University of Alberta

A Knowledge Representation Architecture for Expert Process Monitoring Systems: Applied to a Pulp Bleaching Process

by

Samuel Wyatt Erlenbach ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

in

Process Control

Department of Chemical Engineering

Edmonton, Alberta

Spring 1995

ISBN 0-612-01603-X

MEMORANDUM

TO:	MING RAO SAM ERLENBACH	DATE:	April 24, 1995
FROM:	KEITH DANIELSON	FILE:	
COPY TO:		DOC:	
SUBJECT:	ON LINE MONITORING		

This memo is to approve the reference to DMI, and process information from DMI in Sam Erlenbach's Thesis for the University of Alberta project on "On-Line Monitoring."

[illegible]

Keith Danielson
Pulping Superintendent

KD:LH

University of Alberta

Library Release Form

Name of Author: Samuel Wyatt Erlenbach

Title of Thesis: A knowledge Representation Architecture for Expert Process Monitoring Systems: Applied to a Pulp Bleaching Process

Degree: Master of Science

Year this Degree Granted: 1995

[illegible]

#102, 10731-84th Avenue
Edmonton, Alberta
Canada

May 2 , 1995

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled "A Knowledge Representation Architecture for Expert Process Monitoring Systems: Applied to a Pulp Bleaching Process" submitted by Samuel Wyatt Erlenbach in partial fulfillment of the requirements for the degree of Master of Science in Process Control.

[illegible]

Dr. Ming Rao, Supervisor

[illegible]

Dr. Grant Fisher

[illegible]

Dr. Francis Jeffry Pelletier

I dedicate this thesis

to

Samuel **Braedyn** Patrick Erlenbach, my son,
who I love and hope that my studies open future doors for him,

Diane Marie White, his mother,
whose patience and belief in me is eternal,
and whose sincere, forgiving nature I hope is rewarded in turn,

Loreen Deborah Erlenbach, my sister,
whose own tenacity towards higher education spurred mine,
and who laid forth the means to find the answers I have long been seeking,

Alice **Noreen** Erlenbach, my mother,
for the love, the confidence and the hope she carries for each one of us

and the rest of the **family**,
who together we all share our strengths and give support

Abstract

This thesis proposes a new knowledge representation architecture for process monitoring expert systems. The architecture is based on the object oriented paradigm as a structure to represent world knowledge. The thesis proposes several new perspectives for the paradigm and its application. Several natural forms of experience are interwoven into the structure of the architecture. Experience is applied in a predictive manner. The architecture encompasses the conceptual and physical systems of a world through embedding of interrelationship knowledge into component objects. Associative linking of memory is included in the architecture as a basis for intricate pathways for mental exploration. Human thought processes are replicated through the decoupling of thinking and information knowledge. The architecture provides a structure to capture and execute knowledge and thinking behaviours for process monitoring activities. A prototype based on the proposed knowledge representation architecture was successfully implemented on a pulp bleaching process of a Kraft pulp mill.

Acknowledgments

I would like to express my gratitude to *Dr. Rao*, my Masters studies supervisor. His diligence brought together industry, government and the University of Alberta to form a project that provided the funding and the opportunity to do practical research in the field of artificial intelligence. I am thankful to *Dr. Rao* for the foresight to have taken me on as a student under his supervision, and for the knowledge that he has passed on to me during my studies.

I would like to express my appreciation to members of my defense committee for taking the time out of their full schedules to read and evaluate my thesis, and for the sincere care to do so. These members are *Dr. Krishnaswamy Nandakumar*, the chairman, *Dr. Grant Fisher* and *Dr. Jeff Pelletier*, the committee members.

I would like to thank the companies and their members, Daishowa-Marubeni Int., MoDo Chemetics, Ranger, and Perde Enterprise Inc. for their contribution and sponsorship of the project, and for their ability to envision the future. In particular I would like to thank *Keith Danielson*, Pulping Superintendent at Daishowa-Marubeni, for all his help and effort to provide the resources needed to carry out the project work. I would like to thank *Dan Bowes*, Operator Technician I of Daishowa-Marubeni, who offered his considerable experience knowledge for the project, and who spent extensive effort to see the project's success. *Katrinka Scott-Burns*, Pulping Technician at Daishowa-Marubeni, to whom I am thankful for her willingness to handle the sudden and last minute technical demands. *Sven-Erik Henriksson* of Perde Enterprises Inc., *Jorgen Oloffson* of MoDo Chemetics, and *Christer Henriksson* of Ranger, all of who participated in the planning and organization of the project. A special acknowledgment goes to the Canadian Alberta Partnership in Forestry who provided funding for this and other projects.

I would like to express special gratitude to the following people for the support they gave me. *Yiqun Ying*, who worked long hours with me, and who listened to my brain storming and hypothesizing on our drives to Peace River. *Yiqun* who has a clear mind and was always correct in logistic matters. *Leanne Thomson*, my r[illegible] [illegible]ibraltar, who picked me up every time I fell no matter how stormy the seas had [illegible]. She is a woman with a "heart of gold". *Carolyn Blanchard* for the wonderful editing she did on Chapter 2 of my thesis, and for the encouragement she provided in those dark hours. We never reached Chapter 3.

Finally, I would like to thank the numerous people who played a part in this step of my quest of life. Their passing through the same corridors in time have helped to shape who I have become, and I hope that I have helped shape them. These people include *Alan Ayasse*, *Benedicte Boulanger*, *Janet Cummings* who offered my first refuge, *Kevin Dorma*, *Hassan Farzadeh* who offered me genuine hospitality when I arrived, *Cindy Heisler* far too intelligent for what she does, *Andrew Jenkins*, *Corina Kelepouris* who set the path for me to be, and who I know now was the reason I came here for, *Shulamit Kuttner* to whom my casual questions were the worst ones I could have asked, *Yves Lacombe* we never found the time to party as we always talked about, *Alison Miller*, *Randy Miller*, *Olli* my friend who revived me by spending a week canoeing on the ocean and backwaters at Campbell River, *Susan Richardson*, *Cordula Schink*, *Deepika Shah*, *Birgitte Willumsen* an eternal friend, the members of the Intelligence Engineering Lab, and the Chemical Engineering faculty members who share their academic lives with their students. There are others as well. In addition, I must acknowledge the cafes along Whyte Avenue where I spent much time over steaming cups of coffee contemplating and composing, and the Rocky Mountains where I go to clear my mind and soul.

Table of Contents

List of Figures

Explanation of Styles

Italicized words	Identifies key thesis concepts or words.
Italicized phrases	Emphasize a passage.
Bold words	Identifies an expert system frame attribute name
Bold headers	Chapter, section or special paragraph header text

1 Introduction

This thesis proposes and presents a new knowledge representation architecture for expert systems which are applied for the purpose of process monitoring. The objective of the research work performed for this thesis was to provide a better means for actuating a process monitoring expert system. The work was aimed at eliminating some present barriers and deficiencies in representing knowledge and in process monitoring applications. The architecture is based on the object oriented paradigm as a structure to represent knowledge about a world. The thesis proposes several new perspectives for the paradigm and its application. The architecture interweaves representation of several forms of experience knowledge into the structure, thus adding another dimension of reasoning. Recognition experience knowledge is used in a predictive manner that provides a natural future temporal ability. The architecture is made more encompassing of a world by the integration of interrelationship knowledge in a way that equips an elaborate representation for systems. A powerful component of the human mind, associative linking of memory, is developed into the architecture creating intricate pathways for mental exploration. The architecture offers a means for replicating cognitive thinking through the decoupling of thinking and information knowledge, and the representing of more than one dimension of thinking behaviour. The architecture provides a structure to capture and execute both the knowledge of, and the thinking behaviour for, process monitoring activities.

