modification factor on the final probability distribution function of the indicators being noticed. The distribution based on the relative importance of the indicators will be called Importance Factor (IF).

During routine operation, i.e., in normal mode monitoring, when the values of the system parameters are not expected to vary much, and the Importance Factor of the indicators will remain more or less constant. However, when something abnormal happens, the operator will have an expectation of what indicators will come on next. He will be seeking confirmation and further evidence. In this case, he will be consciously reading certain indicators and in fact, the SF (shows the probability of the indicator in the line of sight) for the indicators will change. Therefore, it can be readily seen that during abnormal operation, the Importance Factor will also be affected.

During abnormal operation, the operator will be monitoring the parameters he expects to change or perceives to be very important. In addition he will also need to continue routine monitoring. From now on, the monitoring that can be performed without any involvement of the operator's consciousness will be termed "random monitoring" and the ones which are influenced by the operator's consciousness will be termed "directed monitoring".

It can be readily appreciated that normal mode monitoring is basically a form of random monitoring with a small influence from the perceived importance of individual indicators. It is argued here that the perceived importance of the indicator will only operate at the subconscious level. Therefore there are two separate but related directed monitoring processes, directed subconscious monitoring and directed conscious monitoring, d^1 and d^2 monitoring respectively. Abnormal mode monitoring will be mainly directed conscious monitoring d^2, with some contribution from random monitoring.

The indicators are usually distributed on some control panels. If the indicators are indexed, we can represent the Spatial, Visual and Importance factors associated with the set of indicators as three vectors, $\underline{S}$, $\underline{V}$ and $\underline{I}$. $\underline{S}$ and $\underline{I}$ can vary but $\underline{V}$ will remain constant for a particular control panel configuration. It is assumed that at any given one time, only one indicator can be looked at.

There are in effect three separate components to the monitoring processes, and each of these processes contribute to the probability of the indicator i to be registered by the operator. The individual contribution from each of the processes will be different. We will now consider the probability P_i for a particular indicator i to be registered by the operator at a given time, given that an indicator is looked at by the operator.

Therefore the probability P_i may be written as:

$$P_i^n = \alpha P_i^r + \beta P_i^{d^1} + \Gamma P_i^{d^2} \qquad \text{------------(1)}$$

where $\Sigma P_i^n = 1$

P_i^n = Probability that indicator is noticed given that an indicator is looked at by the operator and the monitoring performed is in normal mode.

and $\alpha + \beta + \Gamma = 1 \qquad$ ------------------(2)

P_i^r = Probability that indicator i is noticed given that indicator is looked at by the operator operating in random monitoring mode.

$P_i^{d^1}$ = Probability that indicator i is noticed given that an indicator is looked at by the operator operating in directed subconscious monitoring mode.

$P_i^{d^2}$ = Probability that indicator i is noticed given that an indicator is looked at by the operator operating in directed conscious monitoring mode.

α = Probability that the monitoring performed by the operator is in random mode.

β = Probability that the monitoring performed by the operator is in directed subconscious mode.

Γ = Probability that the monitoring performed by the operator is in directed conscious mode.

Let us consider the factors which influence the value of P_i. For the value of indicator i to be registered, first of all it must be in the line of sight of the operator. Here it is argued that the probability of the indicator to be noticed is dependent on the indicator's visual dominance, and its spatial position and how important the operator perceived it to be.

There are in fact two importance factor IFs, I_i^1 and I_i^2. I_i^1 is the underlying importance associated with each indicator, which the operator learnt during training and from past experience. For each indicator i, I_i^1 is a constant and varies according to the type of system.

The second IF, I_i^2 is dynamic and changes with time. Its value is influenced by the expectation of the operator. For the set of indicator $D = \{D_1, D_2, D_3, \ldots\ldots D_i, D_j\}$, it is assumed that Random monitoring is operated without direct participation of consciousness. So:

P^r_i is therefore a function of the visual dominance of indicator i, V_i and the spatial factor S_i only.

$$P^{r^1}_i = f(I_i, S_i) \text{ ------------------------(3)}$$

Directed subconscious monitoring depends only on the importance the operator acquired during training. So:

$P^{d^1}_i$ is therefore a function of the spatial factor Si and its basic importance factor I^1_i only:

$$P^{d^1}_i = g(I^1_i, S_i) \text{ ------------------------(4)}$$

Since directed conscious monitoring is influenced by the expectation of the operator at run time. So:

$P^{d^2}_i$ is a function of I^2_i only (the dynamic importance factor only) because it is assumed that the operator will always direct his line of sight to what he expects to change. The operator may be considered to be seeking confirmation.

$$P_i^{d^2} = h(I_i^2) \text{ ---------------------------(5)}$$

5.3.2 Normal Mode Monitoring

Let us consider equation (1) in more detail (Figure 5.1).

In normal mode monitoring, Γ is equal to zero because directed monitoring only operates at the subconscious level. α and β are the weights of contribution from either modes. So equation (1) becomes (combining equations 1, 3 and 4):

$$P_i^n = \alpha f(V_i, S_i) + \beta g(I_i, S_i) \text{ ---------(6)}$$

and $\alpha + \beta = 1$ -------------------------(7)

5.3.3 Abnormal Mode Monitoring

In abnormal mode monitoring, the operator will be actively seeking confirmation and additional evidence. First of all, it is assumed that directed (conscious) monitoring will be the dominant factor. On the other hand, the operator will

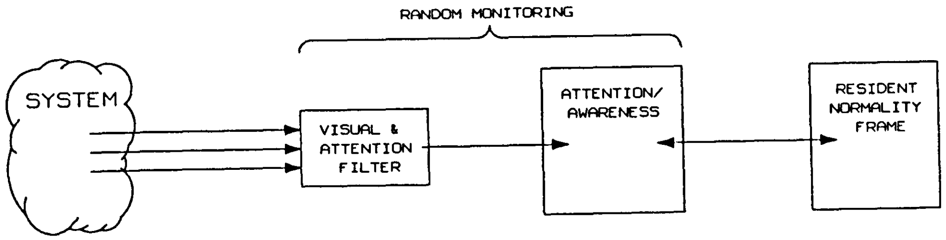


Figure 5.1a Normal Mode Monitoring

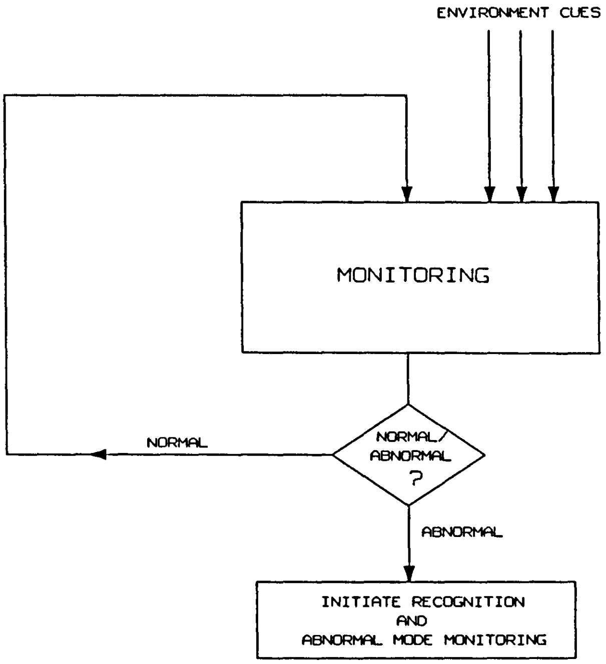


Figure 5.1b Normal Mode Monitoring

still need to monitor the other important parameters although the he does not expect them to change (Figure 5.2). Therefore

$$P_i^a = \delta(\alpha P_i^r + \beta P_i^{d1}) + \Gamma P_i^{d2} \text{--------------}(8)$$

where $\delta(\alpha + \beta) + \Gamma = 1$

α and β hold the same value and meaning as in normal mode monitoring.

δ = Probability that the monitoring performed by the operator is in normal mode.

Γ = Probability that the monitoring performed by the operator is in directed conscious monitoring mode.

Therefore $P_i^a = \delta[\alpha f(V_i, S_i) + \beta g(V_i, I_i)] + \Gamma h(I_i)$

5.3.4 Transition Between Normal and Abnormal Mode Monitoring

As described in section 2, the operator will be monitoring in normal mode until he suspects that something is wrong. He will then attempt to determine whether there is actually an abnormality and if an abnormality exists, he will then

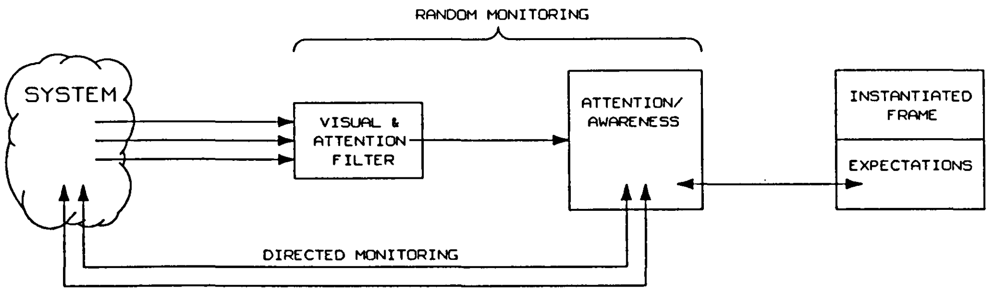


Figure 5.2a Abnormal Mode Monitoring

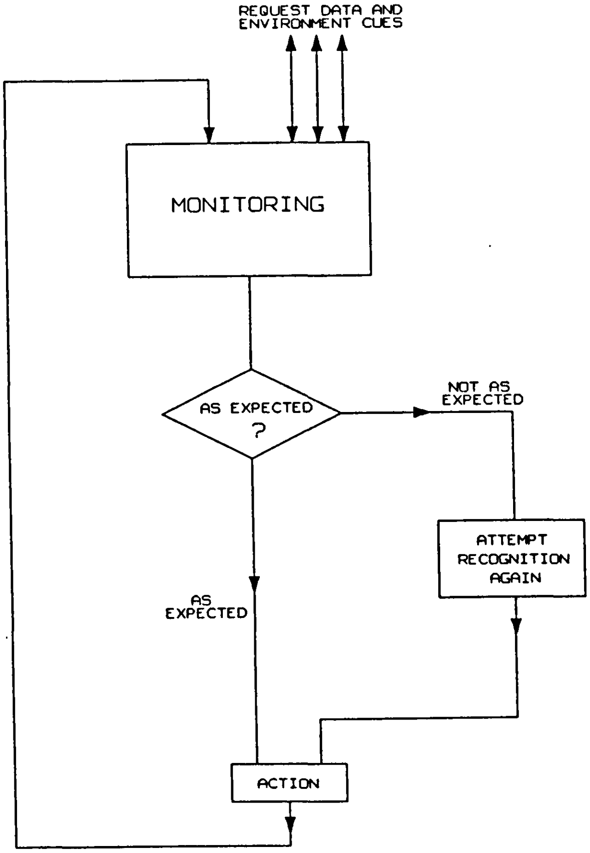


Figure 5.2b Abnormal Mode Monitoring

monitor in abnormal mode. In order to model the above task of the operator, up to the point of realising an abnormal event has taken place, two points must be considered.

First , the transition between the two monitoring modes may not be as straightforward as it is described above. From the moment the operator becomes suspicious to the time he becomes sure of an abnormality, his monitoring behaviour may be gradually changing from the direction of normal mode monitoring to abnormal mode monitoring. The transition in most circumstances will not be abrupt but for the time being, this transition between the two monitoring modes will be treated as discontinuous. This choice is made simply on the ground of simplicity rather than based on any theoretical reason. At a later date, a continuous transition can be introduced into the model but at the moment, an abrupt transition is considered to be sufficiently accurate in the model of the operator's monitoring behaviour. This continuous transition can be modelled by a gradual change in weighting assigned to each monitoring process, α, β, Γ and δ.

Second, some sort of triggering must be included in the model. For the operator to be able to "perceive" that something is wrong, without recognising the fault completely, the operator must be constantly matching the values of the parameter to some template of normality. It appears that the triggering clues that the operators actually look for are sudden changes in any parameter values beyond the permitted limits. This is particularly true when the system is in a steady state. Therefore, in the model, a threshold value can be assigned to each parameter and whenever this threshold is exceeded, a "recognition" process can be initiated. This recognition process will involve the matching of plant values stored in the state frames in the knowledge-base with the current plant values. The modelling of the recognition process will be described in more detail in later sections.

The reason for using threshold value is a simplistic one and is primarily intended for use in monitoring the system when it is in a steady state. When the system is not in a steady state, such as in start up or shut down manoeuvres, the use of threshold values is clearly not applicable because the values of the parameters are constantly changing. One suitable alternative may be to use the gradient of the change in parameters as a threshold.

5.4 Modelling the Recognition Process

Once the operator suspects that something abnormal is happening, he will try to determine whether he possesses any knowledge about that particular abnormality. The operator's attempt at recognition will involve the searching of his memory or in this case, the knowledge base.

There are four different types of knowledge concerning each system state. These include

1) The symptoms which characterise a state.
2) The appropriate response the operator must produce.
3) The expected system behaviour after the operator has taken the associated action.
4) The evidence which negates a certain system state being the probable current state.

In this model, these four types of state information are stored in four types of frame and they will be referred to as Symptom frames, Action frames, Expectation frames and Elimination frames. A State frame will consist of one of each of the above four frames.

The structures and functions of these frames will be discussed in detail in section 5.5 and only a brief description is given below.

The Symptom frames provide the values of the parameters which characterise a system state and these will be used in the initial attempt at recognising the state.

The Action frames provide information on the required actions to be performed if the system is in that particular state. The action frames as a group contains information on how a task is to be carried out and they are grouped into different levels of abstraction. All frames which are not at the lowest level of abstraction can be translated into sequences of action of the lowest level.

Expectation frames provide information on the expected behaviour of the system after a particular action is performed on the system in a particular state. These expectation values can be in the form of system parameter values or gradients of change of the system parameters. Once the operator matches the on-going symptoms of the abnormality with that of a known system state, the operator can simply follow the instructions detailed in the associated Action frame and seek confirmations from the information provided in the Expectation frame.

Elimination frames provide rules which can be used to eliminate a certain system state being valid according to the symptoms. The Elimination frames provide a mechanism which enables a coarse grain but unambiguous selection to be made. The obvious trade off is accuracy.

There are two factors which can affect the recognition process. First, the method of searching and second, the criteria for similarity, i.e., stating that there is a match. The searching method will place a restriction on the set of states to be considered by the operator and this is particularly important if the operator is performing under some time constraint. The criteria of similarity used will affect the eventual choice of frame and will subsequently influence the operator's response. As discussed in section 5.2, SR-B and KB modes will employ different searching mechanisms and similarity criteria. The differences in the modelling methods of the two modes are discussed below.

5.4.1 The Role of Similarity Criteria in Skill/Rule-based Behaviour and Knowledge-based Behaviour

Thibodeau et al (19) concluded in their experiments with physics students that the way students categorise problems in classical mechanics are related to problem solving expertise in physics. The better novice problem solver made more similarity judgements on the basis of deep structures than did poorer novice problem solvers and they also suggested that the relationship between the use of principles in categorisation and problem solving skills is also an appropriate characteristic for making distinction between "good" and "bad" physics students with similar educational experiences.

Novices, by definition, could not have acquired a deep appreciation of the subject and therefore, the novice will be engaged mostly in a skill/rule-based behaviour mode. Basically, the novices would be able to cope readily if the similarity between the problems they were assigned and the examples they were taught is very obvious. That is, the novices can cope adequately if they can recognise the problems they are assigned readily. Experts, by definition, possess deeper understanding of the subject and can perform a similarity test using deep structures.

In several theories of human information processing, a distinction between automatic and controlled (or attentional) processes has been made (20,21,22). Automatic processes are assumed to occur without attention, intention, awareness or interference with another concurrent mental activity and it is a process that cannot be easily stopped or changed. In contrast, a controlled process is claimed to be dependent on the limited attention capacity and is affected by intention and expectation (21).

Shriffrin et al (41) conclude that it is not easy to differentiate between the two types of processes, which are presumably involved in all tasks. Therefore it has been proposed that automaticity should be applied to components of behaviour rather than to behaviour as a whole. This view is reflected in this proposed modelling method for skill/rule-based behaviour and knowledge-based behaviour. In this model, monitoring and recognition are treated as combinations of automatic and controlled processes. The switch between the automatic mode and the controlled mode is well defined. In the case of monitoring, the trigger is the perception of abnormality (section 3.4) and, in the case of recognition, the trigger will be a failure in positive identification.

It is argued in this model that, in SR-B mode, similarity judgements are performed automatically and the actual comparison mechanism is opaque to the operator's consciousness. In KB mode, however, the operator will be aware of the selection criteria he uses. SR-B similarity judgement seems to conform to Wittgenstein's (23) theory of similarity and, for KB similarity judgement, a "negative" approach, namely selection by elimination, seems intuitively appropriate.

5.4.2 Criteria For Similarity in Skill/Rule-based Behaviour - Wittgenstein's Theory of Similarity

Wittgenstein's cluster account of concept similarity (23,24) is based on the observation that objects with extremely diverse properties may bear the same features. In his own words,

> "Consider for example the proceedings that we call "games". I mean board-games, Olympic games, and so on. What is common to them all? -- Don't say : "There must be something common, or they would not be called "games" ". -- but look and see whether there is anything common to all. -- For if you look at them you will see something that is to all, but similarities, relationships, and whole series of them at that. To repeat: don't think, but look! -- Look for examples at board-games, with their multifarious relationships. Now pass to card-games; here you find many correspondences with the first group, but many common features drop out, and others appear. When we pass next to ball-games, much that is common is retained, but much is lost. -- Are they all :amusing"? Compare chess with noughts and crosses. Or is there winning and losing, or competition between players? Think of patience. In ball-games there is winning and losing; but when a child throws his ball at the wall and catches it again, this feature has disappeared. Look at the parts played by skill in chess and skill in tennis. Think now of games like ring-a-ring-a-roses; here is the element of amusement, but how many other characteristic features have disappeared! And we can go through the many other groups of games in the same way; we can see how similarities crop up and disappear. And the result of this examination is: we see a complicated network of similarities overlapping and crisscrossing: sometimes overall similarities, sometimes similarities of detail." (23)

Wittgenstein's theory suggested that although we can say that two cases share certain features in common, we cannot necessarily be able to describe the relevant similarities in advance. This kind of similarity matching mechanism is akin to the one used in SR-B mode because, in both instances, the person performing the matching is not aware of the complete set of criteria used in determining similarity (note the difficulties involved in extracting the common characteristics associated with different types of "games" and this is thought provoking for the Knowledge Engineers). Nevertheless, the operator can make a similarity judgement fairly quickly, without resorting to the use of formal rules.

In addition, from the evidence found in the research into problem solving (19), SR-B similarity matching may be utilising mostly surface features and in this case the values of the system parameters. Therefore, the data structure used to represent a particular system state should include a store for the value of individual system parameters, various combinations of these parameters and their weights in their contribution to the determination of similarity. Within this formalism, two objects are "matched" if the similarity value of the test objects and the reference object is the highest amount the set of reference objects and the minimum threshold value is exceeded.

The similarity value is calculated according to the rules associated with the reference object. It is proposed that the implementation of similarity matching using Wittgenstein's concept of similarity, fuzzy logic is used because of its suitability in representing imprecise knowledge and its natural ability in the handling of weighted contributions.