The research that led to the conceptualization of the proposed knowledge representation architecture originated from a joint venture between the Pulp and Paper industry, and the University of Alberta. (This joint venture is discussed in more detail in Chapter 3.) It was proposed by the Intelligence Engineering Lab, at the University of Alberta, to an Alberta

Kraft Pulp mill, Daishowa-Marubeni Peace River Pulp Division, that expert system technology could be beneficial for their operations. The mill agreed that the technology showed promise for the purpose of monitoring the activities around the pulp bleaching process of their mill. The pulping superintendent, Keith Danielson, saw the potential of a system that could use the recorded knowledge of previous process upset incidents, and monitor the process to avoid their reoccurrence. These incidents are recorded as unique reports. This presented the University with the problem of developing an expert system for process monitoring which could be configured with knowledge in episodic form. This can not directly be achieved with commercial systems. It was further realized that other deficiencies exist with commercial expert systems when applied to process monitoring. (Several deficiencies are discussed in Section 1.5.1). This led to the examination and study of how to better represent an industrial process domain and the process monitoring reasoning activities. The result of this study is the knowledge representation architecture presented in this thesis. In fulfillment of the joint venture, a prototype system was then constructed and implemented at Daishowa-Marubeni, Peace River mill. This prototype system serves two purposes. One of the purposes is to supply the mill with a process monitoring expert system for the pulp bleaching plant that meets their process monitoring requirements and needs. The other purpose of the prototype is to provide a practical case study to evaluate the worthiness of the proposed knowledge representation architecture concept as a real-world application. The case study will also provide an indication of the significance of applying expert system technology to the monitoring of a pulp bleaching process.

The concepts and theory of the knowledge representation architecture are presented in Chapter 2. The development, implementation and results of the case study prototype system at the Daishowa-Marubeni Peace River Pulp mill are discussed in Chapters 3 to 5. The thesis concludes in Chapter 6 with an evaluation of the proposed architecture based on the practical case study results, and with a summary of the main concepts presented in this thesis. This Chapter introduces various concepts involved in the research work and in the prototype project.

1.1 Kraft Pulping Process

The Kraft pulping process is a chemical decomposition of chipped wood into pulp fibres that can be used for making paper and paper board products. Pulp fibre is a natural, highly purified polymer that is made from cellulose and hemicellulose polymer chains. Besides the fibres, wood consists also of lignin, which is a macromolecule composed of carbon, hydrogen and oxygen. Lignin servers the purpose of acting as a glue between the wood fibres. Lignin is also dark and gives the wood its colour. Pure fibres are somewhat transparent, but appear in bulk to be bright white. The aim of the pulping process is to separate the fibre from the lignin. For chemical pulping processes such as the Kraft process, the lignin is extracted by means of a chemical reaction (Reeve, 1989). The following is a brief overview of the Kraft pulping process.

The first stage of a Kraft pulping process is the digester. Wood chips are feed into the digester unit at the top. The chips are pr-steamed to swell them to promote impregnation of chemicals. White liquor (Na_2S and NaOH) is introduced to the chips to react with the lignin. The chips and absorbed liquor move down the digester as a plug. The internal digester temperature and pressure cause the chips to "cook" (chemically react), etching out the lignin. The lignin and the spent white liquor are then removed from the digester as black liquor. The black liquor removal can be done through various arrangements of back washing (counter current flow) near the base of the digester. The extracted black liquor is fed into a chemical recovery process where the essential chemical components of the white liquor are returned to the process. The speed at which the chips descend through the digester is governed by the *blow* rate, or the rate at which pulp is let out of the digester. This speed dictates the latency, or cook time, for the chips. The chemical reaction is exothermic, thus once the reaction starts it is thermally self perpetuating. Most digesters are pressurized which is controlled by small additions of superheated steam. The temperature, pressure, latency time, and chemical addition are all crucial and sensitive for a proper reaction to occur. It is very easy to either over-cook the wood and begin to decompose the cellulose and hemicellulose polymer chains, or under-cook and not separate the fibres. This sensitivity makes the digesting process very difficult to control (Albright & Wilson, 1988).

The digested pulp that is emitted from digester unit is dirty, containing both residual lignin and other trace forms of dirt substances. The next stages applied to the process serve the purpose of cleaning and bleaching pulp. The brightening of pulp is done through chemical reactions, either to discolour the lignin, or more preferably to extract it. This is usually done in several stages. Each stage introduces a chemical reaction, allows time for the reaction to occur, then washes the chemicals and contaminants from the pulp. The washing of chemicals from the pulp has two purposes, to remove the spent chemicals, and to stop further reactions from occurring. This is extremely important. Although the chosen bleaching chemicals have a greater affinity with the lignin, if they are exposed to the fiber long enough they will start decaying the cellulose and hemicellulose polymers. Figure 1.1 depicts a chlorination bleaching stage of a Kraft pulping process.

The common chemical bleaching processes are elemental oxygen, elemental chlorine, and chlorine dioxide. Elemental oxygen is a weak oxidizing agent that will react with lignin under high pressure and temperature in an alkaline solution. It is the cheapest bleaching agent, but it can not be used alone. Most often it is applied as a first stage of bleaching.

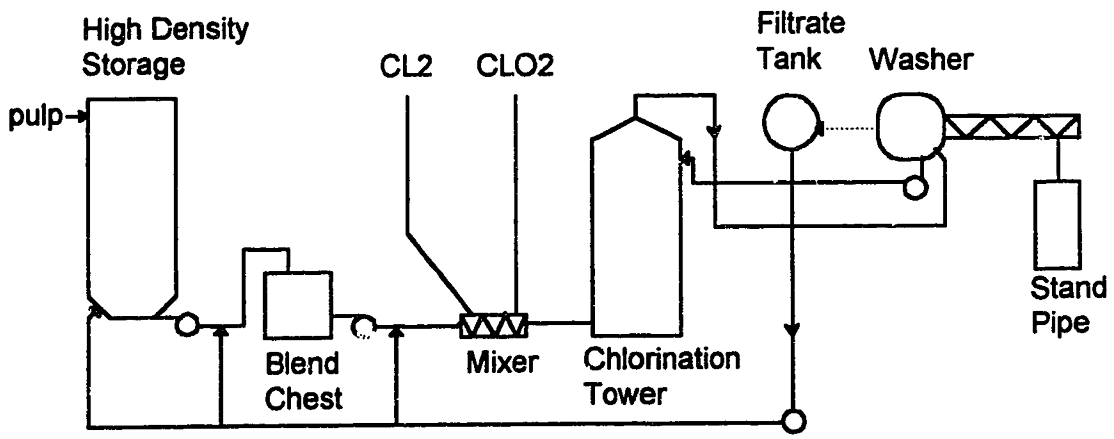


Figure 1.1: Chlorination Bleaching Stage

Elemental chlorine is the second cheapest bleaching agent. It reacts readily with lignin but also is reactive with the fibre. Chlorine dioxide has much better bleaching properties than the other known agents. Its reactions are highly selective towards lignin and can react at lower temperatures; however, it is expensive and highly volatile. Usually it must be generated at the mill site, which creates a safety concern. Chlorine Dioxide is also a much more environmentally friendly oxidizing agent, as it leaves almost no chlorinated organic compounds (Albright & Wilson, 1988). Due to the extreme pressure put on by environmentalists most modern pulp bleaching plants now are tending towards elemental chlorine free pulping. The case study expert process monitoring system for this thesis was applied to a pulp bleaching process at the Daishowa-Marubeni Peace River Pulp Mill. The stages of bleaching used at the Daishowa-Marubeni mill are a chlorine (or chlorine dioxide for efc pulp), followed by an oxygen, and then two possible stages of chlorine dioxide.

The final stage of the Kraft pulping process is the pulp machine, or the pulp drying stage. The pulp machine forms the pulp into mats and dries it for transportation. The pulp is fed into a headbox where it is laid out on a rotating wire mesh. Water is hydraulically removed through the wire mesh forming a moist mat. This mat is then fed into a bank of steam heated drum dryers that complete the drying process. The mats are then peeled of the machine, cut and wrapped for shipping (Smook, 1990; CPPA, 1989).

One of the greatest difficulties about operating a pulp bleaching plant is that the exact chemical composition of the pulp is unknown. Not only is it not measurable on-line by present technology, but it is highly variable. Thus, the control of the process is designed around affected properties that are measurable, such as a brightness scale. Various on-line instruments are employed to determine a measurable value for the affected properties, but by the nature of the installation these can only indicate general averaging over the pulp volume. Further, the pulp process measuring devices are extremely hard to calibrate because the principles upon which the measurement is based are typically nonlinear. A calibration is usually only accurate in a certain range of operation, with measurement degradation occurring at and past the extremes. Another problem with the instruments is that they are exposed to a very rough environment. Wear and natural phenomena, such as surface coating, cause the calibrations to drift off correct settings. The control of a pulp bleaching plant thus, is plagued by the inability to measure input chemical composition, by its time varying nature, and by the inaccurate and lacking measurements of affected properties which indicate process conditions. Precise controls have yet to be implemented for a Kraft process and much of the control is left up to human *know-how*.

There are other difficulties inherent with the operation of a pulp bleaching plant as well. The process is very extensive, which plagues operations personnel with an over abundance of information to deal with. A pulp bleaching process can have hundreds of process and control data values being electronically accessed. In times when operations are hectic, such as startups, the operations personnel can adequately monitor all parts of the process. In such periods the operations personnel must focus on a few operations at a time, leaving the other operations to do as they will. The process is realized by many physical components and devices, each having their own operating nonlinearities and idiosyncrasies that need to be known about. Overall operations of the process can be confounded by abnormal events such as motors tripping off, and by operational problems such as pulp washers plugging. From the process operation personnel's point of view, there are many things that can go wrong. Corrections to a situation are not straight forward. The process is highly interactive and what seems to be an obvious control action often results in other unexpected problems. Operating experience over technical process knowledge is often needed to affect a reliable handling of a situation. Further, when the process becomes upset, the control of the process tends to come apart and the operator is left scrambling to re-establish the process. In such duress the mind may freeze, be less effective, and be prone to making mistakes (Laffey et al., 1988). Much of what the operator should do is overlooked in the moments of confusion. These operation difficulties are more exaggerated for new operations personnel who lack the depth in knowledge and the refinement of experience. The success of the pulp bleaching operations depends heavily on operations personnel's knowledge, ability, and the breadth of experience that they can draw on to handle the operating problems.

A powerful and useful tool for the Kraft pulp process operations would be an expert system that could monitor all aspects of the process, and supply well thought out situation solutions based on expertise gained through process monitoring experience. Such a system could provide twenty-four hour process monitoring as an operational backup aid to assist the operations personnel in noticing and handling operational situations. The potential savings in loss product, costs, and environmental issues are in the hundreds of thousands of dollars. See Appendix A for two estimates of potential savings for the pulp bleaching process. An investment to implement an expert system for this purpose would have the potential for a very quick payback and a high rate of return on the investment. Operations of other industrial processes face similar difficulties as well. The objective of developing expert system technology for industrial process monitoring is thus deem a worthwhile venture.

1.2 Review of the Object Oriented Paradigm

This section provides a brief review that highlights the main concepts of the object oriented paradigm (OOP). This review is far from exhaustive. A very good explanatory presentation of the OOP concepts is done by Entsminger (1990). Martin & Odell (1992), and Brunet et al. (1993) discuss OOP analysis and design practices.

The object oriented paradigm models all aspects of the world as *objects*. It is a model that is felt to replicate how a human consciously perceives his world. There are six main concepts that compose the model, the concepts of *characteristics*, *behaviours*, *encapsulation*, *classes* and *instances*, *inheritance*, and *composition*. *Characteristics* and *behaviours* are similar in that they describe the essence of an *object*. The world is seen to be completely composed of concrete entities, each entity being perceived as an *object*. Each entity embodies *characteristics*; such as colour, taste, or shape, and *behaviours*; such as that it flies, that it rolls, or that it sleeps. The *characteristics* and *behaviours* are properties of an entity. These properties, or attributes, are *encapsulated* in an *object* to represent that entity. The concept of *encapsulation* stipulates that all information about a perceived entity is incorporated in the *object*. Information about an entity is accessed by thinking about the object. In the case of computable representations, accessing the information is done by *sending* a request message to the *object*. Each object and object attribute is symbolically represented in a computable machine by names. A message identifies which object and attribute it is referring to by the enclosing the names in the message.

The paradigm regards the accumulation and organization of knowledge in a similar manner as the human mind, through categorization and classification. When humans first come into the world they are faced with copious amounts of external stimuli and information, but without the ability to reconcile it. They begin to pattern (categorize) the stimuli, and identify repeated patterns, such as a chair, as *objects*. Over time they observe consistencies about the *object's* properties (*characteristics* and *behaviours*), and begin assimilating information about the *object*. As a deeper knowledge of *objects* is gained, the human begins to understand how objects are composed of other *objects*. This knowledge of *composition* is retained in an *object*.

If humans had to remember all information of every single *object*, their brains would quickly fill to capacity. Instead, the human mind continues perceiving patterns and identifies similar properties (classifications) among various *objects*. The mind groups these *objects* which exhibit a commonality of properties into a class of *objects*. It stores the common properties in a *class object* instead of each individual remembered *object* (*instance*). The individual *objects* inherents the properties of the class from which they belong. The human recalls the class properties for an individual *object* from the class *object*. For example, a robin is a bird. A property of a robin is that it has colour red. However, the properties that the robin has wings and that it flies are common to all birds. This information is retained by the bird *object*. When a human thinks of a Robin, they also think of it as a bird. The principle of classification and inheritance forms a taxonomy hierarchy. The taxonomy that is developed in a human's mind can become extremely intricate with many levels of sub and super classes.

A class of objects is also known as a *type* of objects in a computable representation. The attributes encapsulated within an object which are available to the external world form the *interface* of an object. An *object type* will have a specific interface which is inherented by all *instance* objects. This is an important notion, as it is the reason why messages can be sent generically to any object of a *type*. The type interface is consistant with all instances of the object *type*.