5.4.3 Criteria of Similarity in Knowledge-based Behaviour - By Elimination and Minimisation of Risk

It seems intuitively that in most tests of similarity, positive evidence supporting a match will be used and matching by elimination will only be employed when it is not possible to produce a positive match with any member of the available set. When an elimination mechanism is used, the operator will, in the majority of cases, be aware of the reasons for the elimination of each candidate.

In a way, the elimination rules will be a more exact but perhaps less cautious form of the rules used in SR-B mode and in addition each rule is a conclusion. These may take the form of facts which must always be true when that particular rule is fired. The set of conclusions obtained for each of the set of reference frame can then be checked for inconsistencies using some general knowledge-base. This knowledge-base contains physical principles or other common sense knowledge which cannot be violated. Violations at this level implied inconsistencies and it may be possible for the operator to use this information to further narrow down the probable choices.

There must be an added constraint to the elimination selection rules. The operator will first of all limit the size of the reference set of frames to that of the most common or important ones. Secondly, he will also bear in mind the consequence of selecting the wrong frame.

For example, if the present state is A and he has to make a choice between three frames A, B and C which are included in the set to be searched because of their common occurrence and importance. Using the elimination rules, he managed to rule out C and also B but he then needs to consider the "cost" of his decision. Of the three states, B is a very important one and failure in responding correctly to B, such as using the recovery strategies for A and C will entail a high penalty perhaps in the form of loss revenue. The cost associated with failure to respond to A and C, i.e., responding with the recovery strategies for B is not as high and is acceptable. Then the operator will choose B. However, the operator will also register the reasons, namely the "cost" angle for this selection and should subsequent symptoms indicate the state to be otherwise, he will be able to re-examine his reasoning.

The use of a trade-off between correctness and cost is an intuitive one. Due to the nature of the processes the operator has to control, safety is in most cases the prime focus of the operator, particularly in this day of great environmental concerns. A wrong but safe decision is of more value and can in the majority of cases be easier to justify and may be preferable to the most probably right decision which carries a high cost if wrong. In addition, the operator will also have other operating directives which may be financial in nature. A wrong decision may carry a high financial penalty and could influence the operator's choice of actions.

5.5 Conceptual Architecture of the Cognitive Model

The basic architecture of this proposed model is similar to the one used by Cacciabue et al (16). The treatment of the cognitive processes such as recognition and monitoring though is different as are the mechanisms for making a selection among a set of partially matched frames.

The basic architecture comprises three major components (Figure 5.3a), the Knowledge-base , the Working Memory (WM) and the Control Centre (CC). The knowledge base, as its name suggests, contains the knowledge of the operator. The KB can be thought of as a long term memory store in which the knowledge acquired is stored and can be retrieved at most times using some retrieval mechanism. The WM represents the working area of the cognitive processes and provide a temporary storage space for intentions etc. The WM can be regarded as a kind of short term memory which is dynamic and its content is subject to changes. The control centre represents the essence of cognition. It decides on the mechanisms used for various cognitive processes, initiate searches in the knowledge base and checks for consistencies (common sense or otherwise) etc.

5.5.1 Knowledge Base

The knowledge base is basically a large frame store and there are two categories of frames (Figure 5.3a), the State frames and General Knowledge frames. The State frames contain four separate areas of knowledge about the system.

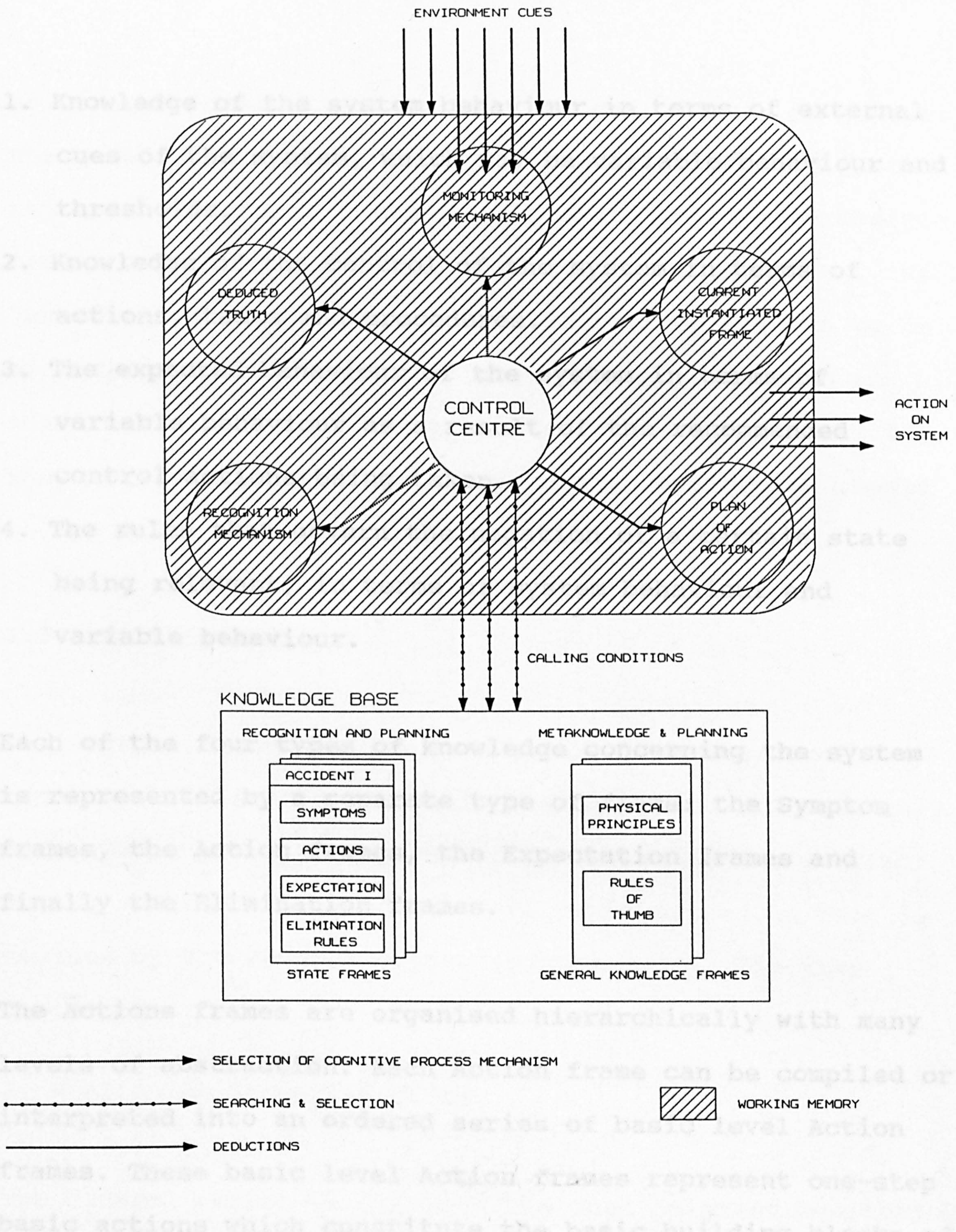


Figure 5.3a Conceptual Architecture of the Cognitive Model

1. Knowledge of the system behaviour in terms of external cues of the system, which may be variable behaviour and thresholds.
2. Knowledge of the control of the system in terms of actions, tasks and procedures.
3. The expected behaviour of the system in terms of variable behaviour as a result of the recommended control actions being taken.
4. The rules that govern the negation of a certain state being relevant, in terms of system behaviour and variable behaviour.

Each of the four types of knowledge concerning the system is represented by a separate type of frame: the Symptom frames, the Action frames, the Expectation frames and finally the Elimination frames.

The Actions frames are organised hierarchically with many levels of abstraction. Each Action frame can be compiled or interpreted into an ordered series of basic level Action frames. These basic level Action frames represent one-step basic actions which constitute the basic building blocks of human activities; in other words "demons" (10).

The General Knowledge frames contain knowledge of the system processes and the structure of the system in terms of general physical and engineering principles. In general, the knowledge captured in these frames consists of system representations with reduced complexity where intermediate causal and structural links are omitted. In addition, the "experience" of the operator will be stored here in terms of "Rules of thumb", general physics knowledge such as conservation of mass, energy and momentum and finally the knowledge of when and how these physics principles should be applied.

5.5.2 Working Memory

The working memory is the working area for the cognitive processes. It is also the main storage place for temporary data. There are five separate data stores in the working memory and the choice of the data to be placed in them is decided by the Control centre (Figure 5.3a). The five stores are:

1. Monitoring Mechanism Store.

The control centre will select the most relevant monitoring mechanism, i.e., Abnormal mode monitoring or Normal mode monitoring, in this store. The external environment cues will be filtered by the monitoring process and the monitoring mechanism employed will affect the admittance of the system parameters into the Monitoring mechanism store and the admitted set of values will be used in Recognition (Figure 5.3b).

2. Recognition Mechanism Store.

The CC will select the most appropriate Similarity matching criteria and place them in the Recognition Mechanism Store. The selected similarity criteria will be used in the State frame searching in the KB. Depending on the criteria placed in the store, the State frame will be selected using either the Symptom frame or the Elimination frame portion of the State frame. The successfully matched frame will be returned to the CC (Figure 5.3b).

3. Current Instantiated Frame Store.

The successfully matched frame will be placed in this store by the CC and the Expectation frame portion of this frame will be used in monitoring (Figure 5.3b).

4. Plan of Action Store.

The Action frame portion of the successfully matched state frame will be compiled into a series of basic action frames and this series of action frames will be stored here. These

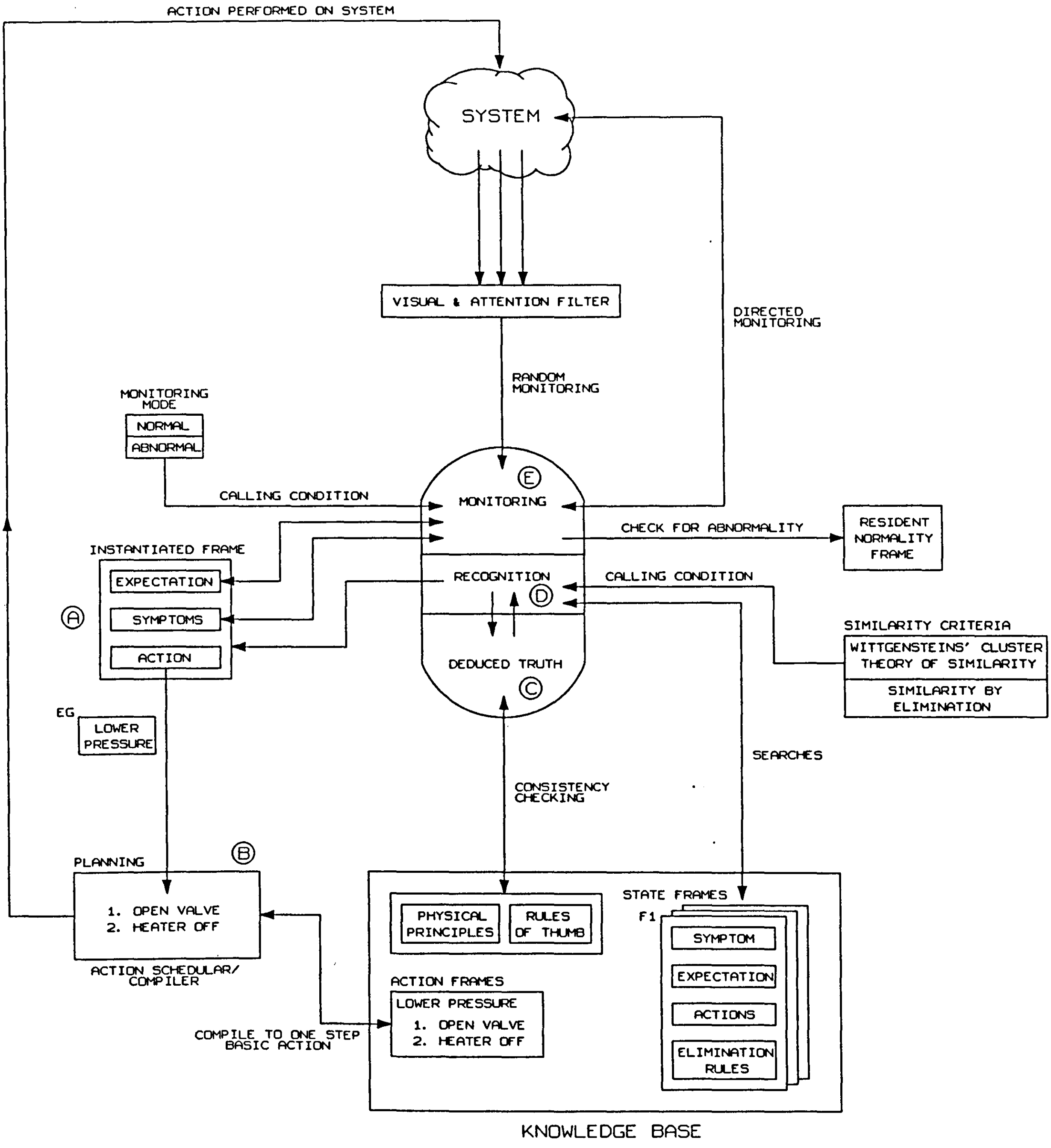


Figure 5.3b Conceptual Architecture of the Cognitive Model

actions will be taken on the system in the same order. The action sequence will be changing dynamically. When a new action is required, the sub series of basic actions associated with the new action is added to the original sequence and the order of the new sequence will be rescheduled (Figure 5.3b).

5. Deduced Truth Store.

Depending on the instantiated State frame in question, a set of truth or facts can be deduced from the system state. The deduced truth set will be placed in this store and will be used by the control centre in conjunction with the general knowledge frames for consistency checking. The result of the consistency check will in turn influence the selection of the State frame (Figure 5.3b).

5.5.3 Control Centre

The control centre is the "soul" of the model. It effectively makes all the decisions based on the results returned from the cognitive processes. It selects the mechanisms for monitoring (Abnormal and Normal Mode) and the Similarity criteria (Wittgenstein's Cluster Theory of Similarity or Similarity by Elimination) to be used in the searching of the State frames in the knowledge base. The control centre also initiates Consistency checking and searching of the knowledge base. The only mechanism for the admittance of a stored unit of knowledge from the knowledge base to the working memory is via successful matching of State frames which is in turn dependent on the similarity criteria chosen.

5.6 Conclusions

The major cognitive functions demanded of the operator are monitoring, recognition and taking remedial actions. These cognitive processes are in many ways interdependent but they also take place in parallel. The Control Centre will only determine the mechanisms to be used in the Monitoring and Recognition processes but once these mechanisms are selected, these two processes can be considered as running independently of the Control Centre. These cognitive processes do communicate with each other indirectly, via the Current Instantiated frame store and there is direct interaction between the Recognition process and the Deduced Truth Store because the data present in the Deduced Truth Store will affect the State frame selected.

The contents of the temporary data stored in the working memory will change dynamically and changes can be initiated by the environment cues (e.g., realizing something abnormal is happening will initiate a recognition process and change the mechanism used for monitoring). Therefore a blackboard computing architecture is suitable to represent the Working Memory. The five temporary data stores i.e., the working memory, Monitoring Mechanism, Recognition Mechanism, Current Instantiated Frame, Plan of Action and Deduced Truth stores can each be represented also as a blackboard with the operations and processes allowed on the contents of these blackboards clearly defined.

One of the major effects of stress is to cause a reversion from the otherwise knowledge-based mode of behaviour to SR-B mode of behaviour. In order to model the effects of stress on the cognitive behaviour of the process control operator successfully, the underlying cognitive model used must in some way be able to model both knowledge-based and skill/rule-based modes of behaviour separately. The cognitive model proposed here will be used as the basis of work on stress modelling (25) described in chapter 6.

6. Application of the Stress Modelling Method to the Cognitive Model

6.1 Representation of Decision Errors

The stress modelling method proposed in chapter 4 is not specific to any particular cognitive model but it is felt that the modified cognitive model in chapter 5 is more suitable to the representation of the cognitive effects caused by stress.

The method used for the representation of Decision errors is independent of the cognitive models used. Under normal operating circumstances, when the inference mechanisms of the operator are not affected by stress or any other factors, the difference in the supports for differenct paths at each decision node is expected to be very large and the probability for the correct path to be taken will still be very high. In other words, the decisions made by the operator is still largely indistinguishable to those made by a "perfect" operator.

When the operator experiences stress, the inference mechanism of the operator will be affected and this can affect the decisions made in two ways. First, the inference mechanism may select an entirely different path to the one selected by the unstressed one.

For example instead of producing the supports for conditions A and B as 0.91 and 0.09, which implies that path 1 is the more appropriate path, the supports produced becomes 0.4 and 0.6 which implies that B is the correct path and if a path is to be selected only on the support value received by each path, a wrong decision will have been made.

Second, the path with the highest support may remain unchanged but the level of support received is changed. For example, in the unstressed case, the supports received by A and B are 0.83 and 0.17. When the supports are combined with the selection likelihood associated with A and B respectively, ie A [.9 .1] and B [.3 .7] (Figure 4.6), the path selection likelihoods for the two paths become [.798 .202]. When the operator is stressed, the supports received by condition A and B being true become 0.65 and 0.35. When the supports are combined with the selection likelihoods, the path selection likelihoods for the two paths become [.69 .31].

When the stress modelling method is applied to the model, it may be more convenient not to include the fuzzifications of the decision paths in the first instant. This may be necessary so that changes made by the use of different and inaccurate inference mechanisms to the decision selected can be studied easily. The fuzzification of the decision paths may cause difficulties in the assessment of the effects of the changes to the inference mechanisms.

6.2 Representation of the Degradation of Inference Ability

In section 4.6.3, different methods for representing different effects on the cognitive processes have been proposed and here the direct application of these methods on the cognitive model proposed in chapter 5 will be suggested.

6.2.1 Problem Solving Rigidity

Problem solving rigidity involves the reluctance to use novel solving methods. This implies that the method selected will be biased towards the ones which are well understood and tested. There may exist various alternative methods which can be used to achieve a goal. For example, there may be several ways to reduce or maintain the steam pressure in the primary system and some of these methods may be more common than others. In the knowledge base, the actions required in "reducing the steam pressure in the primary system" will be stored in an Action frame. The actions detailed in the Action frames may in turn be detailed in other lower level Action frames until the actions described are all skill-based actions.

In order to attach a conventional metric to each plan, each Action frame must itself contain a measurement to indicate its commonness. Each method for reducing the primary steam pressure will have associated with it, the circumstances for it to be used instead of the others and a measurement of its frequency of occurrence. When the operator is functioning normally, the threshold for the tolerance of novelty is set to be high. This means when the plan of actions is thought to be appropriate, the operator will carry out that plan regardless of its novelty and the conventional metric will have no effect on the actions of the operator.

When problem solving rigidity sets in, the threshold for novelty tolerance will be decreased. If the novelties of the plan of actions is above the threshold value, an alternative plan will be generated. First, different methods with higher commonness value for achieving the same goal will be sought, and this will mean that the next most appropriate method will be used even though its calling conditions may not be satisfied entirely.