The object oriented paradigm defines these concepts. An implementation of the object oriented paradigm must realize these concepts for the medium being used. Object oriented programming languages (OOP), such as C^{++}, attempt to develop these concepts for a computable electronic machine. These languages specify how characteristics and behaviours are specified, and how to inform the language compiler of the classes, instances and inherited objects. The language compiler ensures that the paradigm is adhered to. Many applications using OOP languages have been to express visual entities for screens, or dynamic database entities, for such applications as air traffic control. The concept of inheritance has a strong advantage to programming languages, in that a new object can inherit from existing objects, thus eliminating re-engineering of code. One should not think of the object oriented paradigm as a programming language though; it is a paradigm, or a manner of modeling the human perception of the world.

1.3 Expert Systems

The intent of this section is to provide a brief overview into the field of expert systems. Expert systems belong to a general class of study called artificial intelligence, or AI. The discipline of AI has been defined as,

> "AI, which concerns the development of theories and techniques required for a computational engine to efficiently perceive, think, and act with intelligence in a complex environment" (Fox, 1990).

Expert systems form the category of artificial intelligence which is concerned with the understanding of the intelligence that is embodied in human expertise, and applying it to achieve solutions and results in some problem domain. Human expertise is the body of knowledge and capabilities that is encompassed within persons we classify as experts. This has been expressed more succinctly as,

> "An expert is a person who, because of training and experience, is able to do things the rest of us cannot: experts are not only proficient but also smooth and efficient in the actions they take. Experts know a great many things and have tricks and caveats for applying what they know to problems and tasks; they are also good at plowing through irrelevant information in order to get at basic issues, and they are good at recognizing problems they face as instances of types with which they are familiar. Underlying the behavior of an expert is the body of operative knowledge we have termed expertise. It is reasonable to suppose, therefore, that experts are the ones to ask when we wish to represent the expertise that makes their behavior possible" (Boullart, 1988).

Expert system theory includes both the capturing of expert knowledge, and the replication of various human intelligent activities. A combination of the theories from various research disciplines leads to the development of computer based expert system tools. These tools are employed to handle real world problems through the application of human expertise and mental ablilities.

There are roughly four notable reasoning methodologies presently used in expert systems:

- declarative
- procedural
- case-based

- model-based

Declarative takes the approach that one declares knowledge, or premises, on how a world is. Then, when presented with a set of observations about the world, one can infer various other world information, to the extent of the presented facts and the solution space mapped by the premises. (Genesereth & Nilsson, 1987)

Procedural point of view looks at the world in terms of cause and effect, and regards knowledge as being a procedure of reasoning about a world. When procedural reasoning is applied, this method of reasoning processes the information by applying the procedural knowledge to world until a solution is inferred (Cercone & McCalla, 1987).

Case-based is a relatively new form of reasoning that is being established. Case-based sees knowledge as a contiguous array of experience memory. If a problem is presented to a case-based system, the problem is dissected into salient attributes that are used to index the retrieval of pertinent memory experiences. The case-based reasoner then collates the recalled knowledge and generalizes a solution for the problem (Kolodner, 1989; Milne, 1987).

Model-based reasoning attempts to analytically or heuristically model a problem domain. The model is used by a reasoning agent to ascertain conclusions given conditions in the domain (Cauvin, 1992).

These are rather broad categories of expert system reasoning methodologies that offer more of an indication of different approaches than a classification of expert systems.

Another form of classification is *shallow* versus *deep* knowledge (Milne, 1987).

Deep knowledge encompasses information about a device or system and its inner workings. A deep knowledge reasoner can use this information to predict the actions of such entities.

Shallow knowledge is surface knowledge that can be applied. It is usually expressed by rule based systems where shallow assertions can be drawn by pattern matching conditions of a domain. Shallow knowledge does not rely on scientific founded relationships, but rather on observed behaviours (Williams et al., 1991).

The route taken by this thesis falls more into the procedural group methodology. The thesis work attempts, however, to integrate both shallow and deep knowledge representations.

There are several different means to organize and represent knowledge for the reasoning methodologies previously discussed. The following will briefly outline some of the more common ones:

For the declarative approach a couple of means have been used:

- first order predicate logic
- semantic networks

First order predicate calculus is widely used to represent and reason about domain knowledge. Concepts about the domain are expressed by atomic, logic or quantified sentences. These sentences declare the knowledge of the domain (Genesereth & Nilsson, 1987; Jackson, 1985).

Semantic nets began known as graphical method of representing first order predicate calculus, where the links expressed the relationship between terms. The meaning of semantic nets has become somewhat convoluted now, and the term has been applied to various forms of network representation. A useful application of semantic nets is for representing natural language (Cercone & McCalla, 1987; Lim & Cherkassky, 1992).

Procedural reasoning has many forms by which knowledge has been represented, and often a combination of several of the forms are used. Some of the knowledge representation forms are by:

- structural procedures
- rule sets
- frames
- petri nets
- decision trees

Structural procedures are straight-forward to understand. They represent knowledge of how to proceed through a reasoning activity. They are usually employed in conjunction with other forms of representation to express complex intelligence.

Rules represent knowledge in the form of laws of deduction, "for predicting changes over time, consequences of actions, and unobserved things that can be deduced from other observations" (Wood, 1983). The laws are expressed by If-Then statements. Rules are commonly grouped into sets that together express knowledge about some sub domain.

Frames came about as a method of clustering "chunks" of knowledge. This was first brought forth by Minsky (1975), in his *frames proposal*. The concept is that a "chunk" of knowledge can be organized and stored in slots within a frame. There does not appear to be any precise definition of what comprises frames slots. Some of the available frame-based systems consist of variable, rule, and procedure (or method) slots. Further, there seems to have been a definite evolution of frame-based systems to encompass inheritance (Is-A) concepts for frame-based representations (Cercone & McCalla, 1987). The expert system used in the case study project of this thesis is a frame-based system. This system was chosen to construct the proposed knowledge representation architecture because it could be molded to take on the required shape for the architecture.

Petri-nets are a graphical network model, with embedded logic, that represents a sequence of events. The logic at each node of the network specifies the conditions that must be met for the sequence to continue. Petri-nets are very useful, but in a limited scope of applications. They describe a strict predetermined sequence of knowledge and situations (Yao, 1994).

Decision trees are similar, but unlike a network that can take on any topology, the path through trees falls in one direction. Decision trees often represent a path of reasoning that must be performed to actuate an intelligent goal, whether the goal be a safe state for the domain, or to make a decision about the domain.

For case-base reasoning there are a variety of ways to represent the experience memory, depending on the application. The prevalent factor for the representation is the manner in which the knowledge can be indexed and cross-linked with the salient features of a problem. This has been done by any manner from a highly organized hierarchical memory scheme to highly indexed disordered memory (Ketler, 1993). The knowledge to perform the *case reasoning* has been frequently represented by procedural rules (Maher & Zhang, 1991).

Equally diverse as the means to represent knowledge are the means to infer intelligent information from the representations and domain facts. Again, only a partial selection of all the possible means will be reviewed here.

Inference is the prevalent way of deducing conclusions for first order predicate calculus representation of declarative methodology. The process of inferencing is performed through the application of inference laws. These laws express what conclusion can be drawn given a set of conditions. For first order predicate calculus representation, the sentence patterns of the declarative knowledge are matched with the inference conditions.

There are two well-known methods of inferencing for declarative reasoning, both of them use only few laws of inference:

- straight inference
- resolution principle

Straight inference approach uses several laws of inference, such as *modus ponens* and *modus tolens*. These are applied to the known and introduced premises to draw a conclusion.