Secondly, if the novelty value for such an altered plan still exceeds the threshold value, the goal itself will have to be modified. If that proves to be impractical, then the instantiated frame will have to be changed. When the operator recognized a situation, a particular frame will be instantiated. Each frame represents a particular state of the plant and associated with each frame is a frequency or commonness value indicating the likelihood of such a fault occurring. In other words, when the operator cannot find a conventional response, he will revise his estimation of the state of the plant to something more common and produce a more conventional response.

The onset of problem solving rigidity will only affect the response of the operator when an uncommon fault occurs. Otherwise the behaviour of the operator will not be affected as most novel problem methods are associated with the more uncommon faults.

6.2.2 The Narrowing of the Attention and Perceptual Field

The narrowing of the perceptual field is closely related to the operator's ability to monitor adequately. It is proposed in the cognitive model (section 5.3.1) that there exist three separate monitoring behaviours, namely, random monitoring, directed subconscious monitoring and directed conscious monitoring.

When the operator's perceptual field becomes narrower, some of the parameters which would have been noticed normally can become ignored. When this is translated into the language of the modelling of the monitoring processes of the operator, it becomes changes in the monitoring parameters. It is assumed that the narrowing of the perceptual field mainly affects routine monitoring performed by the operator which means that both random monitoring and directed subconscious monitoring can be affected. This restriction may be spatially related, e.g.,. the operator's attention may be focused on the central portion of the control panel.

It is considered that the information specifically requested by the operator is not greatly affected. However, it is possible to argue that when the operator is under stress, he may be more ready to notice indicators which have a high visual dominance. In other words, the registration of changes in the parameters specifically requested by the operator may be more susceptible to the variation in visual dominance.

There may be some indicators whose importance is so great that any change in their value will definitely be noticed by the operator. These indicators can be programmed as "call by value" and whenever their value changes, a message is sent to the operator to cause the change to be noticed. The narrowing of the perceptual field may also cause this automatic message sending to be disrupted.

The narrowing of the perceptual field generally involves changes in the monitoring parameters α, β, Γ and δ, and the three actual probability distribution functions associated with the three separate componets of monitoring:

$$P_i^{r},\ P_i^{d^1} \text{ and } P_i^{d^2}.$$

The actual assignment of the values will depend on the physical configuration of the construction panel and the perception of the operator. These values should be obtained by close consultation with the operators.

6.2.3 Mindset

The modelling method suggested in section 4.6.3.3 can be readily applied to the present cognitive model. The deviation tolerated by the operator can be increased before the instantiated frame is rejected. In addition, the monitoring behaviour of the operator should be dominated by the directed conscious mode of monitoring.

This is because the operator is quite convinced that the instantiated frame is the correct frame and therefore will be actively seeking confirmation rather than looking for contradictions. The dominance of the directed conscious monitoring mode coupled with the increase in the tolerance of deviation will cause the operator to abandon a false hypothesis at a much later stage than otherwise.

6.2.4 Reversion to Skill-based Behaviours

When knowledge-based behaviour occurs, it usually means that some degree of independent thinking is required. The operator will have to assess the situation for himself and seek confirmation for the hypothesis he produced. The engagement in knowledge-based behaviour mode implies that something is not readily recognizable or common, because otherwise the operator would have been able to recognize the situation immediately and use the remedial methods learnt in training to cope with the situation.

The operator will have made some intermediate plans of actions to perform on the system. Some of these actions may be less frequently used or practised. Skill-based behaviours are those which are very well practised and a reversion to skill-based behaviour can be modelled by using the most similar and more frequently practised actions. In some instances, it may be that a sequence of actions which are called for are not often used and in others, it is the actual fault that is very rare. In the extreme case, the reversion to skill-based behaviour can cause the wrong but nevertheless similar plan of actions to be used instead. In this case, the monitoring behaviour will also be affected. In effect, the directed conscious monitoring will be biased towards the parameters detailed in the Expectation frames associated with the substituted frame.

It must be emphasised that the instantiated frame though should remain the one selected by the operator. Although the operator substituted his actions with a group of more familiar actions, he is essentially aware of what the on-going situation is. Therefore, at some point, the operator should revert back to the original plan of actions though it cannot be said with certainty at which point the operator will do this.

6.2.5 Experience

As mentioned in approximation (7) in section 4.5, the only effect of experience is to increase the size of the knowledge base available to the operator. This approximation is for ease of conceptualization and implementation of the cognitive model.

In this model, experience will increase the size of the knowledge base in the following ways:

1. Each subsystem may have certain physical laws which are particularly appropriate to it. For example, the pressuriser may have associated with it, the law of the conservation of mass, phase transition laws and the gas laws etc. In order to satisfy each law, the system must conform to a certain configuration or assumption. The violation of a particular law or laws will imply that a certain condition exists in the subsystem. For an experienced operator, he can use this extra knowledge to check for the integrity of the subsystem. This type of checking is particularly useful when the fault is not a standard one and conflicting information is received by the operator. An experienced operator can

use this extra knowledge to attempt to resolve the conflicts. When an the operator is an inexperienced one, such knowledge will not be available to him.

2. When the situation is not developing in ways expected by the operator, he will begin to suspect that some assumptions must be wrong. This is particularly true when there is an equipment failure. Experienced operators will have, due to past encounters, feelings, for the relative likelihood of different component failures. This information is important to the operator because the operator can examine the more likely hypothesis first and can result in savings in the time taken for diagnosis.

3. The experienced operator will know when to initiate a particular constraint checking. First for all, the operator will need to recognize the symptoms which indicate the need for constraint checking. The physical laws associated with different subsystems only serve to indicate which physical laws are appropriate to the subsystem but in themselves do not provide any indication on when these laws should be checked. On the other hand it is not possible to perform constant constraint checking as this would prevent the operator performing at a reasonable speed.

This interpretation of the role of experience carries certain implications on the implementation of the cognitive model (the Program). The implemented cognitive model will have to perform constraint checking as a matter of course.

The constraint checking method takes the following form:

1. The Program will check in the reserved experience section of the knowledge base for any particular calling conditions for constraint checking initiation. If there is none, as in the case of an inexperienced operator, no constraint checking is expected unless called for by the core knowledge base. This implies that constraint checking can still be performed as trained but the operator will not be able to start the constraint checking on his own initiative.

2. The general knowledge of physics is available to all the operators as should be the case. However, in the reserved experience section, the physical laws which are applicable to a subsystem are grouped together. When there is a perceived problem and constraint checking is called for, the experienced operator will know which are the essential laws to check for a particular subsystem and as a result will be able to diagnose the problem quicker than an inexperienced

operator, who may not have such short cuts available to him.

3. The relative likelihood for different components failures and faults will also be stored in the reserved experience section of the knowledge base.

In actual fact, the experience section of the knowledge base may remain static and contain the extra knowledge from a very experienced operator. The experience section can then be partitioned into many levels and the access to different levels of the experience knowledge base will be controlled by the experience level "clearance" of the operator (Figure 4.7). This configuration is particularly attractive because changes in the experience of the operator to be modelled will not necessitate changes in the experience section of the knowledge base.

6.2.6 Assessing Stress Levels and Associated Effects

The effects of stress on the cognitive behaviour of the operator have agreed on several aspects, as discussed in chapters 3 and 4. The findings from these effects (2,8,11,12,13,14) do not provide any indication on the relative degree of the known effects of stress.

Therefore although it is known that the operator's perceptual field will be narrowed, it is not at all certain whether this will be a dominant effect or whether this narrowing of perceptual field will always happen when the operator experiences stress.

Although cognitive effects of stress can be modelled as described in chapter 6, the relative degree of each effect is not known and as a result, in the implementation of the operator model, the relative importance of these effects will have to assigned initially. In addition, there is no indication in the findings of the stress research on the level of stress required for these cognitive effects to appear.

Before a different degree of cognitive effects can be associated with a particular stress level, the stress levels associated with different states of the plant have to be found. Generally speaking, stress measurement falls into three categories: subjective, behavioural and physiological (14).

Physiological stress measurements as its name suggests measures stress as a physiological response to stress. Different physiological parameters are closely related to the stress level experienced by an individual, namely heart rate, skill response and muscle tension etc (14). Physiological measures are difficult to fake but are more difficult to obtain during operating conditions.

Behavioural measurements of stress map the level of stress experienced with the performance of the operator and allows assessments of the ways in which performance can be degraded when the subject is placed under stress. Degradation in performance is not sufficient evidence for the presence and levels of stress though and should be used with care (14).

Finally, subjective measurements of stress rely on the perceived level of stress perceived by the operator. This measurement of stress is consistent with the definition of stress adopted in chapter 4 (section 4.1). In addition stress is a personal reaction and it is considered that subjective measurements of stress is more appropriate here.

It is proposed that a notional scale of stress levels should be used here. For each plant situation, there will be associated with it a stress level. These stress levels are to be obtained from the operator. Since the rating of different plant situations into different stress levels is effectively a discriminal process, it is proposed that Thurstone's Paired Comparison method (29 and 51) is appropriate here (Appendix A).

After a notional scale of stress level is obtained, various combinations of cognitive effects of stress can be assigned for different stress levels.

6.3 Summary

In this chapter, the application of the modelling methods for the effects of stress to the proposed cognitive model was discussed and in the next chapter, a software architecture suitable for the implementation of the cognitive model is suggested.

7. Architecture Design

7.1 Introduction

The modelling techniques proposed in chapters 4, 5 and 6 are complex. The modelling method proposed in the modelling of the behaviour of a process control operator contains many new modelling techniques. Unfortunately, it is not possible to assess the success of each new technique individually without full implementation of the cognitive model.

In this chapter, an architecture design that defines the architecture of a software system will be provided. This architecture supports a modular approach to the implementation of the cognitive model. In addition, it will support the anticipated experimentations with various model implementation techniques.

In this chapter, the functionality of each module will be defined and a top-down approach is adopted in the design. It is not considered appropriate to provide a detail implementation design here and the architecture design is complete when the system is decomposed into individual components where a detail implementation design can proceed. At this point, the relationship between each component of the system will have been provided and in some cases, algorithms will also be proposed.

The techniques used in describing the architecture design can be found in (28).

7.2 The System

Before deciding on the tools/methods used in the implementation of the system, the following points must be borne in mind.

1. Due to the complexity involved in the model, the maximum degree of freedom and control over the implementation of the model is needed. In addition, the cognitive model itself, as well as the modelling methods for the effects of stress utilise the uncertainties involved in the inference processes. Therefore, it seemed that an artificial intelligence language which supports easy handling of uncertainties should be used instead of an artificial intelligence toolkit.

2. One of the major criteria of the operator model is to be able to produce credible behaviour of a process control operator. Therefore, the implementation of the operator

model must, at the very least, be able to control a simulation of the process.

3. Finally, for the reason of assessment and validation of the model, the interaction between the operator model and the simulator should be monitored.

7.2.1 Uncertainties Handling Requirement

Bearing the above points in mind, it is suggested here that the artificial intelligence language FRIL (31) should be used. FRIL provides a sophisticated representation of uncertainties in the form of support logic programming. Support logic programming is based on fuzzy logic and FRIL allows everyday concepts such as "reasonably", "very" and "fairly" to be represented as fuzzy sets with ease. Fuzzy logic provides a useful method in handling uncertainties and with care can be very accurate in representing human knowledge. Fuzzy logic handling and fuzzy set representation are integral parts of FRIL and therefore, the amount of programming required for uncertainties handling is reduced by a large amount.

7.2.2 Controlling A Simulator

FRIL provides a very versatile foreign language interface for C, which can in turn be used to provide links with any Fortran program. This capability of FRIL means that any simulation can be "extended" or incorporated into FRIL. This means that the software version of the operator can issue a direct command to the simulator within FRIL and no hardware will be required to link the operator model and the simulator.

7.2.3 Context Diagram

The highest level description of the system is given in Figure 7.1. The architecture design starts from this diagram. The monitor component will not be included in the design because at its simplest level, the monitor needs to only log the time when events happen and all the commands issued by the operator module. The degree of monitoring functions required is very much dependent on the choice of the implementor of the model and the experimentation required.

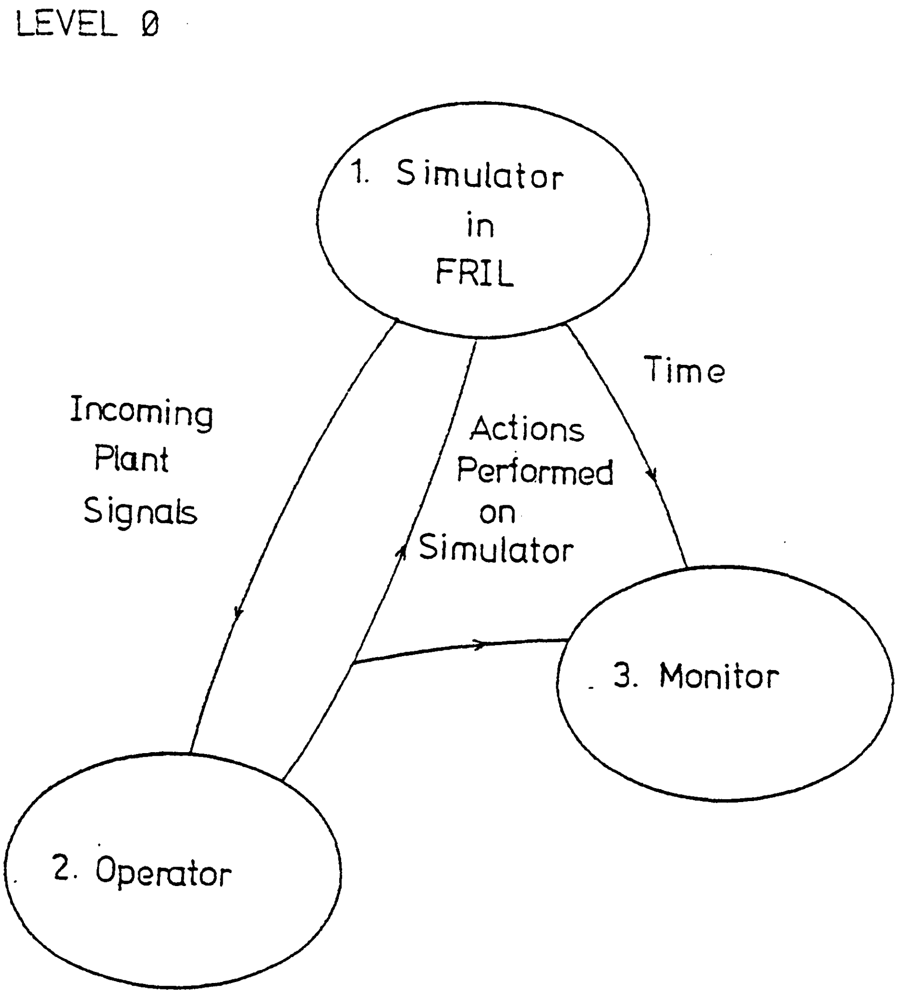


Figure 7.1 Context Diagram

7.3 Simulator in FRIL (Module 1)

The process of including a simulator in FRIL is represented in Figure 7.2. This process involves using the Foreign Language Interface provided by FRIL (31).

The simulator program, which can be in either Pascal or Fortran, is first compiled in to object code. The interface routine required by FRIL is written by the implementor in C. The interface routines are complied into object code. Finally, the file describing the mapping of the Fortran functions used to control the simulator to FRIL predicates should be provided.

The object codes of FRIL, the simulator, the interface routines and the mapping of functions and predicates are fed to the foreign language interface. The output will be a new version of FRIL with an embedded simulator (Module 1). The functions used to control the simulator can be accessed via the mapped FRIL predicates as if they were provided by the system.

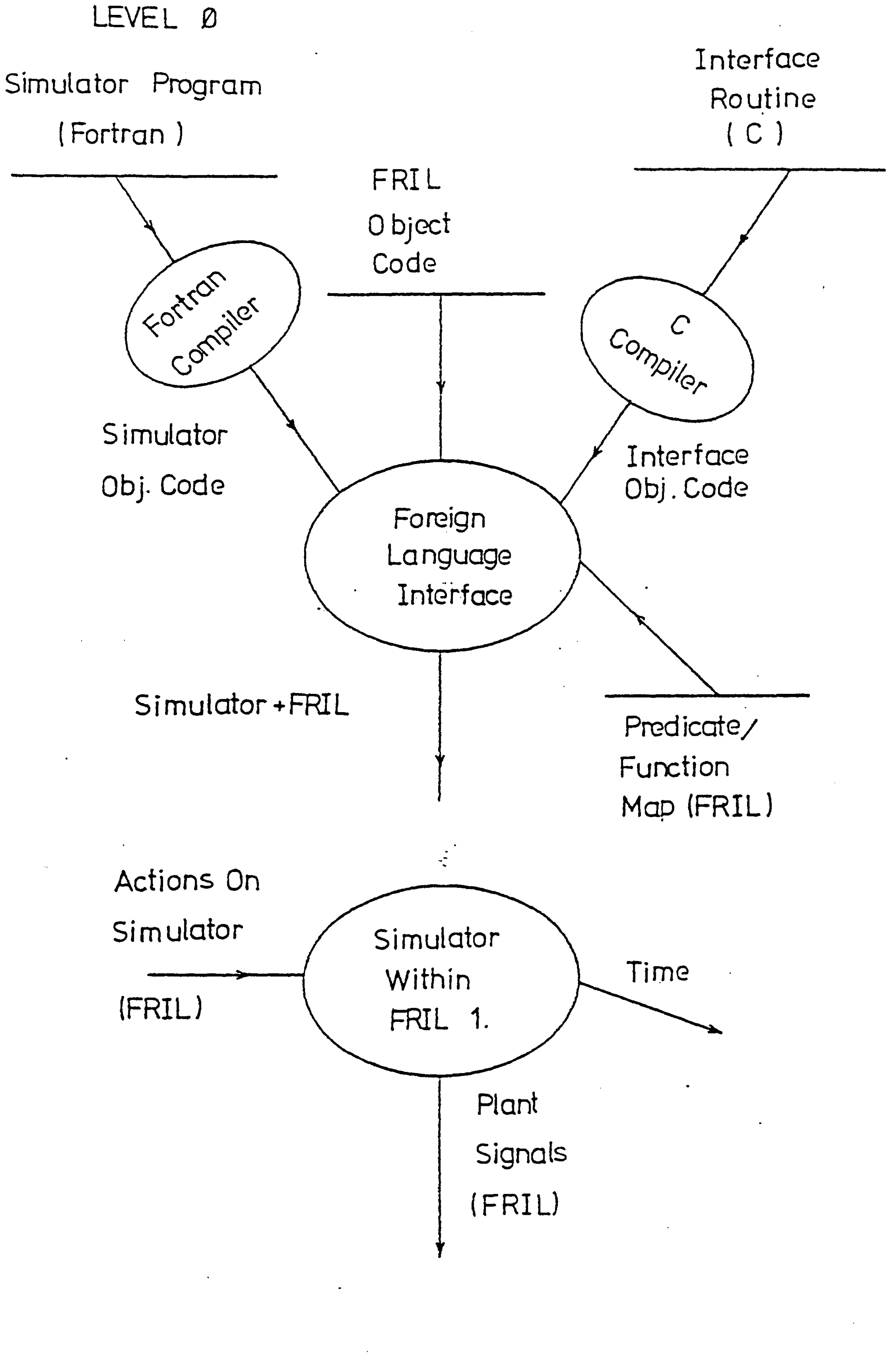


Figure 7.2 Simulator in FRIL (Module 1)

7.4 The Operator (Module 2)

The inputs to the operator module are visual and audio plant signals available to a human operator. The outputs from this module are actions to be performed on the simulator.

The operator module can be further decomposed into three modules. These are monitoring (module 2.1), recognition (module 2.2) and the get action and expectation (module 2.3) modules (Figure 7.3)

The inputs to the monitoring module are the incoming plant signals and the expectation frame. The output from the module is a current symptom frame. The function of the monitoring module is to produce a symptom frame, using the current value of the simulator parameters. The relevant expectation frame is used in performing monitoring.

This current symptom frame serves as input to the recognition module. The function of this module is to decide which state the system is in. The recognition module will use the current symptom frame, together with the recognition algorithm used at the time to make a decision. The recognition module searches through the symptom frames and elimination frames in the knowledge base in order to

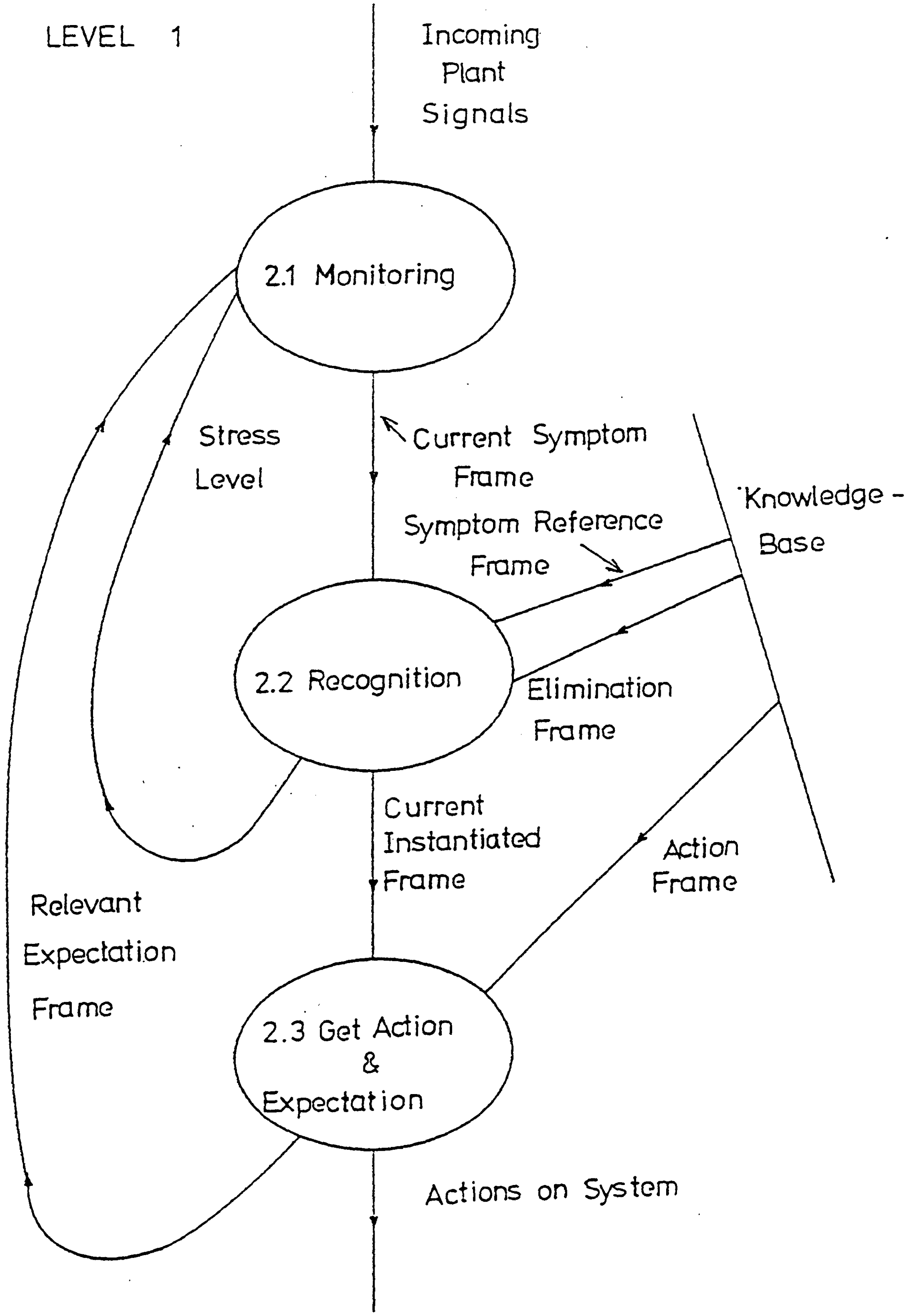


Figure 7.3 The Operator (Module 2)

make a decision on the state of the simulator. The output from the recognition module is the current instantiated frame which represents the current state of the simulator.

The current instantiated frame serves as input to the get action and expectation module. The function of this module is to retrieve from the knowledge base, the associated actions the operator needs to perform when the system is in the current instantiated frame. In addition, this module will retrieve the expectation frame associated with the current instantiated state and the expectation frame will be used by the monitoring module to perform the next monitoring operation.

7.5 The Monitoring Module (Module 2.1)

The monitoring process (Figure 7.4) can be further decomposed into the following modules, the weighted input filter (module 2.1.1), the natural language interpreter (module 2.1.2) and the frame compiler (module 2.1.3).

The function of the weighted input filter (Figure 7.5) is to select which plant signals will be registered by the operator. The actual filtering is dependent on what signal changes are expected by the operator, the stress level the

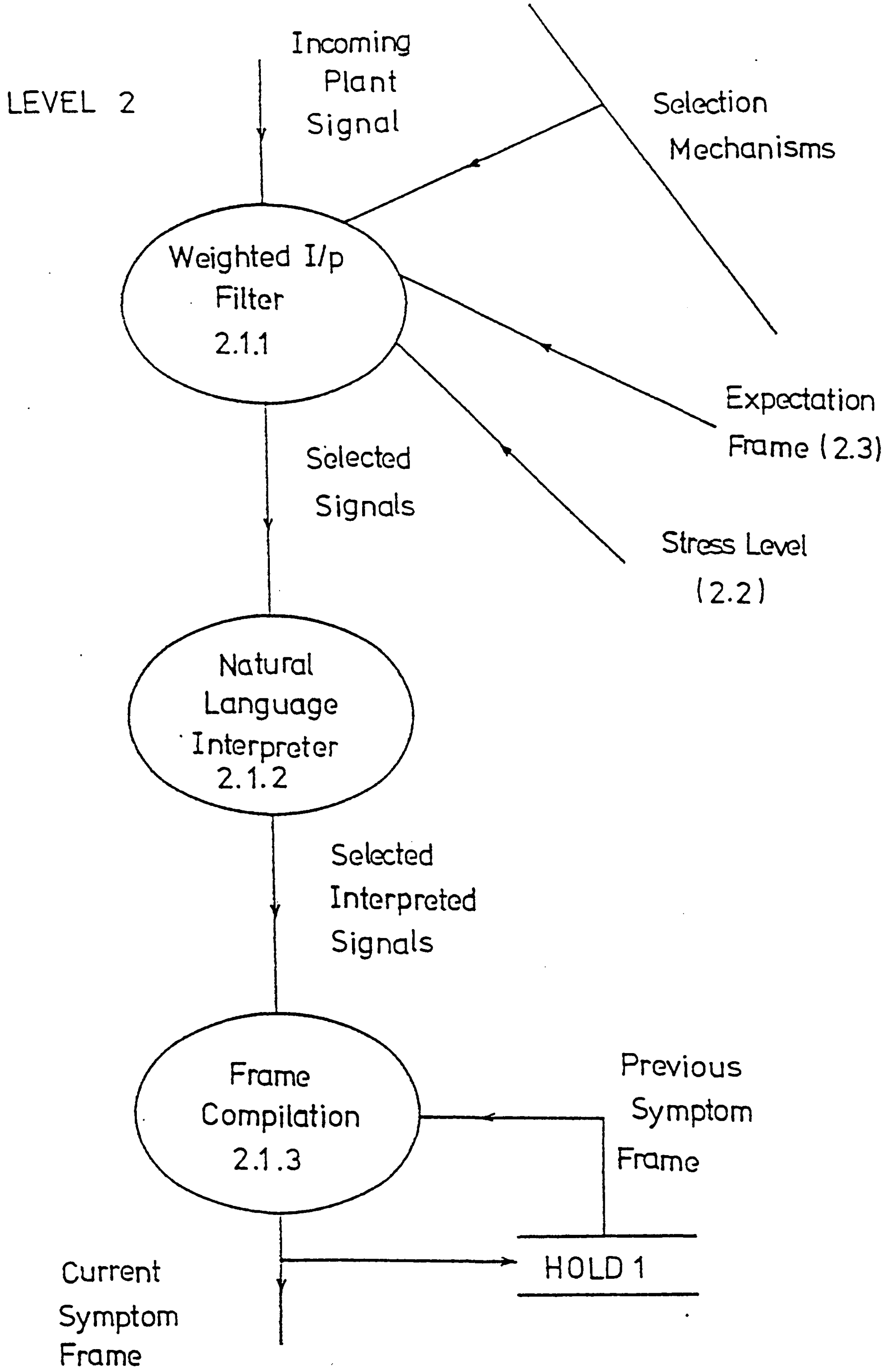


Figure 7.4 The Monitoring Module (Module 2.1)

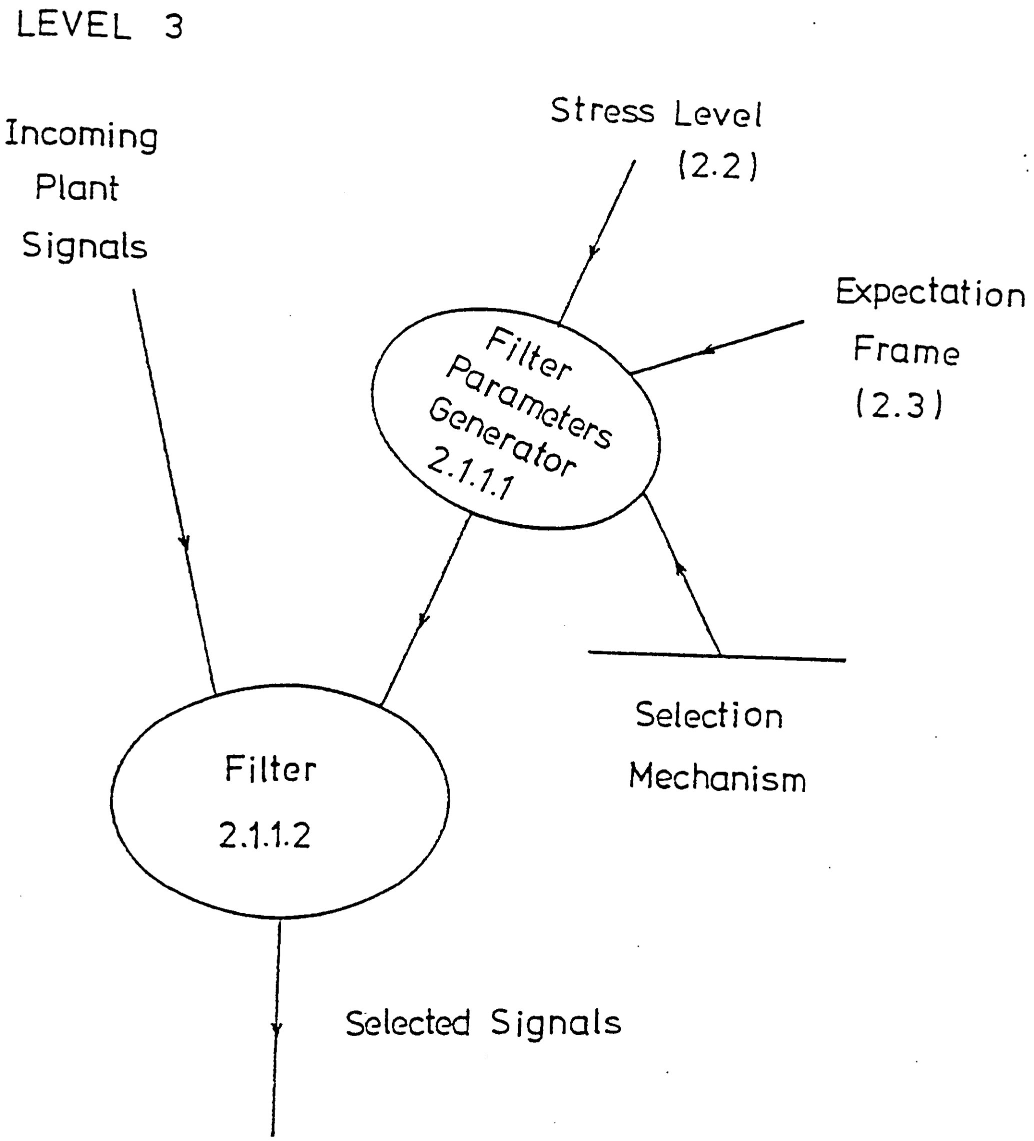


Figure 7.5 The Weighted Input Filter (Module 2.1.1)

operator is experiencing and the selection mechanism operating at the time.

The selection mechanism in operation is the mode of monitoring the operator is in, which may be a weighted mixture of the three different modes of monitoring described in section 5.3. The output from the weighted input filter is a set of simulator signals which will be registered by the operator. The current value of this set will be used with the other plant parameters to form the current symptom frame.

The function of the natural language interpreter is to convert the numerical value of the plant parameters into the natural language descriptors used by the human operator. This conversion is necessary because the knowledge obtained from the human operator and hence the knowledge in the knowledge base is in the form of a natural language descriptor. For example, the pressure may be described by the human operator as high and rising. This same form of knowledge representation will also have been used in the description of a state. The natural language interpreter will be used to convert the signals supplied by the simulator, such as 100 C/min into temperature rising very rapidly.

The frame compiler uses the interpreted new parameter values together with the previous symptom frame and produce a current symptom frame. This current symptom frame can then be used in the recognition process, using a user defined frame matching algorithm.

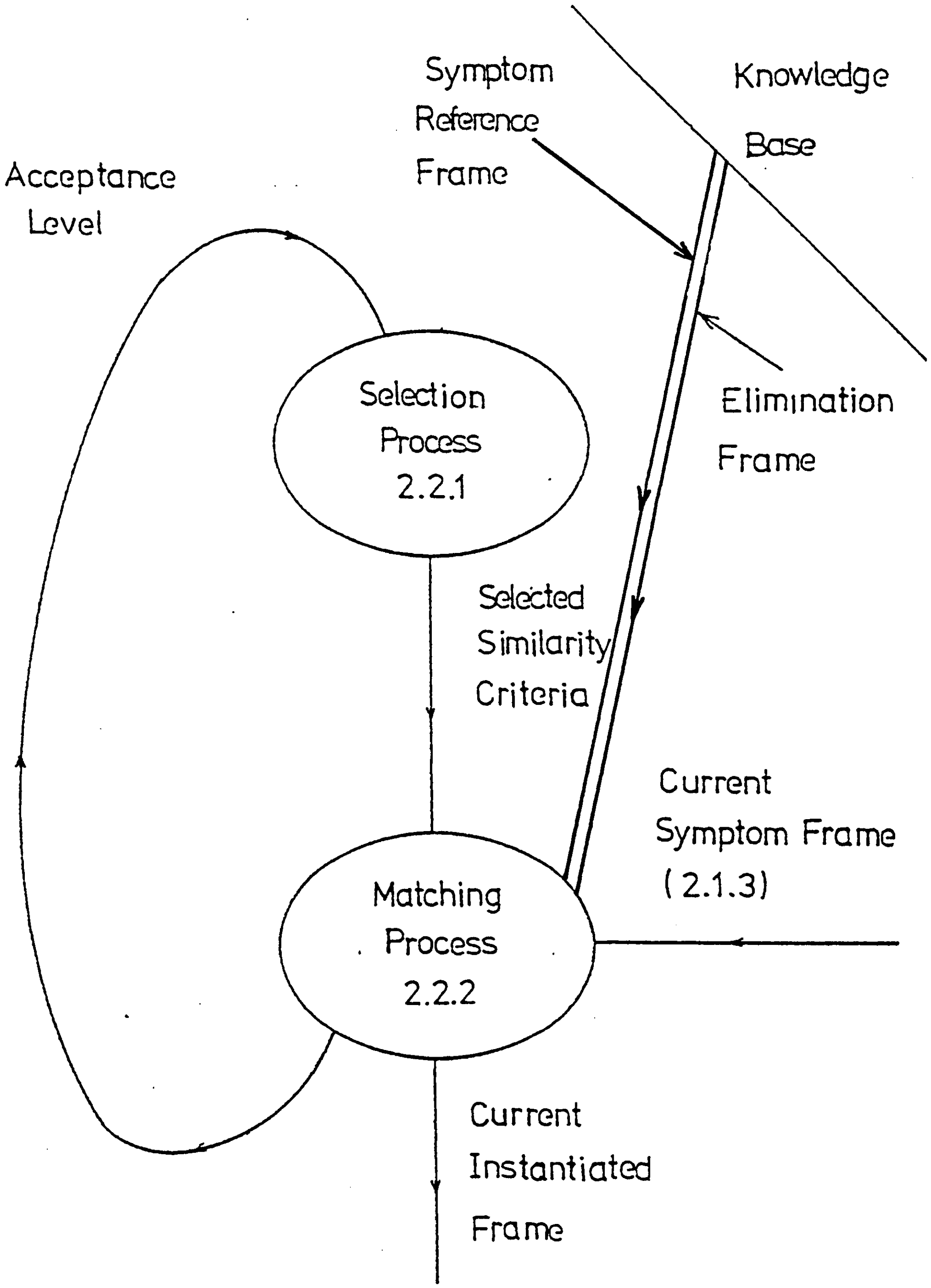
7.6 The Recognition Process (Module 2.2)

The function of the recognition module is to determine which state the plant is actually in. The input to this module is the current symptom frame. The recognition module can be further decomposed into two modules, the selection process (module 2.2.1) and the matching process (module 2.2.2).

The selection process decides which similarity matching mechanism to use on the basis of the acceptance level produced as a result of the similarity matching process. The output of this module is the selected similarity criteria that will be used by the matching process.

The matching process (module 2.2.2) will match the current symptom frame against the symptom reference frames or the elimination frames in the knowledge base, depending on the similarity criteria selected at the time.

LEVEL 2



The Recognition Module (Module 2.2)

7.7 Get Action and Expectation Module (Module 2.3)

The function of this module is to retrieve from the knowledge base, the actions the operator must perform when the plant is in a state represented by the current instantiated frame (Figure 7.6). This module can be further decomposed into the following modules, the retrieve action frame module (module 2.3.1), the retrieve expectation frame module (module 2.3.2), the high level action scheduler (module 2.3.3), the low level action compiler (module 2.3.4) and the action interpreter (module 2.3.5)

The input to the retrieve action frame module and the retrieve expectation frame module is the current instantiated frame. Both modules will search through the knowledge base and the output from these two modules will be the relevant action frame and expectation frame.

The expectation frame will be used as input to the monitoring module (module 2.1). The action frame retrieved by this module will describe actions at a higher conceptual level. For example, the action frame may contain actions such as "lower pressure level in primary circuit", or "check for constraint violation". These high level actions will serve as input to the high level action scheduler, whose function is to assign priorities to each action.