Resolution principle uses only one law of inference. This method applies symbolic manipulation to the sentence structures of the declarative knowledge to achieve a unified form. The resolution principle is then applied to reduce the clauses to a concluding statement (Genesereth & Nilsson, 1987).

Inferencing is also applied to rule representation. Although rules are regarded as a procedural thinking methodology, their method of concluding information is quite similar to declarative inferencing. Rules are actually very specific laws of inference which state what information can be inferred from specific domain conditions. The inferencing of a set of rules can be thought of as progression of knowledge through reasoning layers, where conclusions established in an upper layer lead to information for rules at lower levels. The layering may be quite flat or it could be tree-like in structure. The procedure of inferencing is one of chaining through the levels of rules and applying each rules inference to the facts of the domain. Inferencing often works in either a forward or backward chaining manner.

Forward chaining of rules is where the condition part of the rule must be satisfied to draw a conclusion. The derived conclusions can then set conditions for rules of the next reasoning level that infer further conclusions. The end product is either a set of deduced conclusions from the conditions of the domain or no conclusions if the rules are unable to reach a deduction (Frost, 1986).

Backward chaining of rules is where the condition part is assumed if the conclusion exists. For backward chaining the intent is to work from a domain conclusion and validate the conditions of the domain that must be present for the conclusion to exist.

Search methods are widely used. Searching comes in many forms, each with its ability to optimize the search in some way for particular search spaces. The optimization must consider the search direction, such as depth first or breadth first. Searching appears in almost every form of intelligent reasoning. Even the application of rule chaining one must decide on the search method to employ to reach all the reasoning layers. Representations like decision tree and procedures, though they step through the layers, must establish whether the steps will proceed (search) down a path first or across a layer of branches.

In addition to the basic well-established means to infer information just mentioned, there are many other techniques. These techniques include conditional inferencing, statistical inferencing, fuzzy logic, the Dempster-Shafer evidential reasoning, genetic searches and neural networks, to name a few.

The present preferred technology for industrial expert system applications appears to be rule based, at least from surveys of the commercial expert tools. Declarative reasoning systems have not been used much because despite the beautiful and completeness of first order logic it is,

> "unworkable in any practical setting. The number of predicates needed to describe any real world would be immense, and the attempt to make them fully consistent would be nearly impossible" (Cercone & McCalla, 1987).

Case-base systems have not been developed to industrial market strength yet. Much learning and research are still being done on them. The large commercial vendors of industrial expert systems base their systems on rule-based technology, but offer a diversity of other features to aid in applying the technology. Several of the vendors offer an ability to integrate model-based, or simulation, packages. All the vendors offer libraries of mathematical functions that can be applied on raw incoming data to massage, filter, or enhance the value of the data. These functions, as well as any user developed routines, can be embedded in both the antecedent and the conclusion of rules. The vendors also provide various rule firing controls such as priority and context groupings that help in developing a meta-level of reasoning. The big vendors include such companies as Gensym Corp. (G2), Talarian (RTWorks), Comdale, and Activation Frameworks. There are also a variety of research systems that have been developed for process monitoring. A selection of these systems is discussed in Section 1.4.

Knowledge representation must be formed into an architecture to be able to replicate a system of human mental activities. Straight declarative or rule base reasoning can be used, but such an application is narrow and limited to strictly predefined solution spaces. Incorporating meta levels (reasoning about how to employ reasoning), or learning abilities requires an architecture be defined. There appears to be very few established knowledge representation architectures. Architectures are often individually designed for each application. These architectures generally take a modular layered approach, where intelligent activity at a lower layer generates the appropriate intelligent activity of higher levels until conclusions are reached An example can be seen in Alaman et al., (1992). Another approach taken is to make a meta layer of intelligence which controls the execution of a layer of intelligent tasks. See Ingrand et al., (1992) for an example. Several process monitoring systems and their knowledge representation architectures are discussed in Section 1.4. Two established and well-defined architectures are Blackboard and SOAR.

Blackboard is not a knowledge representation architecture as much as an expert system integration architecture. The basis of blackboard is that information is shared between various intelligent processes through the use of a *blackboard*, a common volatile data cache. Blackboard architecture can be used to glue together many different manners of expert system constructions (Arzen, 1992; Nii, 1986a; Nii, 1986b).

SOAR is an intensely researched architecture for representing general intelligence. The architecture provides a system that can

> "(1) work on a full range of tasks, from highly routine to extremely difficult open-ended problems; (2) employ the full range of problem solving methods and representations required to these tasks; and (3) learn about all aspects of the tasks and its performance on them" (Laird et al., 1987).

The SOAR architecture is built around the concepts of "chunking" (adding chunks of knowledge) and universal subgoaling (automatic generation of sub-goals to meet requirements or to perform tasks for an active goal. This thesis proposes some concepts similar to the subgoaling of SOAR.

1.4 Process Monitoring Expert System Applications

The application of expert system to process monitoring has been done before. Not all processes have had the technology applied, and some processes benefit more than others from the technology. The process monitoring applications can be split into two categories, those that have applied the standard features of commercial expert tools, and those that come about through research in different approaches towards process monitoring. The standard applications use limited knowledge representation architecture. The applications are built on rule sets and rule set firing controls, and in some cases, representation of the process equipment by objects. In general these applications offer no novel approaches to representing or executing process monitoring knowledge. Discussions on the applications express a moderate success in that they are "useful to the operators as a good guidance" (Lee et al., 1992). The draw backs that are discussed are similar. These include the need to develop special skills to maintain the system (Hardeveld, 1992), the inability to encompass aspects of the real world such as nonmonotonicity, and not being able to address or generalize knowledge to new situations (Cauvin et al., 1992). The applications have been worthwhile, but have not shown the success that the technology can offer.

Perhaps more pertinent to the thesis is the research process monitoring expert systems and the concepts that they have employed. There are a number of such projects that have been written about, however, no process monitoring research projects for the pulp process were found. A selection of five papers, each taking a different approach will be reviewed here. These approaches express a cross section of concepts towards representing and executing process monitoring knowledge. Some other less novel approaches can be found in the following literature: Wagenaar & Schrijnen, (1988); Thomson, (1988); Krijgsmann & Jager, (1992).

An Architecture for Real-Time Reasoning and System Control (Ingrand et al., 1992) This paper presents an architecture for expert systems that is being employed to diagnose problems for the NASA Space Shuttle's reactor control systems. This architecture and approach is the most novel and extensive of all the process monitoring systems reviewed. Diagnostic knowledge is represented in *knowledge agents* (task blocks) as procedures in graphical form. Each KA has goal and a context incorporated in it. The goal and the context are used to select it from a library of KA's when some purpose is at hand. The procedure in a KA is a cross between a petri net and a decision tree. It is executed one step at a time as long as the context stays valid. To progress across an arc (link) to the next graphical node, the goal of the arc must be met. The arc goal may be a logical statement or an action that must be performed. The KA's are executed by a *procedural reasoning system* (PRS). The PRS executes some meta-level KA that serves the purpose of establishing goals from information in a central database (*beliefs database*). A goal (monitoring requirements are in the form of goals) is presented to the PRS which searches through the KA library to find a suitable candidate. The PRS creates an intention stack for each goal that the system must contend with. The PRS executes a step from the current KA in each intention stack. If the arc statement (goal) of a KA can not be met, the PRS searches the KA library for a KA to deal with that goal. For example, if a manifold pressure is not high enough, a KA to examine the manifold is initiated. Eventually either all the KAs respond by achieving their goal, and the system is normal, or *primitive* level KAs are reached. The invoked primitive KAs are actions that the system can do, such as inform the operations personnel of the problem or to make control modifications. This system is based on a goal and subgoal architecture with goal achieving tasks in the form of graphical decision trees. These tasks can be applied generically to process components. The system is operated by a real-time multi-task executing shell that can focus the system by controlling the execution of each task at any time.