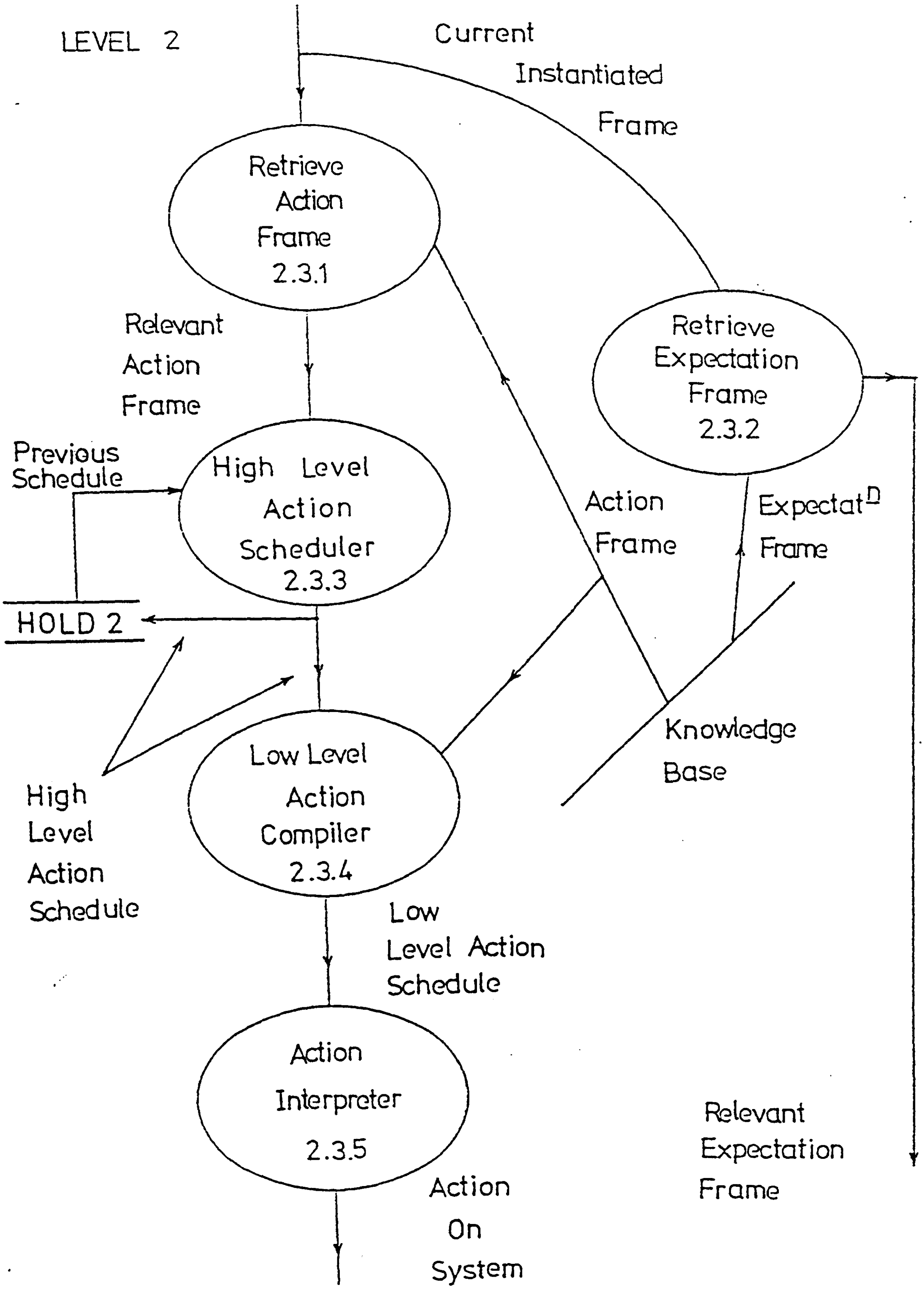


Figure 7.6 Get Action and Expectation Module (Module 2.3)

It is anticipated that the scheduling of actions performed by this module will be based on the actual rules of thumb used by the human operator. No attempt is made to utilise the current know-how in solving scheduling problems. The emphasis is on the modelling of operator behaviour and not on scheduling efficiency.

The output from the high level action scheduler is a high level schedule. This schedule serves as the input to the low level action compiler. This compiler will retrieve from the knowledge base the recipe to achieve the higher level actions.

The high level action schedule can be equated to the rule based behaviour level and the low level action schedule is equivalent to skill-based behaviour.

Finally, the action interpreter translates the actions required to perform from the language of the operator to that required by the simulator. The predicates used to control the simulator were determined in the predicate/function map used in producing the combined FRIL and Simulator Module.

7.7 Example Implementation of Similarity Matching Based on Wittgenstein's Theory of Similarity

FRIL does not provide any object oriented programming tool. In order to represent system states, different levels of action frames and symptom frames easily, object oriented programming techniques need to be used. In addition, the nature of the cognitive model requires that inheritance to be supported.

An experimental software package, which enables object-oriented programming methods to be used in FRIL has been produced. This package provides predefined data objects, such as reference frames. The package supports similarity matching for frames based on Wittgenstein's theory of similarity. Since the knowledge used in Wittgenstein's Cluster similarity matching is not visible to the operator, this package has been designed to perform knowledge hiding. The local knowledge used in this matching will be removed by the system immediately after matching is performed and as such will provide automatic maintenance of the knowledge base.

This package is called OBJ-IV and runs under FRIL (Version 4). A programming manual, source code and examples are provided in Appendix B. Examples are given in this section on possible representation techniques for the symptom frames and elimination frames only. For syntax of the package, refer to Appendix B and for a detail explanation of the meaning of the support pairs refer to (31).

7.7.1 Example Symptom Frame

Two classes are supplied by the OBJ-IV and these are ref_frame and frame. A ref_frame is specialised version of a frame and it contains special information that are used determine whether any given frame describes the entity described represented by the ref_frame.

In this example, a class called mammal is defined. A mammal is a frame and associated with this mammal frame is a set of valid attributes associated with mammals. These are intelligence, hair, size, legs, hands, appetite, nose_size and tail. Each of these attributes can take different values that are special to that particular mammal.

The class mammal will be defined as follows:

```
(isa mammal frame)

(def_obj mammal (intelligence
                 hair
                 size
                 legs
                 hands
                 appetite
                 nose_size
                 tail))
```

An additional class called mammal_ref is defined. Mammal_ref is a mammal and also a ref_frame. This means automatically that , each instance of mammal_ref contains information which characterises this particular mammal and the built-in similarity matching method can be used. The class mammal_ref is defined as:

```
(isa mammal_ref mammal ref_frame)
```

One instance of mammal_ref, human is defined. The characteristics of a human being are:

High intelligence, possesses little body hair, body size is medium (compare with all mammals), has two legs , has hands, is an ominvore, has small nose (compare to an elephant) and has no tail.

The knowledge of how an expert decides whether a mammal is human, i.e., the different similarity criteria are

1. If the mammal is highly intelligent and has no tail, it is very likely that it is a human. On the other hand if it is not then it is not likely that it is human.

2. If the mammal has two hands and two legs, and it eats both meat and vegetable, but some may be vegetarians, it is fairly likely that it is human. Otherwise, there is still a fair chance that it is human.
3. If the mammal has very little body hair, with a small body size, two legs and a small nose, there is a fairly likely that it is human. Otherwise there is fair chance that it is human.
4. If the mammal has no tail, it is very likely that it is human otherwise there is very little chance that it is human.

The above knowledge is translated and represented in the mammal_ref frame as:

```
(generate human mammal_ref
            (intelligence high)
            (hair little)
            (size medium)
            (legs two)
            (hands yes)
            (appetite omnivore)
            (nose_size small)
            (tail none)
              (mini_expert
                 (begin
                    (and intelligence tail
                      (supp ((0.8 1)(0.0 0.35))))
                    (and hands legs
                      (or appetite (appetite herbivore))
                      (supp ((0.4 1)(0.2 0.6))))
                    (and hair size legs nose_size
                      (supp ((0.4 1)(0.2 0.6))))
                    (and tail (supp ((0.8 1)(0 0.2)))))
                 end)
))
```

For any rule represented in FRIL, two support pairs can be assigned.

```
((Fact_A)
    (Fact_B)) : ((a b)(c d))
```

This rule means that if Fact_B is true, there is a minimum support of a for Fact_A being true and there is a possible support of Fact_A being true.

If Fact_B is not true, there is a minimum support of c for Fact_A being true and a possible support of d being true.

Two instances of mammal, Peter and monkey are generated. It is fairly certain Peter is highly intelligent, very certain that he has little body hair, with medium size body, two legs, a vegetarian, has a small nose and no tail.

For monkey, it is certain that he is only of medium intelligence. He has a lot of body hair, with medium body size, two legs, has hands, and eats both meat and vegetables. He has a small nose and also a tail.

Both Peter and Monkey will be represented as instances of the mammal frame as below:

```
(generate Peter mammal
    (intelligence (high supp (0.8 1)))
    (hair (little supp (0.9 1)))
    (size (medium supp (1 1)))
    (legs (two supp (1 1)))
    (hands (yes supp (1 1)))
    (appetite (herbivore supp (1 1)
    (nose_size (small supp (1 1)))
    (tail (none supp (1 1)))
)

(generate monkey mammal
    (intelligence (medium supp (1 1)))
    (hair (a_lot supp (1 !)))
    (size (medium supp (1 1)))
    (legs (two supp (1 1)))
    (hands (yes supp (1 1)))
    (appetite (omnivore supp (1 1)))
    (nose_size (small supp (1 1)))
    (tail (yes supp (1 1)))
 )
```

A built in method for ref_frame and frames can be used to decide whether Peter and monkey are human.

```
  ((compare_frame monkey human X)(pp X))
(0.4 1)
```

This means that there is a minimum of 40% support for monkey being human.

```
  ((compare_frame Peter human X)(pp X))
(0.8 1)
```

This means that there is a minimum support of 0.8 for Peter to be a human being.

It can be readily seen that the above frame representations and frame matching can be used as symptom frames and ref_frames can be used as symptom ref_frames. The criteria used for determining similar frames can be stored in the mini_expert section of the ref_frame and frame matching can be performed by using the built-in method compare_frame.

7.7.2 Example Elimination Frame

Elimination frames can be represented by using the predefined ref_frame representation. This can be achieved by using the second support pair allowed in the mini_expert section of the ref_frame.

For example, if we define a frame called mammal_elimination

```
  (generate human mammal_elimination
                  (intelligence ......)
                  ( ..................)
                      .......
                      .......
                  (mini_expert
                     (begin
                        (and (appetite omnivore)
                          (supp ((0 1)(0.0 0.3))))
                        (and tail (supp ((0 1)(0 0))))
                     end))
  ))
```

This means that if the mammal is not an omnivore, there is only a maximum possible support of 0.3 for it to be a human. If the mammal has a tail, there is only no chance for it to be a human.

7.9 Discussion

The architecture design is based on a modular approach and in some ways can be treated as a blackboard system. The two major processes performed by the operator are monitoring and recognition. Although different mechanisms for performing the two processes are used in the model, the functions of these processes remain unchanged.

The modular approach adopted in this architecture design will allow alternative mechanisms to be tried. Modifications to the implementation of a mechanism, or to the model of the underlying mechanisms used will not necessitate drastic revision to the codes.

In addition, a modular approach will also support a partitioned experience knowledge mentioned in chapter 6. A modular architecture will also enable different modelling techniques for monitoring and recognition to be experimented without requiring a drastic change to the code.

It must be emphasised that the implementation will only be a discrete version of the operator model. In the operator model, it is implied that the monitoring and recognition can be performed in parallel and in fact, this innate parallelism is hinted in some psychology literature. It is obvious that the implementation will not be able to perform different processes in a truly parallel fashion and approximations will have to be taken. The effects of such approximations should be considered after the knowledge on operating the plant is elicited and certainly before the actual implementation takes place.

8. Conclusions

The objective of this work (Section 1.2) is to produce a formal framework for the evaluation of the reliability of a system when the operator is affected by stress. It is expected that the framework will enable a computer simulation of an operator, built to cognitive specifications, to be constructed. This software can then be interfaced with a plant simulator and the whole system can be used as tool to study the overall system reliability.

The modelling methods proposed in Chapter 4, together with the cognitive model proposed in Chapter 5 together have met the objective of this research work. This research work has produced a new cognitive model that can account for the behaviour of a process control operator under stress through the amalgamation of Reason's cognitive model of error (10) and Rasmussen's process control operator model (9). This new cognitive model represents this work's contribution to knowledge. Although the example used in the development of the new cognitive model is a nuclear power plant operator, the modelling framework developed in this work is essentially generic, and can be mapped into other areas of process control, with a particular relevance to power control.

In addition, work has started towards producing the simulation tool for the study of overall system reliability. A simulator of a nuclear power plant has already been incorporated into our version of FRIL, as described in Figure 7.2. All the interfaces necessary for the software "operator" to control (via software commands) the nuclear power plant are already in place. I have implemented OBJ-IV, an object oriented extension to FRIL, to support the use of various similarity matching criteria described in Chapter 5 (Sections 5.4.2 and 5.4.3). OBJ-IV allows object oriented programming methods to be used within a fuzzy logic programming environment. I recommend that the next area for closer investigation and implementation to be qualitative modelling methods for use in the modelling of experience.

The modelling exercise has produced some solutions to the modelling of the effects of stress but it has also raised many more questions. The modelling methods proposed are primarily for a single cognitive entity. In reality, operators work together as a group in the control room. In the case of a nuclear power plant control room, a team of operators on shift duty in the control room are lead by a shift operator. The shift operator is actually in charge of the overall control of the plant. It is observed that the shift operator actually issues instructions to be carried by other operators. During operation, it is the shift supervisor who is responsible for the ultimate interpretation of the plant symptoms and is also responsible for determining actions to be performed on the plant.

In view of the general operational practice, it is reasonable to treat the shift supervisor as the operator to be modelled. The actions to be performed on the system are very often carried out by other operators. The monitoring of the system parameters are also performed by other operators. One of the major tasks of the supervisor, it is argued here, is recognition. The supervisor will delegate some lower level of responsibility such as "monitoring the steam pressure" or "maintain the water level" to other operators. Indeed, it may be possible that some degree of freedom is available to the other operators in the carrying out the instructions from the shift supervisor.

One of the important aspects concerning the behaviour of a group is the communications between individual members of the group. Depending on the effectiveness and the adequacy of the communication links, errors due to misinterpretation and misunderstanding can occur.

The implemented operator model can be modified in order to study this group behaviour. The changes involved will not be difficult to achieve for group actions to be studied.

The training of process control operators is both time consuming and expensive. Mistakes in the design of the training of the operators will be costly to remedy and indeed may be difficult, as the common saying, "old habits die hard", seems to suggest. In addition, once a cognitive demon is formed and confirmed, habit intrusions and reversion to first learnt behaviour can become a real problem.

It may be possible to implement the procedures and the reasoning required of the operator in the control of a new process control plant using the cognitive model and stress modelling methodologies proposed. This implementation can then be used to test for ambiguities in the training design and if ambiguities do exist, the training methods can be redesigned before training starts.

There remain many problems to be addressed before such a tool can become reality. The implementation of the cognitive model is by no means trivial. Some of the problems associated with the implementation of the cognitive model are closely related with the general problems of Artificial Intelligence, particularly how to model deep knowledge. There are also many problems associated with knowledge acquisition and representation in general. Finally, the last but least problem which has to solved is the validation of the model and effective validation methods have to be found and agreed upon.

References and Bibliography

1. Stephens, Mark (1980). Three Mile Island, the Hour-by-Hour Account of What Really Happened. Junction Books.
2. Rippon, Simon (1986). Chernobyl Operators Mesmerized by Mind-set. Modern Power Systems p45-63
3. Human Reliability Assessors's Guide. HRAG/UKAEA February 1988.
4. Swain, A.D. and Guttmann, H.E., "Handbook of Human Reliability Data Bank for Nuclear Power Plant Operators, Volume 1: A Review of existing Human Reliability Data Banks, "USNRC Report NUREG/CR-1880, January 1981.
5. Report of the President Commission on the Accident at Three Mile Island. Pergamon Press.
6. Hollnagel, E. (1989). Human Reliability in Knowledge-Intensive Tasks. Proceeding of the Meeting on Human Reliability in Nuclear Power. Organised by IBC Technical Services Limited.
7. Bainbridge, Lisanne (1989). Cognitive Science Approaches to Process Operation - Present Gaps and Future Requirements - A Response to the First European meeting on Cognitive Science Approaches to Process Control. Proceedings of the Second European Meeting on Cognitive Science Approaches to Process Control in Siena.

8. Beare, A.N., Dorris, R.E., Bovell, C.R., Crowe, D.S., Kozinsky, E.J., "A Simulator Based Study of Human Errors in Nuclear Power PLant Control Room Tasks." NUREG/CR-3309 January 1984.

9. Rasmussen, J. (1974). The Human Data Processor as A System Component. Bits and Pieces of A Model. Riso-M-1722. Riso National Laboratory, Roskilde, Denmark.

10. Reason, J. and Mycielska, K., (1982) "Absent Minded ? - The Psychology of Mental Lapses and Everyday Errors." Prentice Hall.

11. Becker, Gerhard (1985) Storungsimulation am Kraftwerksimulator - zum Problem der Interpretation von Operateurhandlungen und deren kognitiver Hintergrunde als Basis fur Gestaltungemfehlungen. p.137-154 In: Deutsche Gesellschaft fur Luft- und Raumfahrt e.V., Bonn (F.D.R.). Bewertung von Mensch-Maschine-Systemen - Methoden und Problematik.

12. Schnauder, H.and Smidt, D.. "The Human Factor in Operating Nuclear Power Plants During Crisis Situations."(in German) Kernforchungzentrum Karlruhe G.m.b.H., F.D.R. Institut fur Reaktorentwicklung. KFK-3234 October 1981.

13. Hancock, P.A., (1989) A Dynamic Model of Stress and Sustained Attention. Human Factors,Vol. 31, No. 5.

14. Baker, S. and Marshall, E. Stress and Control Room Operator Performance - A Summerised Literature Review.

OECD Halden Project Report No. HPR-332, December, 1987.

15. Reason, J. (1986). Recurrent Errors in Process Environments: Some Implications for the Design of Intelligent Decision Support Systems. In E. Hollnagel, G. Mancini and D.D. woods (Eds), Intellignet Decision Support In Process Environments, NATO ASI Series, Springer-Verlag, Berlin.
16. Bersini, U., Cacciabue, P.C. and Manicini, G.. (1987) "A Cognitive Model for Representing Knowledge, Intentions and Actions of Process Plant Operators", Proceedings of the First Eupropean Meeting on Cognitive Science Approaches to Process Control, Marcoussis.
17. Cacciabue, P.C., Decortis, F., Mancini, G., Masson, M. and Nordvik, J.P. (1989). A Cognitive Model in A Blackboard Architecture: Synergism of AI and Psychology. Proceedings of the Second European Meeting on Cognitive Science Approaches to Process Control, Siena.
18. Rasmussen, Jens, Lind, Morton. Coping with Complexity. June 1981. Riso-M-2293, NKA/KRU-P2(81)43.
19. Thibodeau, P., Hardiman, R.D. and Jose, P.M. (1989) The Relation Between Problem Categorization and Problem Solving Among Experts and Novices. Memory and Cognition, A Journal of the Psychonomic Society Inc.