The MIP Project (Alaman et al., 1992)

The MIP expert system is applied to a petrochemical reactor in a refinery. It supplies information and recommendations in real time to help diagnose and make decisions on controlling the plant. The goal of the project designers was to build a knowledge architecture that was close to the human knowledge representation and was modular. The designed architecture is a four level tier. The bottom level represents the plant data and structure. It is implemented using frames. The second level is a piece wise simulator for various parts of the plant. A global simulator is not achievable. The third layer is local heuristic knowledge in the form of rules. This layer captures heuristic knowledge for diagnosing subsystems in the plant. The top layer is global heuristic knowledge implemented using an evidential reasoning scheme. The flow of information is in the up direction only, with higher layers accessing results from lower layers. The idea of the architecture is that data is accessed and manipulated at the lowest level. This data is used to provide simulated predictions of the operations by the second layer. The third layer uses the first two layers to diagnose problems through expertise captured as rules. The top layer relates anomalies of subsystems to the global picture. The system is developed using blackboard architecture; the reasoning module is an IBM product (TIRS); the user interface is an IBM hypermedia tool (AVC); the simulator was written in C code. This project was carefully implemented, and integrated respecting the plant management and operations staff concerns. Much of the success in the implementation is accredited to that. The paper reports an estimated 1% or better improvement in productivity due to the system.

Formentor (Pennings & Saussais, 1993; Wilikens et al., 1993)

Formentor is a safety oriented decision support system that employs a goal tree-success tree (GTST) model. This system is similar to the knowledge agent entities of the Space Shuttle system in that it consists of goal established decision trees. In Formentor's case though, the processing goes from the bottom of the tree to the top. Several main trees express goals that are desired to be met. The tree specifies all the sub-goals that must be met. The subgoals form their own GTST trees. The tree arcs in this case are logical relations, such as AND or OR, linking several subgoal nodes. The execution starts with the storing of incoming data into a data base. The lower nodes then evaluate a goal status. Rules may be employed to determine the status. These nodes may need to summon other goal trees. A status of bad, warning, or okay is determined and passed up to the next node which determines its status. The status and execution of goal nodes filter upwards. The worst status is always passed on. When the top is reached the final status is displayed. If the status is not okay, the system retraces the steps to find the failed goal nodes. linked to these nodes are textual explanations for countermeasures that can be taken. If these measures are successful, then the final state becomes okay. Formentor is described by the authors to be very useful within the scope of its intended application, that being to reason about the functioning of a system. They suggest that this system should be employed with other systems which reason about the behaviour and on the structure of the system. Formentor has been applied on the Infrared Space Observatory satellite and in nuclear plants.

EAGOL (Pople et al., 1994)

EAGOL has been applied to generating emergency operation procedures for a nuclear plant. EAGOL is based around a qualitative model that uses heuristics to express cause-and-effect relationships. This model is used to envision future outcomes. The system is modular with the lowest level being a behaviour analysis. This module processes raw data and looks for certain kinds of behaviour changes. Behavioural anomalies trigger the system to investigate possible reasons for the behaviour. This in turn invokes the system to evaluate possible outcomes. The novel aspect of this system is that it takes into account the automatic control moves of the system when deliberating on the outcome or suggesting an action. The knowledge of the emergency operating procedures is implemented in the system, however, these are overridden if the action is determined to result in another undesirable state. This system depends heavily on technical specialists to specify the system behaviour and to translate the knowledge to implement it into the system.

Designing Real-Time Knowledge Based System with PERFECT

(Sassen & Jaspers, 1992)

PERFECT is not so much an expert system as a knowledge base programming language. It takes the view of describing knowledge of a plant structure in the form of objects, and uses the hierarchy principles to express subsystems. Each object contains attributes such as external measured values. An object can have a hook to other units that can perform calculations or other forms of data manipulations. PERFECT allows the user to also create a rule based intelligent module that analyzes the knowledge objects for anomalies. The user creates another module with knowledge to generically search the subsystem hierarchies to find problem causes. This system then compiles the information into knowledge base configuration code for expert systems. Presently the system has been utilized to create COGSYS knowledge base code. A unique feature of the system is that it performs a pre-analysis of the knowledge base to ensure that there is sufficient knowledge to complete all reasoning tasks and that the execution of the reasoning can complete within a prescribed time period.

1.5 Thesis Definition

This thesis work addresses several problems that exist with expert system applications for process monitoring. Some of these problems are inherent in the present expert system technology. This work also is aimed at studying the significance of applying expert process monitoring to a pulp bleaching process. A prototype expert system was constructed and installed in a pulp bleaching process for this purpose. This prototype implements the proposed architecture thus provides practical feedback on the worthiness of the architecture. This section defines the problems that the thesis work is addressing, the objectives that the thesis is trying to achieve, and a criterion by to evaluate the prototype results to judge the success of the architecture.

1.5.1 Problem Definition

Inability to accurately represent all the necessary aspects of a domain: Present knowledge representation schemes only partially represent the diversity of a domain and can not account for all of its inconsistencies. This is especially true of those systems employing only one manner of representation. With present schemes, one continues indefinitely to insert addendums to account for new perceived variations of a domain. For declarative and rule base systems, the ability to continue to accurately expand and maintain the knowledge eventually becomes incapacitated. "It is impossible to axiomatize the world" (Schank and Slade, 1991). With other schemes, only specific aspects of a domain are represented. Present systems and techniques do not capture and represent the rich diversity of domain knowledge.

Inability to handle novel situations: Present expert systems do not have the ability to form solutions to handle new, and hitherto, unseen problems. Expert systems do not *think*; they do not reason how to solve a problem, nor do they generalize knowledge to form solutions. Present expert systems generally regurgitate captured knowledge of predetermined situations. Expert systems do not replicate the *thinking expertise* of a human. (Expert systems have many valuable and needed applications in the present form, but there is a greater need to have a system that can think. Process monitoring is such an application.)

Difficulty to implement knowledge: It is very difficult to express human knowledge in the manner of present knowledge representation structures, such as in the form of rules

(Schank and Slade, 1991). It is much easier for humans to recount their knowledge as episodes. Because of this difficulty a knowledge-base specialist is needed to acquire the relevant knowledge from a human and *translate* it to the form of the expert system. The expressing of knowledge through an intermediary source looses much of the content of the valuable knowledge. The translation of the knowledge to the foreign form, implementable in an expert system, dilutes the knowledge further. Present knowledge representation schemes are not in a form that is natural to humans.

Difficult to comprehend and understand the knowledge base after the fact: Present knowledge representation forms are hard to re-interpret and understand at a later date. Due to this, it is difficult to integrate future modifications and expansion of the knowledge base. Unpredictable and conflicting results can occur. Often a structure, such as a priority scheme, has to be adjusted to accommodate new fixes to the knowledge. The knowledge base can become stagnant due to the effort required to maintain it. This is prominent in the process industries where technical staff are already overloaded with work and have little available time to spend.

In general, expert system applications are not able to "solve intelligence problems", but address small niche problems (Schank and Slade, 1991). The cognitive thinking activities for monitoring a process require the employment of a wide range of mental abilities and must encompass a large knowledge domain. To provide useful assistance, an expert system must be capable of performing the equivalent intelligent activities.

1.5.2 Objectives

Given the problems that face expert systems applications for process monitoring, the following objectives were set for the theoretical thesis work. A knowledge representation architecture applicable to process monitoring was to be developed for an expert system which:

- is able to represent the *inconsistent* nature of a domain and adjust the knowledge content to account for the domain inconsistencies.
- is able to perform *predictive* reasoning; that is to reason in a manner to predict future events and domain conditions.
- represents *knowledge* in the form that a human consciously perceives and retains their domain knowledge. The architecture should embody many of the forms of knowledge representation that are inherent in a human's mind. The belief is that a computable system should be developed in the imitation of human's thinking, rather than forcing the human to have to think like a machine[1]. After all, the human creates the machine.
- represents and applies *experiences* (episodes) in their natural knowledge forms.
- represents and executes the *intelligent thinking* and thought processing abilities of a human. The architecture should encompass the ability to apply the same variety of thinking activities towards reasoning that a human can. Further, the architecture should allow the reasoning activities to focus on a problem similar in manner to that of a human.
- has the ability to apply the intelligent thinking knowledge generically to a variety of situations.

[1]This notion was expressed by Dr. Rao, University of Alberta, in his artificial intelligence graduate studies course.

It is felt that by replicating the knowledge and thinking structure of a human in an expert system, many of the discussed problems of representing knowledge and of being capable of dealing with problems would be eliminated. The represented knowledge in an expert system would then be as good as the human's ability to perceive the world. It is felt that knowledge could be implemented far easier because it could be expressed in a natural form by humans. The objective of this thesis is to try to narrow the gap that now exists between present expert system knowledge representations and how knowledge is percieved by a human. Note, however, that this thesis does not hope or profess to completely close that gap, but is to offer an advancement towards doing so.

1.5.3 Prototype Issues

A prototype is a working model established to evaluate a system's worthiness. The issues that are faced when constructing a finished product are equally assumed when building a prototype system. Several papers outline a list of issues and problems faced with real-time expert system installations: Arzen, 1992; Clements & Preto, 1993; Laffey et al., 1988; Rivera et al., 1993; Williams et al., 1991. Not all of these issues can be addressed due to the developmental nature of a prototype. The prototype system developed for this thesis does meet many of the issues. This section lists the issues that were considered when constructing the thesis prototype system:

- The system must implement a sufficient portion of the proposed knowledge representation architecture to obtain a proper evaluation of the architecture.
- The system must run continuously without undue loading on the computers and yield a reasonable performance. A reasonable performance is to be able to execute the knowledge-base and reach valid conclusions in a timely fashion.

- The system must be integrated into the mill computer systems and information flow. The system must communicate and synchronize with other real-time software processes. This is especially true with the thesis prototype system as both the data interface and the user interface are established through communicating with a commercial process management information system software package.

- The system must be able to access and handle real world/real-time data.

- The system must produce reliable and consistent results as it is being implemented in a production mill's control room. The system must not produce redundant and cycling results, as such nuisances would quickly deter its use by mill operations staff.

- The system must be self explanatory to use, and require a limited number of steps to operate it.

1.5.4 Evaluation Criteria

Establishing experimental evaluation criteria for a knowledge representation architecture is difficult because quantitative measures do not exist. How does one go about evaluating the ease of implementing knowledge, or the ability to more completely represent a domain? However, a system implemented after the knowledge representation architecture must *at least* perform as well as any other expert system in the same situation. This leads to the problem of how to measure performance in an uncontrolled environment. There are few quantitative results documented for other process monitoring applications, and no pulp bleaching applications were even found. It takes several months, if not a year, to build up a sufficient case study to produce actual cost savings figures. Even then, the events at two different installations are totally uncorrelated making a comparison meaningless. The state of the art is not such that mills presently have expert systems by which one can do side-by-side comparisons of performance. Testing an expert system in a controlled laboratory environment proves that a system works, but does not test its robustness when faced with real world situations. Lacking quantitative measures, one must look at qualitative feedback and perhaps the intuitive response that is invoked by the concepts. Taking this approach, the following is a list of qualitative criteria that can be used to help evaluate the architecture proposed in this thesis. The evaluation is based on the theoretical objectives of the thesis and on the requirements for process monitoring expert system applications:

- The prototype system implementation of the architecture must work and satisfy the prototype issues.
- It must be able to accurately detect process monitoring situations and provide information on how to handle the situation as a result.
- The detected process monitoring situations and the provided information must be considered beneficial by the process operation personnel.
- The system must be capable of predicting situations prior to their occurrence.
- The system must handle the inconsistent nature of the process. One manner in which inconsistencies manifests in the pulp bleaching process is the changes in the process behaviour with changes in operating conditions, such as production rate, or chemical ratios, or grades.
- The knowledge-base must be capable of having *experiences* or episode knowledge directly configured into it.

- The system must be such that ıt is easy to express the knowledge to configure it. A test is if non-technical personnel can express meaningful information, ir configuration form, for it.

- The system must capable of equivalently representing how the operations staff views their process monitoring activities and its domain.

2
A New Approach to Knowledge Representation

Present methods and schemes are deficient in their ability to represent and execute knowledge in an expert system. These deficiencies are felt to have limited the effectiveness of applications of expert system technology and would prevent the successful development of a process monitoring system. The deficiencies stem from fundamental weaknesses within the theory that forms the foundations of expert systems. These weaknesses needed to be addressed so that a better manner of representing and processing knowledge could be developed. Work towards this end was carried out resulting in a new architecture for the construction of an expert system. This architecture is felt to be a general foundation that represents many of the facilities exhibited by a human mind. The conception of this architecture, however, was driven by the requirements for process monitoring. The concepts that comprise it, thus, were developed under the context of process monitoring. The full extensibility of the architecture to other realms of mental activities has not been explored.

In order supplant the present weaknesses of expert system technology, a study was conducted along several paths. The goal of this study was to discover and develop effective means to replicated human mental activities in a computable system. To achieve this, human cognition was examined by the observation of human thinking behaviour, especially that used in the practice of process monitoring. Cognition was further explored by self contemplation of how knowledge manifests in the human's mind. The studies resulted in concepts that were verified, supported, or modified, through research of psychology theory. In the following sections the derived concepts will first be presented. An investigation of various phychology theories to find these concepts will follow. The derived concepts were compared with present knowledge system related technologies. The comparison resulted in a selection of meaningful and viable technologies. These technologies were enhanced and acutely interwoven to align with the concluded concepts. The work resulted in a unique architecture, though with a face that has many recognizable traits, that was felt to better represent a human's mental activities.

This chapter presents the derived concepts, the required technological enhancements and the proposed architecture resulting from the study. Sections 2.1 to 2.5 discuss the concepts and the technology necessary to realize them. Section 2.6 draws all the concepts together and describes an overall architecture to represent and process knowledge.

2.1 New Perspective on Representing Human Thinking

The purpose of an expert system is to capture and execute the expertise of a human. A human's expertise comprises not only what he knows, but also his ability to think and reason. Expert systems built on present perspectives of human thinking lack many of the abilities that humans have to "think." To be a viable aid to a human, an expert system must be able to "reason through" problems similar to a human. It was realized that a better understanding of human thinking was required to develop these abilities so that they could be represented in a computable expert system.