Vol. 17 No.5 p 627 - 638.

20. Hasher, L. and Zacks, R.T. (1979) Automatic and Effortful Processes in Memory. Journal of Experimental Psychology: General, 108 p 356 - 388.
21. Schneider, W and Schriffrin, R.M. (1977). Controlled and Automatic Human Information Processing: In Dectection, Search and Attention. Psychological Review. 84 p 1 - 66.
22. Ben-Zur, Hasida (1989). Automatic and Directed Search Processes in Solving Simple Semantic-memory Problems. Memory and Cognition, A Journal of the Psychonomic Society Inc. Vol. 17 No.5 p607-616
23. Wittgenstein, 1. 1958. Philosophical Investigations. Basil Blackwell, Oxford, England.
24. Helman, H.D., Fay, Y.S. and Vredenburgh, D.W. (1988). Applications of Theories of Similarity to the Classification of Complex Records. In R.J. Uttamsingh (Ed) Proceedings of the Conference on AI and Simulation, Simulation Series Vol 20. No.1 p 7 - 11.
25. Kan, C-C.F, Roberts, P.D. and Smith, I.C. (1989). A Framework for Modelling the Behaviour of A Process Control Plant Operator Under Stress. Proceeding of the Specialists' Meeting on Artificial Intelligence in Nuclear Power Plants, IAEA, Helsinki.
26. Szabo, Zoltan. A Stressz Az Atomeremuevek Bistonsagos Uezemviteleben. Energ. Atomech. ISSN 0013-7316. Vol.

37(10), p.458-463, 1984.

27. Blackboard Systems, edited by Robert Engelmore and Tony Morgan, 1988, Addison-Wesley.

28. Tom de Marco, Structured Analysis and System Specification, 1978. Published by Yourdon inc.

29. Hunns, D.M., "Discussion Around A Human Database. An Interim Solution: The Method of Paired Comparisons." In "High Risk Safety Technology", edited by A.E. Green, J. Wiley and Son 1982, p181 - p215.

30. Object-Oriented Programming: Themes and Variations. Mark Stefik and Daniel B. Bobrow. The AI magazine, AAAI- 86 Vol. 6. No. 4.

31. FRIL Programming Manual. J.F. Baldwin, T.P. Martin and B. W. Pilsworth, FRIL Systems Limited, Bristol Information Technical Centre, St. Annes House, Bristol, BS4 4AB, United Kingdom.

32. W.F. Clocksin and C.S. Mellish. 1981. Programming In Prolog. Springer-Verlag

33. Prolog Programming for Artificial Intelligence. Ivan Bratko, 1986. Addison-Wesley.

34. POPLOG User Guide. Integral Solutions Limited, Unite 3, Campbell Court, Bramley, Basingstoke, Hampshire RG26 5EG, United Kingdom.

35. A. Ramsay and A. Sloman. 1985 Ellis Horwood. POP-11, A Practical Language For Artificial Intelligence. R. Barrett.

36. Mike Burton and Nigel Shadbolt. 1987 Addison-Wesley. POP-11, Programming For Artificial Intellignece.

37. FLAVOURS teach file in POPLOG, usepop:[pop.teach]flavours. by Mark Rubinstein 1986.

38. Reason, James (1990), University of Manchester, UK. Personal Communication,

39. Senders, John (1989), University of Maine, USA. Personal Communication,

40. Senders, John (1980), Visual Monitoring, 1980, Ph.D. Thesis, University of Maine

41. Schriffrin, R.M., Dumais, S.T. and Schneider, W. (1981). Characteristics of Automatism. In J.B. Long and A. Baddeley (Eds) Attention and Performance IX p 223 238. Hillsdale, N.J: Erlaum.

42. Building Expert Systems, edited by Frederick Hayes-Roth, Donald A. Waterman and Douglas B. Lenat. 1983, Published by Addison-Wesley.

43. Schmucker, Kurt J., Fuzzy sets, Natural Language, Computation and Risk Analysis. Computer Science Press 1984.

44. Klatzy, R.L. (1984) Memory and Awareness: An Introduction to Processing Perspective. New York: W.H. Freeman.

45. Mike Breenwell. 1988. Knowledge Engineering For Expert Systems. Ellis Horwood.

46. Alison L. Kidd. (Editor) 1987. Plenum Press. Knowledge Acquisition For Expert Systems. A Practical Handbook.

47. Herbert, Mike and Williams, Herbert An Examination of Qualitative Plant Modelling as A Basis For Knowledge-based Operator Aids in Nuclear Power Stations. Central Electricity Research Laboratories.

48. Keravnou, E. T. and Johnson, L., Intelligent Handling of Data by Integration of Common-Sense Reasoning. Knowledge-based Systems Vol.1 No. 1 December 1987 Butterworth.

49. Hannaman, G.W. and Surgin, A.J. Systematic Human Action Reliability Procedure, EPRI NP-3583, June 1984. Electric Power Research Institute, Palo Alto, California.

50. Hannaman,G.W., Spurgin, A.J. and Lukic, Y.. A Model For Assessing Human Cognitive Reliability in PRA Studies. (1985). IEEE Third Conference on Human Factors and Nuclear Power Plants, Monterey, California, June 23-27.

51. Thurstone, L.L., A Law of Comparative Judgement. Psychology Review. Vol. 34, p273 - 286. 1927.

Appendix A

The Method of Paired Comparison

This method makes use of the human judgement ability in obtaining a ranking or scaling of a given set of items with respect to a specified attribute. In this case it is the stress associated with different incidents that is being scaled.

The items are presented in all combinations of two and the subject has to decide which member of the pair presented are greater or smaller with respect to the attribures to be scaled.

For n incidents, the complete set of pairs will be ${}^{n}C_{2}$,

$$\text{Number of all possible combinations} = {}^{n}C_{2} = \frac{n(n-1)}{2}$$

In order to produce a statistically usable set of results, the entire process has to be repeated many times. The same subject can be used for each assessment. This comparison positions stimuli along a psychological continuum which is in fact the first of the postulates of Thurstone's law of comparative judgement. "Each stimulus when presented to an observer gives rise to a discriminal process which has some value on the psychological continuum of interest"

The scale obtained in this way will be a notional scale with the highest rated attributed as "1" and the lowest rated one as "0". The second postulate asserts that "because of momentary fluctuations in the organism this notional scale value will vary."

When the same stimuli are presented to the same observer a large number of times, the distribution of the scale will be approximately Gaussian in form.

The mean and standard deviation of such a distribution are termed the mean scale value, μ, and the discriminal dispersion, σ. Figure A.1.

Consider two such distributions with mean scale values μ_j and μ_k along the psychological continuum of interest. The observer will exercise a discriminal process and assign notional scale values, d_j and d_k , to the two stimuli. The difference between these two values will produce the instantaneous discriminal difference, d_{jk}.

As the difference between two Gaussian distribution is also a Gaussian distribution, the mean scale distribution is also a Gaussian distribution The mean scale separation between two stimuli, j and k, is given by

$$\mu_{kj} = \mu_k - \mu_j$$

However, as the distribution of the two discriminal process will overlap, some of the discriminal differences will be negative. This means that when the pair of stimuli (j and k) is presented to the observer a number of times, the observer will rate j as higher than k sometimes and the reverse is also true.

The proportion of occasions when the observer reverses his judgement will be proportional to the area of overlap between the two distributions. Figure A.2.

$$\mu_{kj} = \mu_k - \mu_j$$

$$\sigma_{kj} = \sqrt{\left\{(\sigma_k + \sigma_j - 2r_{jk}\sigma_j\sigma_k)\right\}}$$

(see Figure A.1)

The pairs of stimuli must be presented in such a way that the observer will not be able to remember his previous choices.

If k is rated higher than j f_k out of f times (f_k /f)when the pairs (j,k) were presented, then 1 - f_k/f is proportional to the shaded area (Figure A.1). The shaded area in fact is proportional to the probability that k is less than j and f_k/f is proportional to the probability that k is greater than j.

Probability that k is greater j is :

$$P(k>j) = \frac{1}{\sqrt{2\pi}} \int_0^\infty \frac{1}{\sigma_{kj}} \exp\left[\frac{-(\mu_{kj} - d_{kj})^2}{2\sigma_{kj}^2}\right] d(d_{kj})$$

where double subscript kj ≡ k - j

make the substitution :

$$z = \frac{d_{kj} - \mu_{kj}}{\sigma_{kj}} \quad \text{and} \quad dz = \frac{d(d_{kj})}{\sigma_{kj}}$$

$$P(k>j) = \frac{1}{\sqrt{2\pi}} \int_0^\infty \exp\left(-\tfrac{1}{2}z^2\right) dz$$

when $d_{kj} = 0$, $z = - \dfrac{\mu_{kj}}{\sigma_{kj}} \quad (= - x_{kj})$

$$\boxed{P(k>j) = \frac{1}{\sqrt{2p}} \int_{-x_{kj}}^{\infty} \exp(-\tfrac{1}{2}z^2)\,dz = \frac{1}{\sqrt{2p}} \int_{-\infty}^{x_{kj}} \exp(-\tfrac{1}{2}z^2)\,dz = \Phi(x_{kj})}$$

ie: unit, zero mean, normal distribution.

Thus, in reverse, if it is known that P(k>j) = 0.7 (say) then from a normal distribution table $x_{kj} \approx 0.52$ can be found and from which μ_{kj} can be calculated in this case, $\mu_{kj} = 0.52 \cdot \sigma_{kj}$

From basic statistics, the standard deviation of the difference distribution, σ_{kj} can be obtained from the standard deviations of the two processes, σ_j and σ_k.

$$\sigma = \sqrt{\left\{ \sigma_k^2 + \sigma_j^2 - 2r_{jk} \cdot \sigma_k \cdot \sigma_j \right\}}$$

ie: $$\mu = x_{kj} \sqrt{\left\{ \sigma_k^2 + \sigma_j^2 - 2r_{jk} \cdot \sigma_k \cdot \sigma_j \right\}} \qquad \cdots\cdots\cdots\cdots\cdots (1)$$

μ_{kj} : the mean separation along the psychological continuum for process i
and j

σ_j, g_k : the discriminal dispersion for processes j and k

x_{kj} : the unit normal deviate corresponding to the probability that k is rated higher than j

r_{jk} : the correlation coefficient for discriminal processes d_j and d_k

For n stimuli, ${}^{n}C_{2}$ equations of the form of equation (1) will be

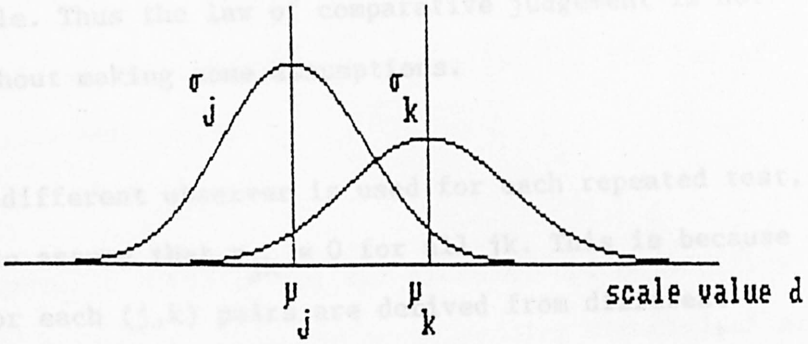


Figure A.1. Gaussian Variation of Discriminal Processes

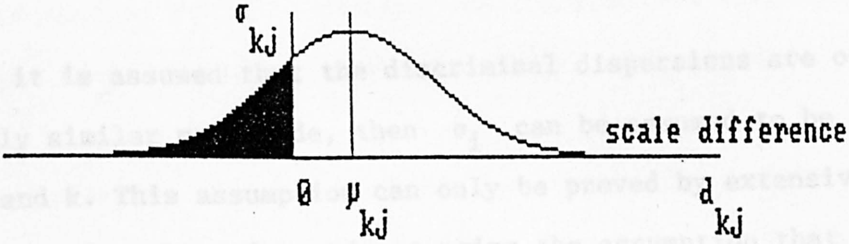


Figure A.2. The Distribution of Differences For Discriminal Process produced, ie. n(n - 1)/2. However, for n stimuli, there are (n - 1) intervals, n discriminal dispersions, and n(n - 1)/2 correlation coefficients to be found. It can be readily seen that just n(n - 1)/2 equations on their own are not sufficient to obtain the notional scale. Thus the law of comparative judgement is not solvable without making some assumptions.

First, if a different observer is used for each repeated test, it seems fair to assume that $r_{jk} \simeq 0$ for all jk. This is because the decisions for each (j,k) pairs are derived from different individual psychological continua. If the population of observers are closely influenced by each others view, such as they are all trained by the same method, then r_{jk} could be of importance. In this case, r_{jk} could perhaps be assigned a small constant value for all i and j. If the correlation coefficient is indeed virtually zero, then the equations are solvable.

Second, if it is assumed that the discriminal dispersions are of sufficiently similar magnitude, then σ_i can be assumed to be equal for all j and k. This assumption can only be proved by extensive testing of the law, by using and not using the assumption that all σ_i are equal. As would be expected, the assumption is more likely to apply when different observers are used for the repeated testings.

By using the first and second assumptions, equation(1) becomes

$$\mu_{kj} = x_{kj} \cdot \surd\sigma \qquad \ldots\ldots\ldots\ldots\ldots(2)$$

Since σ is a constant, it merely serves as a scaling factor and therefore can be ignored.

$$\therefore \mu_{kj} \equiv x_{kj} \qquad \ldots\ldots\ldots\ldots\ldots(3)$$

Equation (2) is in fact the Thurstone's case V of the law of comparative judgement (41).

If the continuum of interest is, for example, the probability of a certain event happening, then the corresponding discriminal scale obtained will be the perceived likelihood of an event happening. It is suggested by Pontecorvo that the relationship between the discriminal separation and the relative probability is logarithmic (28). It is felt that the same technique could be used to obtain the likelihood of individual scenarios happening as expected by the nuclear power plant operator (using the same ranking method), valuable information could be obtained on the possible effects of stress on the decision making mechanism of the plant operator.

It must be emphasized that for each of the comparative judgement test on a different observer, the decision consistency must be monitored (28).

As the above mentioned ranking method is psychologically based, it is not always easy to verify the scale obtained. In the case of stress, this may be possible because there exist symptoms of stress, which can be monitored. It would be useful to compare the findings using physiological criteria with the results from the comparison method. The validity of the paired comparison method can then be evaluated.

Appendix B - OBJ-IV

B.1. Introduction

B.1.1 Object Oriented Programming

Object oriented programming is to datatypes what structured programming is to procedures. In conventional programming, the emphasis is placed on procedures. and they can operate on the data and there is no provision for the data to refuse the procedure. Object oriented programming is to change the emphasis from "procedures which operate on data" to "data to which things are done". Data are organised into data objects.

These data objects can communicate with other data objects, but only via well defined channels. These objects store information about themselves and reply to messages sent to them by other objects. These messages may contain requests for information about their states or request them to change some aspects of their internal state. The store of their internal state is called attributes. There are many different implementations of object oriented programming and an introduction to the basic ideas of programming with objects is provided by (29).

B.1.2 Terminology

Object:

This term is used loosely to refer to any data object. In general, an object is supposed to represent some real world entity, at an appropriate level of abstraction. In OBJ-IV, an object denotes a class or type of objects. For example person, student, professor, valves etc.

Frame:

A frame is a data structure. A frame is consist of a predefined set of attributes which characterise an object. In many ways it can be considered as identical to an object. It is used mainly to denote a particular situation or state. For example, a mammal must suckle their young, have fur, and is warm blooded. Therefore, a frame for mammal will consist of attributes (suckle_young yes) (blood warm) and (fur present). Special provisions were made to enable frame matching to be performed and so a special format has been reserved for frames. Frames can be regarded as objects and this artificial distinction is made more on the ground of ease of programming than real distinctions.

Instance:

An instance is a representation of a particular individual entity, which belongs to a class. For example Peter is an instance, so is Mark. Both Peter and Mark belong to the object type person.

Message:

This is how the objects communicate with each other and how the outside world communicates with an object. Each object type will respond to a set of predefined messages only, and as oppose to a procedure being applied to the object, a message is sent to the object.

Method:

An object uses a method to respond to a particular message. In OBJ-IV, a distinction is made between a message which requests the information about the internal state of an object and a method to which the object can respond. The methods to which a class of object can respond is defined by the object definition. All instances of the same object type can respond to these methods.

Inheritance:

If we want to define a new type of object, called professor which is a person, it will be rather tedious to have to include all the attributes of a person into the definition of the professor. If the definition of a person changes, it means the definition of the professor will have to be changed as well. It would be very convenient if we could simply specify that a professor is a person and then add in the special features associated with the object type professor. When we want to change the definition of a person, the definition of the professor will be automatically updated. The professor will inherit the characteristics of a person.

Multiple Inheritance:

If we now have three types of objects, professor, person and french and we would like to define a new object type called french professor, it would be useful if the new class can inherit the characteristics associated with the professor and those of the french. The new object type french professor will contain all the characteristics of the object type french and professor but it will also include those features specific to the french professors. If the definition of the object type professor is changed, the characteristics of the french professor will also be updated. Inheritance can be regarded as a "spreadsheet" for programming.

B.1.3 Motivation

B.1.3.1 Modelling the Decision Processes of A Process Control Operator

Part of the needs of task 106200P of Dynamics and Control Group is to construct an expert system which models the decision processes of a process plant control operator (25).

An operator will have to make decisions constantly and the basis of his decisions is his perception of the state of the plant. The operator needs to map the current plant state to that of a familiar state and once this matching is achieved, he can select the relevant operation procedures from his memory store, ie from the manual or but he may have acquired the skill during training. He will also need to predict the effects of his actions on the plant.

An operator will therefore have within his mind a small "internal simulator". He will use this simulator to predict the outcome of his actions on the plant. However, this "internal simulator" will not be similar to the real training simulator because clearly the human mind cannot cope with the complexities on such magnitude. This "internal simulator" is mostly likely to be constructed using qualitative modelling techniques. Within the object oriented paradigm, individual parts of the plant can be represented as objects. The information regarding these objects, and their interactions can be easily encapsulated and handled. In particular, the structure of the plant can be represented hierarchially, employing different levels of abstraction.

Associated with each scenario or state, is a set of attributes. These attributes characterise a particular state. Therefore the state of a plant is best represented by a structure which will allow a set of attributes to be represented and manipulated. Some scenarios may be characterised by a particular sequence of events. A sequence of such events is called a script. Recognition of a state implies a successful matching of two frames or scripts. Due to the fuzzy nature of the real world, it is not expected that a perfect match can be achieved. It is clear that a matching mechanism which can handle fuzziness is required and it is decided that the frame matching method should take advantage of the support logic programming capabilites provided by FRIL (30) and (42).

B.1.3.2 Objective

The need to apply object oriented programming techniques within the support logic programming language FRIL became apparent (25). It was therefore decided that a software package that will enable object oriented programming techniques to be implemented with ease should be developed. This implementation, OBJ-IV is based on an object oriented programming package called FLAVOURS within the POPLOG environment (33) and FLAVOURS was written in POP11, (34), (35), (37). OBJ-IV supports multiple inheritance as in FLAVOURS and the algorithm used for selecting the most relevant method is identical to that used by FLAVOURS. Details of the inheritance protocol can be found in the teach files on FLAVOURS within POPLOG (37). Furthermore, OBJ-IV has been extended. Frames and the uncertainties associated can be represented within the support programming framework of FRIL. A similarity matching algorithm utilising support logic was developed. The extension will allow sophisticated frame representation to be made.

B.2. OBJ-IV Programming Manual

Defining a Class

```
(def_obj X (attr1 attr2 attr3 .......))
```

X is the name of the class.

Attr1, attr2 ... are the attributes associated with class X.

eg.

```
(def_obj person (sex age (health well) profession))
```

Attri can either be an attribute_name or a two member list, (name_attri P). Name_attri is the name of the attribute and P is the default value of the attribute name_attri. If a value is assigned to that attribute in an instance of the class, it will take precedence over the default value.

Defining a Method for an Object:-

```
(def_method X L)
```

X is the class_name and L is the name of the method to which the object will respond. If an object is to respond to more than one method, each method must be declared separately.

eg.

```
(def_method person birthday)
(def_method person ill)
```

Declaring an Instance

```
(generate X Y attr1 attr2 ......)
```

X is the instance_name, with class Y. Attr is a two member list, the head of which is the attribute_name and the tail of which is the attribute_value. X should not have been declared before, but instances of different class can have the same name, but check_obj must be used to select the right instance.

Updating the Instance Attributes

```
(update_obj P (X Y))
```

P is the instance_name and X is the attribute_name and Y the attribute_value. The attribute X will be assiged if it is not already done so or otherwise it will be overwritten.

Sending a Message to an Instance (to Obtain the Attribute Value)

```
(message INSTANCE_NAME ATTRIBUTE_NAME X Q)
```

The attribute_value of the attribute ATTRIBUTE_NAME will be assigned to X. If there is a specific value assigned to the attribute of the Instance, Q is spec, otherwise Q is default. If there is no specific value for the instance, then X is the default value (if one is provided) or not_defined is assigned to X. If the attribute_name is not a valid attribute for the class of instance_name, then not_valid is assigned to X.

Inheritance - Mixing Classes

```
(isa X Y Z .....)
```

To define a class X with mixed classes, the mixing must be declared before defining its special features using def_obj.

X is the class_name. Y and Z etc are the superclass of X. The class on the left takes precedence over the class on the right. All attributes associated with the superclasses Y Z will be available to class X.

Sending a Method to an Instance

```
(operate METHOD_NAME (INSTANCE_NAME INPUT) Q OUTPUT)
```

The method will be send to the instance INSTANCE_NAME. INPUT is the parameters required by the method METHOD_NAME. Q is the class whose method is chosen and OUTPUT is the output values from the method METHOD_NAME,

Generating a Method

The method must be in the form of

```
(method_name class_of_instance (instance_name INPUT) OUTPUT)
```

INPUT can be a single value or a list. INPUT is received from operate and OUTPUT will be passed to operate.

Compiling and Executing

The definitions of objects, instances and methods should be declared as follow:

```
((prog)
    (def_obj .........)
    (def_method ..........)
    (generate .........)
              .....
              .....
              .....
    (         .....       ))
```

The declared methods can be placed either before or after the declaration.

To compile the program, reload file and type: "run prog".

Predefined Objects

Two classes for the representation and comparison of frames are provided. The class ref_frame allows the attributes which are associated with a particular frame to be included and in addition, the combinations of these attributes, which characterize the frame can be represented. The class frame allows the value of each attribute of a frame to be represented. A method for the matching of these two frames is provided. A support for the degree of match between a frame and the ref_frame will be given as a result of the call of the matching function. However, the attributes of which are associated with the ref_frame and the test frame must be identical otherwise type mismatch will result.

Ref_frame

Ref_frame consists of two fields, attr_list and mini_expert. Attr_list contains all the attribute names which are associated with the ref_frame and each of the attribute defined in attr_list must be defined in the instance (of class ref_frame). Mini_expert contains the combinations of the attributes which characterize the ref_frame.

```
           eg. (generate ref ref_frame
                          (attr_list (attr_1 attr_2 attr_3 attr_i ...))
                          (attr_i value_i)

                              ...............
all the attributes
defined in attr_              ...............
list must be
represented here              ...............
                         (mini_expert
                           (begin
                          ┌─   (and attr_i ... <compl_1><compl_i> ...
                          │               (supp (<n1> <p1>)))
<<clause>> of form:       │    (or <i> <j>... <comp2_1><comp2_2> ...
                          │               (supp (<n2> <p2>)))
                          └─   ((not <j>) (supp (<n3> <p3>)))
                           end)
                         )))
```

<i>   : integer

<compl_i> : compound clause, of the following form only :-

(or attr_<j> (attr_<k> value_new)... (not attr_<p>)...attr_<q>)

if a special value, other than the one defined (value_<k>) is required, (attr_<k> value_new) should be used. If attr_<k> only is used, value_<k> is assumed.

<comp2_i> : compound clause, of the following form only :-
(not attr_<s>)

<nl> : necessary support
<pl> : possible support

Bold words are built in words.

Frame

Frame consists of field attr_list. Attr_list contains all the attribute names which are associated with the frame and each of the attribute defined in attr_list must be defined in the instance (of class ref_frame).

```
        eg. (generate frame_name frame
                    (attr_list (attr_1 attr_2 attr_3 attr_i ...))
                    (attr_i (value1 supp ((<nl> <pl>)))

                         ...............
all the attributes
defined in attr_         ...............
list must be
represented here         ...............  )
```

<i> : integer
<nl> : necessary support
<pl> : possible support

Bold words are built in words.

To Compare Two Frames

```
((compare_frame FRAME_NAME REF_NAME S))
```

compares two frames, frame_name which is of class frame and ref_name which is of class ref_frame. S is the support for the match. The knowledge base created is erased once the support is calculated. The decision can be traced if required. An option, to trace or not is provided at run time. The support logic shell fs.frl must be already loaded in the system.

B.3. Examples

B.3.1 Example 1

```
((prog0)
   (def_obj person (name age sex))  % object type person is defined
   (def_method person birthday)     % object person can respond to
                                    % method birthday and
                                    % print_self
   (def_method person print_self)
   (isa professor person)           % professor is a person, ie
                                    % professor possesses all the
                                    % characteristics of a person
   (def_obj professor (telefon_no   % professor has certain extra
                    inaugural_lecture_subj % characteristics
                    subject))
   (def_method professor write_paper) % professors can respond to
                                      % the message (request) to
                                      % write_paper
   (def_method professor print_self)  % professors can respond to
                                      % the message print_self
                                      % note that print_self is
                                      % defined for both person and
                                      % professor. However, only the
                                      % most specific method will
                                      % be used.

    (generate peter person            % an instance of person is
                                      % created. The instance name
                                      % is peter but the value fo
                                      % the attribute name of
               (age 22)               % instance peter is
                                      % Peter_Townsley
               (sex male))

    (generate roberts professor % an instance peter of professor is
                                % created
           (name Peter_Roberts) % although attribute name is not
                                % defined in professor, it is valid
                                % because it is declared in the
                                % object definition of person and
                                % Professor is a person.
           (age 55_only_a_guess_sorry_prof)(sex male)
           (telefon_no xxxxx)  % special characteristics associated
                               % with professor only.
           (inaugural_lecture_subj Hierarchical_control_system)
           (subject control_engineering)) )

 ((print_self person (X _) Z)  % methods for object type person
    (message X name Z _)
    (p 'I am' X)
    (pp))
```

```
((birthday person (X _) Z)
    (message X age Z _)
    (p Happy birthday.)
    (pp)
    (sum Z 1 Y)
    (update_obj X (age Y)) )

((print_self professor (X _) _) % a more specific for print_self
                                % for professor only
    (message X name A _)
    (message X subject B _)
    (p My name is A and 'I' am professor of B)
    (pp))

((write_paper professor (X _) Z) % methods for professor only
    (message X inaugural_lecture_subj Z _)
    (p 'I will write a paper on' Z)(pp))
```

When this example is loaded, the information for the instances peter and roberts can be interrogated. Bold type represent outputs from program.

```
?((message peter name X _)(pp X))

Peter_Townsley

?((message peter age X _)(pp X))

22

?((update_obj peter (age 21))(message peter age X _)(pp X))

21

?((operate birthday (peter _) _ _))

Happy Birthday.

?((message peter age X _)(pp X))
22

?((operate print_self (peter _) _ _))

My name is peter

?((operate print_self (roberts _) _ _))

My name is Peter_Roberts and I am professor of control_engineering

?((delcl print_self 2))  % ie the method of print_self for
                         % professor is deleted

?((operate print_self (roberts _) _ _))

My name is roberts.
```

B.3.2 Example 2

```
((progl)
     (isa spec_frame frame) % create object type spec_frame, which
                            % is a frame

     (def_obj spec_frame (attrl attr2 attr3 attr4 attr5))

     (isa ref_ref_frame spec_frame ref_frame) % create object
                                              % ref_ref_frame

     (generate spec spec_frame % define an instance of object type
                               % spec_frame

                 (attrl (valuel supp (0.9 1)))    % these attribute
                 (attr2 (value2 supp (0.8 0.9)))  % values are used
                 (attr3 (value2 supp (0.4 0.6)))  % as facts in
                 (attr4 (value3 supp (0.5 0.7)))  % local expert system
                 (attr5 (value4 supp (0.2 0.5)))  % generated for
                                                  % identification
                                                  % purpose

     (generate ref ref_ref_frame % define instance of object type
                                 % ref_ref_frame
                    (attrl valuel)
                    (attr2 value2)
                    (attr3 value3)
                    (attr4 value4)
                    (attr5 value5)
          (mini_expert  % a frame is ref if the following rule
                         % governing identification is met
                  (begin
                      (and attr3 attr2 (supp (0.8 0.9)))
                      (and (or (attrl hello) attr2) attr4
                           (supp (0.6 0.9)))
                      (or attrl attr4 (supp (0.5 0.7)))
                      (and (or (not attr3) attrl) attr2
                           (supp (0.4 0.7)))
                      ((not attr5) (supp (0.6 1)))
                   end)
          )
    ))
```

B.3.3 Example 3

```
% Example of how different natural language descriptors can be
% modelled using fuzzy sets

/*
(high [0.4:0 0.5:0.4 0.8:1 1:1 1.01:0])
(low [-0.99:0 0:0.5 0.2:1 0.45:1 0.5:0.4 0.6:0.1 0.7:0])
(little [-0.99:0 0.3:1 0.5:0.3 0.6:0 0.7:0])
(a_lot [0.3:0 0.6:0.5 0.7:0.9 0.8:1 1:1 1.01:0])
(two [0:0 2:1 2.01:0])
(none [-0.99:0 0:1 0.1:0 1:0])
(yes [0.99:0 1:1 1.01:0])
*/

((prog)
    (isa mammal frame)  % create object type mammal, which is a
                        % frame

    (def_obj mammal (intelligence hair size % define special
                     legs hands appetite    % features particular
                     nose_size tail))       % to mammal

    (isa mammal_ref mammal ref_frame) % mammal_ref is a mammal
                                      % and a ref_frame

    (generate human mammal_ref  % define an instance of object
                                % type mammal_ref, called human

         (intelligence high) % this is the set of attributes
         (hair little)       % by which a typical mammal is
         (size medium)       % defined and the values of these
         (legs two)          % attributes are used in the local
         (hands yes)         % expert system generated for frame
         (appetite omnivore) % matching purpose
         (nose_size small)
         (tail none)
         (mini_expert
            (begin
               (and intelligence tail
                (supp ((0.8 1)(0.0 0.35))))
               (and hands
                legs (or appetite (appetite herbivore) )
                (supp ((0.4 1)(0.2 0.6))))
               (and hair size legs nose_size
                (supp ((0.4 1)(0.2 0.6))))
               (and tail (supp ((0.8 1)(0.0 0.2))))
             end)
    ))

    (generate Peter mammal  % define an instance of object
                            % Peter, which is a mammal

         (intelligence (high supp (0.8 1))) % these attribute
         (hair (little supp (0.9 1)))       % values are used as
         (size (medium supp (1 1)))         % facts in the local
         (legs (two supp (1 1)))            % expert system
         (hands (yes supp (1 1)))           % generated for
         (appetite (herbivore supp (1 1)))  % identification
         (nose_size (small supp (1 1)))     % purpose
         (tail (none supp (1 1)))
```

```
    )

(generate monkey mammal % define an instance of object monkey,
                        % which is a mammal
        (intelligence (medium supp (1 1)))
        (hair (a_lot supp (1 1)))
        (size (medium supp (1 1)))
        (legs (two supp (1 1)))
        (hands (yes supp (1 1)))
        (appetite (omnivore supp (1 1)))
        (nose_size (small supp (1 1)))
        (tail (yes supp (1 1)))
))
```

When this example is loaded, the information for the references human, and instances Peter and monkey can be interrogated. Both instances, Peter and monkey can be compared with the reference human and a degree of support for each instance being a good match for the "concept will be provided.

Bold type represents outputs from program.

```
?((run prog))

yes

?((compare_frame monkey human X)(pp X))

,,bl 2
Investigate matching criteria - (y/n)? n
(0.4 1)
```

There is a miniimun support of 0.4 for a monkey to be a human.

```
?((compare_frame Peter human X)(pp X))

Investigate matching criteria - (y/n)? n
(0.8 1)
```

There is a minimun support of 0.8 for Peter to be a human.

```
?((compare_frame Peter hman X)(pp X))

Investigate matching criteria - (y/n)? y  % using the FRIL
                                          % support shell for
                                          % checking of
                                          % inferencing
to return to program, enter b at ?
Current goal is ($human match)  % a frame is compared with
                                % a reference frame, a local
                                % expert system, for the question
                                % ($FRAME_NAME match) is created
Shell level ($human)
Option ? (h for help) : s (see FRIL manual or detail)
Clause 1 of rule $human is:     % first way of defning a human

(($human match)
     ($intelligence high)
```

```
    ($tail none)) : ((0.8 1) (0 0.35))

Inferences from body of clause 1 of $human are:

    (($intelligence high)
        ($tail none)) : (0.8 1)

Inerences from clause 1 of $human are:

    ($huamn match) : (0.64 1)  % support from usiing the first
                               % viewpoint of "human"

Clause 2 of rule $human is:  % second defintiion of human

(($human match)
    ($hands yes)
    ($legs two)
    (or ($appetite ominvore)
        ($appetite herbivore))) : ((0.4 1) (0.2 0.6))

Inferences from body of clause 2 of $human are:

    (($hands yes)
     ($legs two)
     (or ($appetite ominvore)
         ($appetite herbivore))) : (1 1)

Inferences from clause 2 of $human are :

    ($human match) : (0.4 1)  % support from using the second
                              % defintion

Clause 3 of rule $human is: % third definition for human

(($human match)
    ($hair little)
    ($size medium)
    ($legs two)
    ($nose_size small)) : ((0.4 1) (0.2 0.6))

Inferences from body of clause 3 of $human are :

   (($hair little)
    ($size medium)
    ($legs two)
    ($nose_size small)) L (0.9 1)

Inferences from clause 3 of $human are:

   ($human match) : (0.38 1) % support from using the third
                             % definition

Clause 4 of rule $human is : % fourth defintion for human

(($human match)
   (tail none)) : (0.8 1) (0 0.2))

Inferences from body of clause 4 of $human are:

   ($tail none) : (1 1)
```

```
Inferences from clause 4 of $human are :

  ($human match): (0.8 1)  % support from using the fourth
                           % definition


Shell level ($human)
Option ? (h for help) : b
Completed nvestigations on goal ($human match)
Backing up a level:
(0.8 1)  % the final support from combining supports obtained
         % through different defintions for human
yes
```

B.4. Program Listings

B.4.1 Program OBJ-IV.frl

```
?((lload (object frame2  fs ))
  (p The 'OBJ-IV' is now loaded)(pp)
  (pp)
  (p There is a demonstration program, called Example1.frl)(pp)
  (p To load, type "load example1")(pp)
  (pp)
  (pp)
  (p For a demonstration on "OOP" , type "run prog0")
  (pp)
  (p For a demonstration on frame representation and frame
     matching,)
  (pp)
  (p type "run prog1" and follow instructions in the documentation)
  (pp)
  (p "OBJ-IV")(pp))
```

B.4.2 Program Object.frl - This is the core program.

```
*************************************************************
*                                                           *
*  Pstart is the Initialising Procedure. If yes, fs.frl and *
*  list_proc.frl will be reloaded.                          *
*                                                           *
*************************************************************
*/

((pstart)
      (p reload file "ownsys.frl" - "y/n" ?)
      (flush stdin)
      (getb stdin QQ)
      (if (eq QQ 121)
          ((reload ownsys ))
          ((dum)) ))

?((pstart))




/*
((root def (level) ))
((root method (printself) ))

((printself root)
    (p "<" root ">" "(level 0)"))
*/

/*
```

```
****************************************************************
*                                                              *
* Def_obj declares a class X, with allowed attributes L. L is  *
* a list of the attribute names. Default values can be         *
* embedded. (see related document on syntax). It creates an    *
* two predicates: (X def....) and (X precedence....).          *
*                                                              *
****************************************************************
*/

((def_obj X L)
    (addcl ((odict X)) )
    (addcl ((X def L)) )
    (if (cl ((X precedence X|Q)) )
        ((dum))
        ((if (cl ((X precedence|R)) )
             ((delcl ((X precedence|R)) )
              (addcl ((X precedence X|R)) ))
             ((addcl ((X precedence X)) )) )) ))


((dum))

/*
****************************************************************
*                                                              *
* Generate creates an instance X of class Y, with attributes   *
* Z. If no specific attribute is given, instance can assume    *
* the default value of the class Y. It generates internal Fril *
* clause (casel X Y.....)                                      *
*                                                              *
****************************************************************
*/

((generate X Y|Z)
    (append (casel X Y) Z P)
    (addcl (P)) )

/*
****************************************************************
*                                                              *
* check_obj gives the class (type) of the instance X as L.     *
*                                                              *
****************************************************************
*/

((check_obj X L)
    (casel X L|P))

/*
****************************************************************
*                                                              *
* Update_obj changes the associated value of attribute X to Y. *
* It can only change one attribute at a time. It operates on   *
* the internal clause (casel X ......). It checks that the     *
* attribute X is a valid one as defined in the class, internal *
* clause (X def...). If no specific value is assigned to the   *
* attribute, then it is appended to the internal clause (case) *
*                                                              *
****************************************************************
*/
```

```
((update_obj P (X Y) )
    (casel  P Q |R)
    (show_fields Q Z)
    (orr ((member X Z))
         ((member (X YY) Z)) )
    (orr ((delete (X T) R R1)
          (append ((X Y)) R1 R2))
         ((append ((X Y)) R R2)) )
    (delcl ((casel P Q|R)) )
    (addcl ((casel P Q|R2)) ))

/*
****************************************************************
*                                                              *
* Def_method defines the allowed method for class OBJ. It gen- *
* erates an internal clause (OBJ methods (....)).              *
*                                                              *
****************************************************************
*/

((def_method OBJ METHOD)
    (orr ((cl ((OBJ methods X)) )
          (member METHOD X))
         ((cl ((OBJ methods X)) )
          (append X (METHOD) Y)
          (delcl ((OBJ methods X)) )
          (addcl ((OBJ methods Y)) ))
         ((addcl ((OBJ methods (METHOD) )) )) ))

/*
****************************************************************
*                                                              *
* Operate means to execute the METHOD. It checks the precedence*
* list, and select the most specific method. The method is     *
* operated on instance CASE, with P as input to method and S   *
* as output. Q is the class whose method is chosen.            *
*                                                              *
****************************************************************
*/

((operate METHOD (CASE P) Q S)
    (casel CASE OBJ|Z)
    (OBJ precedence|Y)
    (choose_method METHOD Y Q (CASE P) S))

/*
****************************************************************
*                                                              *
* Choose_method selects the most specific method to use. It    *
* checks the precedence list and fires the method with the     *
* higher precedence. Method must be defined as                 *
* (METHOD CLASS (INSTANCE INPUT) OUTPUT)                       *
*                                                              *
****************************************************************
*/

((choose_method METHOD () no_defined_method (CASE P) undefined))
((choose_method METHOD Y Q (CASE P) S)
    (eq Y (A|B) )
    (orr ((A methods L)
```

```
          (member METHOD L)
          (METHOD A (CASE P) S)
          (eq A Q))
         ((choose_method METHOD B Q (CASE P) S)) ) )


/*
***************************************************************
*                                                             *
* Message returns the attribute value associated with the     *
* attribute name. It also indicates whether the attribute     *
* value returned is a value specifically assigned to the      *
* Instance or a default value to the class.                   *
*                                                             *
***************************************************************
*/

((message CASE FIELD X spec)
     (case1 CASE OBJ|Z)
     (member (FIELD X) Z))
((message CASE FIELD X default)
     (case1 CASE OBJ|Z)
     (show_fields OBJ L)
     (orr ((member (FIELD X) L))
          ((member FIELD L)
           (eq X unknown))
          ((eq X not_valid)) ))

/*
((isa X Y)
      (addcl ((odict X)) )
      (orr ((cl ((X precedence|Q)) )
            (p X is already defined as Q))
           ((cl ((Y precedence|Z)) )
            (append (X precedence X) Z Z1)
            (addcl (Z1) )) ))
*/

/*
***************************************************************
*                                                             *
* Isa defines a class X, which is a subclass of Y etc. It     *
* creates a updated precedence list, using the priority of    *
* left up to joint strategy as used in FLAVOURS in POP11.     *
*                                                             *
***************************************************************
*/

((isa X|R)
      (eq R (A|B))
      (initial (A|B) LA)
      (make_precedence B LA LB Y)
      (append (X precedence) Y K)
      (addcl (K) ))

((initial (A|B) LA)
      (A precedence|LA))

/*
```

```
**************************************************************
*                                                            *
* Make_precedence creats the precedence list due to the isa  *
* declaration. It locates the first common class in the      *
* hierarchy list, and then partition the lists into two      *
* sections. They are then appended and a new precedence list *
* is produced.                                               *
*                                                            *
**************************************************************
*/

((make_precedence () LA _ LA))
((make_precedence (B|C) LA LB LF3)
      (B precedence|LB)
      (member XX LA)
      (member XX LB)
      (!)
      (partition XX LA LA1 LA2)
      (partition XX LB LB1 LB2)
      (append LA1 LB1 LF1)
      (append LF1 LA2 LF2)
      (make_precedence C LF2 Z LF3))
((make_precedence (B|C) LA LB LF3)
      (B precedence|LB)
      (append LA LB LF2)
      (make_precedence C LF2 Z LF3))

/*
**************************************************************
*                                                            *
* Partition, part and rappend deals divides the list into    *
* two separate parts where there is a common class. Since    *
* append must have the first list instantiated, rappend is   *
* written to allow the first part of the list to be joined   *
* on to give a new list.                                     *
*                                                            *
**************************************************************
*/

((partition X (A|B) L1 L2)
      (part X (A|B) L2)
      (rappend L1 L2 (A|B)))

((rappend X Y Z)
      (reverse Z Z1)
      (reverse Y Y1)
      (append Y1 X1 Z1)
      (reverse X1 X))

((part X () () ))
((part X (A|B) L2)
      (if (eq X A)
          ((eq L2 (X|B)))
          ((part X B L2)) ))

/*
```

```
****************************************************************
*                                                              *
* Show_fields obtain all the allowed fields for the object.    *
* It will catenate all the valid fields inherited from the     *
* superclasses. The clause (X def ...) for each superclass is  *
* checked.                                                     *
****************************************************************
*/

/*
((show_fields OBJ Q)
      (OBJ precedence|Y)
      (allowed_fields Y () Q)
      (p fields defined for OBJ are)
      (pp)
      (p Q)
      (pp))
*/

((show_fields OBJ Q)
      (OBJ precedence|Y)
      (allowed_fields Y () Q))

((allowed_fields () P P))
((allowed_fields (A|B) P Q)
      (A def L)
      (append P L R)
      (allowed_fields B R Q))

((run X)
      (clear prog)
      (?((X)) ))


((clear prog)
      (odict X)
      (delcl ((X def|Y)) )
      (fail))
((clear prog)
      (odict X)
      (delcl ((X precedence|Y)) )
      (fail))
((clear prog)
      (odict X)
      (delcl ((X methods|X)) )
      (fail))
((clear prog)
      (kill casel)
      (kill odict)
      (fail))
((clear prog)
      (def_obj frame ())
      (def_obj ref_frame (mini_expert) ))
/*
```

B.4.3 Program Frame2.frl - Frame Matching

```
err_han new_error
```

```
****************************************************************
*                                                              *
* generate_expert_system takes the list from the attribute     *
* mini_expert of FRAME_NAME which is of the type ref_frame and *
* pass each member of the list to be processed by the          *
* predicate process_clause.                                    *
*                                                              *
****************************************************************
*/

((generate_expert_system (end)  FRAME_NAME))
((generate_expert_system (X|Y) FRAME_NAME)
     (process_clause FRAME_NAME X)
     (generate_expert_system Y FRAME_NAME))


/*
****************************************************************
*                                                              *
* process_clause generate one bundle element from the given    *
* list. Each list can be of the following forms (and|Y) or     *
* (or|Y).                                                      *
*                                                              *
****************************************************************
*/

((process_clause FRAME_NAME (and|Y))
     (anding FRAME_NAME Y () Q))
((process_clause FRAME_NAME (or|Y))
     (oring FRAME_NAME Y CLAUSE))
((process_clause FRAME_NAME ((not X)|Y))
     (anding FRAME_NAME ((not X)|Y) () CLAUSE))

/*
****************************************************************
*                                                              *
* oring and pre_oring generates a clause which involves or.    *
* The terminating condition is either the end of the list (ie  *
* the or is nested withing an AND) or ((supp Q)) (ie or is the *
* beginning of the list). Pre_oring takes the number of the    *
* attribute, look it up in the attr_list, and appends all the  *
* ors together to form a clause1 of the form (or (<atom>) ...) *
* if it is terminated by encountering a (). Otherwise if it is *
* terminated by encountering ((supp Q)) then a clause          *
* (($FRAME_NAME match)(clause1)): Q is added, which forms part *
* of the local expert system. The clause added will be part of *
* bundle for ($FRAME_NAME match).                              *
*                                                              *
****************************************************************
*/

((oring FRAME_NAME (X|Y) CLAUSE)
     (pre_oring FRAME_NAME (X|Y) CLAUSE ()))


((pre_oring FRAME_NAME () CLAUSE1 FIX)
     (append (or) FIX CLAUSE1))
((pre_oring FRAME_NAME ((supp Q)) CLAUSE1 FIX)
     (append (or) FIX CLAUSE2)
     (add_dollar FRAME_NAME DOLLAR_FRAME_NAME)
     (append ((DOLLAR_FRAME_NAME match)) (CLAUSE2) CLAUSE1)
```

```
     (addcl CLAUSE1 : Q))
((pre_oring FRAME_NAME ((not (SS Q))|Y) CLAUSE FIX)
     (add_dollar SS DOLLAR_SS)
     (append FIX ((not DOLLAR_SS Q)) SPIT)
     (pre_oring FRAME_NAME Y CLAUSE1 SPIT))
((pre_oring FRAME_NAME ((not SS)|Y) CLAUSE1 FIX)
     (message FRAME_NAME SS Q _)
     (add_dollar SS DOLLAR_SS)
     (append FIX ((not DOLLAR_SS Q)) SPIT)
     (pre_oring FRAME_NAME Y CLAUSE1 SPIT))
((pre_oring FRAME_NAME ((SS Q)|Y) CLAUSE1 FIX)
     (add_dollar SS DOLLAR_SS)
     (append FIX ((DOLLAR_SS Q)) SPIT)
     (pre_oring FRAME_NAME Y CLAUSE1 SPIT))
((pre_oring FRAME_NAME (SS|Y) CLAUSE1 FIX)
     (message FRAME_NAME SS Q _)
     (add_dollar SS DOLLAR_SS)
     (append FIX ((DOLLAR_SS Q)) SPIT)
     (pre_oring FRAME_NAME Y CLAUSE1 SPIT))

/*
****************************************************************
*                                                              *
* anding takes a list and form a clause which "ands" all the   *
* predicates together. It is always terminated by encountering *
* ((supp Q)). It looks up the Xth attribute in the attr_list,  *
* and its associated value. Each of these attributes are       *
* joined together via "and". If the element is (not N) or      *
* (or|Y), it performs the not or use oring.                    *
*                                                              *
****************************************************************
*/

((anding FRAME_NAME ((supp (X Y))) P Q)
     (add_dollar FRAME_NAME DOLLAR_FRAME_NAME)
     (append ((DOLLAR_FRAME_NAME match)) P Q)
     (addcl Q : (X Y) ))
((anding FRAME_NAME ((not A)|XT) P Q)
     (message FRAME_NAME A QV _)
     (add_dollar A DOLLAR_A)
     (append (not DOLLAR_A) (QV) DUMP)
     (append P (DUMP) R)
     (anding FRAME_NAME XT R Q))
((anding FRAME_NAME ((or|Y)|XT) P Q)
     (oring FRAME_NAME Y CLAUSE)
     (append P (CLAUSE) R)
     (anding FRAME_NAME XT R Q))
((anding FRAME_NAME (A|T) P Q)
     (message FRAME_NAME A QV _)
     (add_dollar A DOLLAR_A)
     (append (DOLLAR_A) (QV) DUMP)
     (append P (DUMP) R)
     (anding FRAME_NAME T R Q))

?((def_obj frame ()))
?((def_obj ref_frame (mini_expert)))
```

```
**************************************************************
*                                                            *
* make_database takes a frame, which must be of type frame.  *
* From the attr_list of the frame, a local set of facts are  *
* generated. These clauses are of the form (($attr value)): Q *
* It uses the sub predicate process_attr, which the attr_list *
* as argument.                                               *
*                                                            *
**************************************************************
*/

((make_database FRAME_NAME A_LIST)
     (check_obj FRAME_NAME L)
     (L precedence|X)
     (member frame X)
     (show_fields L A_LIST)
     (process_attr A_LIST FRAME_NAME))


/*
**************************************************************
*                                                            *
* make_expert takes a frame of type ref_frame and generate a *
* localised expert system using generate_expert_system.      *
*                                                            *
**************************************************************
*/

((make_expert FRAME_NAME)
     (check_obj FRAME_NAME L)
     (L precedence|X)
     (member ref_frame X)
     (message FRAME_NAME mini_expert (begin|XT) _)
     (generate_expert_system XT FRAME_NAME))

/*
**************************************************************
*                                                            *
* process_attr takes the attr_list and obtain the support and *
* value for each attribute. An internal clause in the form of *
* (($attr value): Q is generated for each attribute in the   *
* list.                                                      *
*                                                            *
**************************************************************
*/

((process_attr () F))
((process_attr (X|T) FRAME_NAME)
     (message FRAME_NAME X (V supp Q) _)
     (add_dollar X Y)
     (addcl ((Y V)) : Q)
     (process_attr T FRAME_NAME))


/*
```

```
****************************************************************
*                                                              *
* multi_kill takes a list of predicate names as argument and   *
* remove all the named predicates in the database.             *
*                                                              *
****************************************************************
*/

((multi_kill ()))
((multi_kill (X|T))
     (kill X)
     (multi_kill T))

/*
****************************************************************
*                                                              *
* compare_frame takes two frame names as arguments. These two  *
* frames must have the same attr_list defined. Otherwise type  *
* mismatch will result. Compare_frame uses make_expert to      *
* generate a local mini_expert system from the the REF_FRAME   *
* which is of the type ref_frame. Make_database is used to     *
* generate the localised database from FRAME_NAME which is of *
* the type frame. If reasoning is required, the FRIL support   *
* logic shell is invoked, ie fs must be already loaded. When   *
* control is returned to compare_frame (after the shell is     *
* terminated using b), the support for the clause ($ref match) *
* is returned and then all local database and $ref are deleted *
* and the support for clause (($REF_FRAME match)) is given as  *
* S. The support for (($REF_FRAME match)) gives an indication  *
* of how similar the frame in question is similar to           *
* the REF_FRAME.                                               *
*                                                              *
****************************************************************
*/

((compare_frame FRAME_NAME REF_FRAME S)
     (!)
     (make_expert REF_FRAME)
     (make_database FRAME_NAME A_LIST)
     (add_dollar REF_FRAME DOLLAR_REF_FRAME)
     (p investigate matching criteria - "(y/n)? ")
     (get stdin X)
     (name (X) XN)
     (!)
     (if (eq XN y)
         ((p "to return to program," enter b at ?)
          (fs (DOLLAR_REF_FRAME match)))
         ((dum)))
     (supp_query ((DOLLAR_REF_FRAME match)) S)
     (kill DOLLAR_REF_FRAME)
     (add_dollar_list A_LIST DOLLAR_A_LIST ())
     (multi_kill DOLLAR_A_LIST))
((compare_frame FRAME_NAME REF_FRAME not_valid)
     (p FRAME_NAME and REF_NAME are not of same type))


/*
```

```
((new_error 400 support _Message E L S)
      (!)
      (pp E)
      (pp L)
      (abort))
((new_error N A B C D E)
      (pp N A B C D E)
      (p second branch)
      (pp)
      (error N A B C D E))
*/

/*
*****************************************************************
*                                                               *
* find_order gives the pth element of list (A|B) as Y           *
*                                                               *
*****************************************************************
*/

((find_order 1 (A|B) A))
((find_order P (A|B) Y)
     (sum Q 1 P)
     (find_order Q B Y))


/*
*****************************************************************
*                                                               *
* add_dollar_list takes a list of strings and add $ before      *
* each string.                                                  *
*                                                               *
*****************************************************************
*/

((add_dollar_list () Y Y))
((add_dollar_list (X|T) Y Z)
      (add_dollar X DOLLAR_X)
      (append Z (DOLLAR_X) Z1)
      (add_dollar_list T Y Z1))

/*
*****************************************************************
*                                                               *
* add_dollar takes a string as argument and adds $ to the       *
* string.                                                       *
*                                                               *
*****************************************************************
*/

((add_dollar X Y)
     (name XN X)
     (append (36) XN XXN)
     (name XXN Y))

/*
```

```
((prog)
    (isa mammal frame)
    (def_obj mammal (intelligence hair size
                     legs hands appetite
                     nose_size tail))
    (isa mammal_ref mammal ref_frame)
    (generate human mammal_ref
         (intelligence high)
         (hair little)
         (size medium)
         (legs two)
         (hands yes)
         (appetite omnivore)
         (nose_size small)
         (tail none)
         (mini_expert
            (begin
               (and intelligence tail (supp ((0.8 1)(0.0 0.35))))
               (and hands
                legs (or appetite (appetite herbivore) )
                (supp ((0.4 1)(0.2 0.6))))
               (and hair size legs nose_size (supp ((0.4 1)(0.2 0.6))))
               (and tail (supp ((0.8 1)(0.0 0.2))))
              end)
    ))

   (generate Peter mammal
         (intelligence (high supp (0.8 1)))
         (hair (little supp (0.9 1)))
         (size (medium supp (1 1)))
         (legs (two supp (1 1)))
         (hands (yes supp (1 1)))
         (appetite (herbivore supp (1 1)))
         (nose_size (small supp (1 1)))
         (tail (none supp (1 1)))
    )

 (generate monkey mammal
         (intelligence (medium supp (1 1)))
         (hair (a_lot supp (1 1)))
         (size (medium supp (1 1)))
         (legs (two supp (1 1)))
         (hands (yes supp (1 1)))
         (appetite (omnivore supp (1 1)))
         (nose_size (small supp (1 1)))
         (tail (yes supp (1 1)))
 ))
```

```
((prog1)
     (isa spec_frame frame)
     (def_obj spec_frame (attr1 attr2 attr3 attr4 attr5))
     (isa ref_ref_frame spec_frame ref_frame)
     (generate spec spec_frame
                     (attr1 (value1 supp (0.9 1)))
                     (attr2 (value2 supp (0.8 0.9)))
                     (attr3 (value3 supp (0.4 0.6)))
                     (attr4 (value4 supp (0.5 0.7)))
                     (attr5 (value5 supp (0.2 0.5))))
     (generate ref ref_ref_frame
                     (attr1 value1)
                     (attr2 value2)
                     (attr3 value3)
                     (attr4 value4)
                     (attr5 value5)
            (mini_expert
                     (begin
                         (and attr3 attr2 (supp (0.8 0.9)))
                         (and (or (attr1 hello) attr2) attr4 (supp (0.6
                         (or attr1 attr4 (supp (0.5 0.7)))
                         (and (or (not attr3) attr1) attr2 (supp (0.4 0.
                         ((not attr5) (supp (0.6 1)))
                      end)
             )
      ))


/*
(high [0.4:0 0.5:0.4 0.8:1 1:1 1.01:0])
(low [-0.99:0 0:0.5 0.2:1 0.45:1 0.5:0.4 0.6:0.1 0.7:0])
(little [-0.99:0 0.3:1 0.5:0.3 0.6:0 0.7:0])
(a_lot [0.3:0 0.6:0.5 0.7:0.9 0.8:1 1:1 1.01:0])
(two [0:0 2:1 2.01:0])
(none [-0.99:0 0:1 0.1:0 1:0])
(yes [0.99:0 1:1 1.01:0])
*/
```

B.4.4 Demonstration Program - Example1.frl

```
/*
****************************************************************
*                                                              *
*                 Demonstration Program                        *
*                                                              *
****************************************************************
*/

/*
Program - Example1.frl
*/

((prog0)
   (def_obj person (name age sex))
   (def_method person birthday)
   (def_method person print_self)
   (isa professor person)
   (def_obj professor (telefon_no inaugural_lecture_subj subject))
   (def_method professor write_paper)
   (def_method professor print_self)
   (generate peter person (name Peter_Townsley)(age 22)(sex male))
   (generate roberts professor (name Peter_Roberts)(age 55)(sex male)
                     (telefon_no xxxxx)
                     (inaugural_lecture_subj Hierarchical_control_system
                     (subject control_engineering)) )

((print_self person (X _) Z)
   (message X name Z _)
   (p My name is X)
   (pp))

((birthday person (X _) Z)
    (message X age Z _)
    (p Happy birthday.)
    (pp)
    (sum Z 1 Y)
    (update_obj X (age Y)) )
((print_self professor (X _) _)
    (message X name A _)
    (message X subject B _)
    (p My name is A and 'I' am professor of B)
    (pp))
((write_paper professor (X _) Z)
    (message X inaugural_lecture_subj Z _)
    (p 'I will write a paper on' Z)(pp))
```