Human reasoning behaviour of problem solving or situation handling was observed to amount to a process of thinking tasks (a *thought process*). *Thinking tasks* are applied sequentially to work towards obtaining a desired goal or a possible state from an initial state. Intermediary decisions are made throughout the flow of the *thought process* to evaluate the nearness of achieving a goal and to decide the next step to take. The next step could be the employment of a *thinking task* or the summoning of another *thought* *process* to meet some new challenge. A *thinking task* is a procedure of actions that performs a thought dependent task. *Thinking tasks* generally embody the following actions: the acquisition of information, the issuance of commands, and the execution of *thinking activities*. The acquisition of information is chiefly the recall of knowledge from memory, but could be a joint activity with an issuance of a command to acquire external information. The issuance of a command is the interface of the human mind to the human's physiological mechanisms (external world). A *thought process* can be summarized as an interwoven combination of *thinking tasks* that are incrementally applied to search or work towards achieving a desired or possible state.

For an example of the *thought process* concept, regard the task of troubleshooting a stalled automobile engine (Figure 2.1). The *thought process* could consist of the following *thinking tasks*: check if there is fuel flow, or air flow, or compression, or ignition, or a generally visual check to see if all is in order. The *thought process* would consider the external temperature and the sounds issued forth when attempting to start the car to decide which *thinking task* check to perform first. The *thinking task* of checking for ignition would consist of issuing the command to hold the spark plug lead slightly away from the engine body to acquire information on ignition power (see figure 2.2). From this information the *thinking task* would conclude whether there is sufficient power to cause ignition. The *thought process* then would use this information in evaluation if this is the problem source or not.

The execution of *thinking activities* is done by *thinking mechanisms*. It was observed that humans did not rely on a single form of thinking mechanism but had a variety of thinking abilities to draw on. These abilities, or *thinking mechanisms*, were inherent to a human's mind and differed in acuteness between people. Such abilities include, for example, inferring, reasoning with uncertainty, evidential reasoning, reasoning based on previous

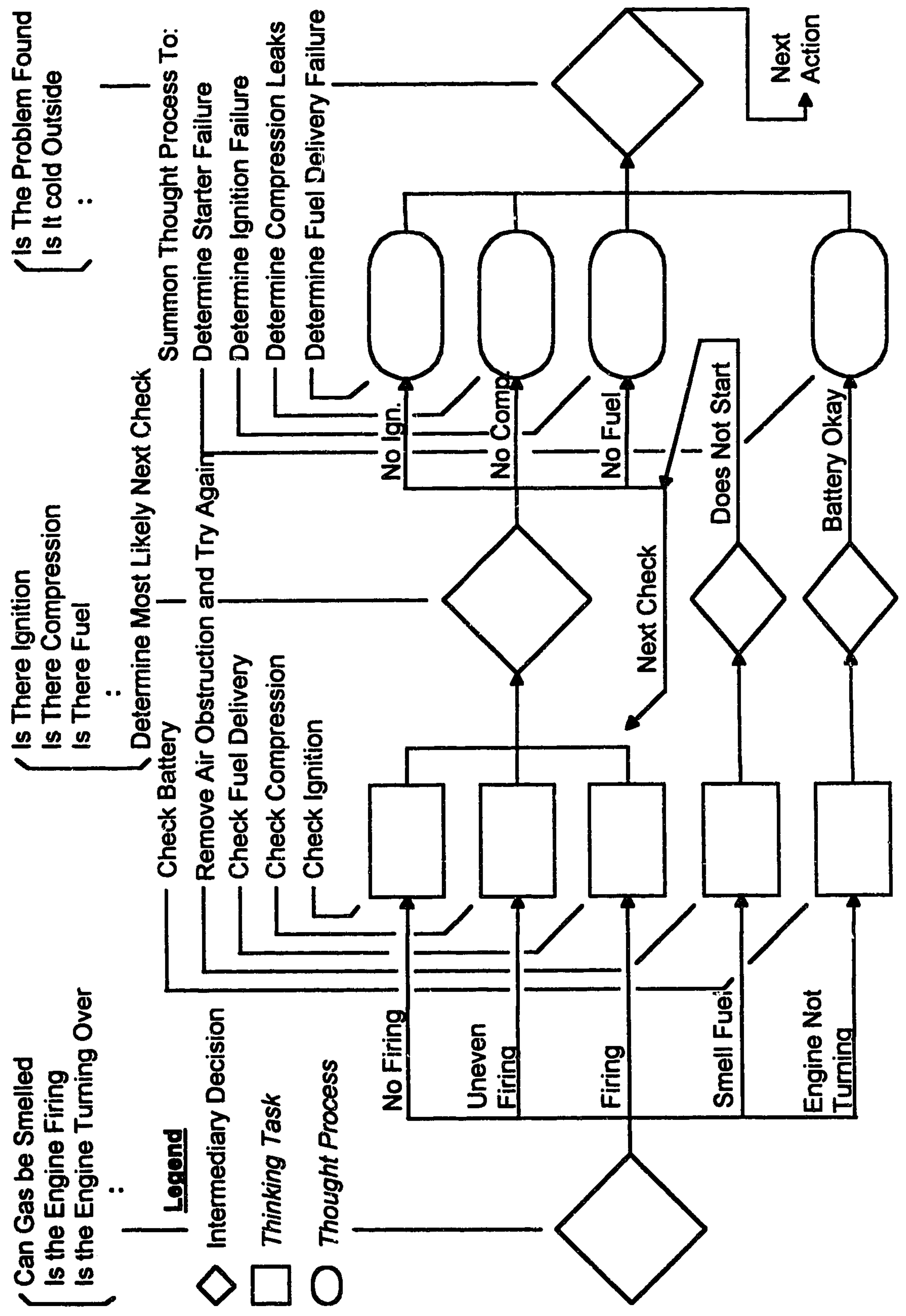


Figure 2.1: Example *Thought Process* Structure

Thinking Task for Checking the Ignition of an Engine

```
Identify plug:
    • Call on a set of inference rules to process visual information to identify the engine spark plug
Remove plug:
    • Recall from knowledge about this type of engine how to remove the plug
    • Issue commands to remove a spark plug
Ground plug:
    • Recall from knowledge of spark plugs where the grounding locations are
    • Issue commands to ground the spark plug on the engine frame
Observe spark:
    • Call on visual recognition experience (Neural Net) to identify a spark and to quantify the intensity of the spark
Evaluate Condition:
    • Call on a set of inference rules to determine the status of the ignition check from the recognition experience results. The thinking activity would decide on:
    •• the success of the test
    •• the need to do a re-test; the conditions that must be changed (find a better ground on the engine block)
    •• the ability for the spark to ignite a fuel mixer given the present context (ambient temperature)
    •• the need to check other plugs given the sound of the engine with a single plug removed
```

Figure 2.2: Pseudo Code Depicting a *Thinking Task*

experiences, pattern recognition, and various knowledge searches. The *thinking mechanisms* appear to be virtually separate from knowledge. The available *thinking mechanisms* of a human are employed on the relevant knowledge of the present domain context. During a *thought process* sequence, *thinking mechanisms* are activated by the *thought process* to perform the intermediary evaluation decisions. *Thinking mechanisms* are also activated by *thinking tasks* to perform the required *thinking activities*. In the automobile troubleshooting example, *thought process* knowledge of how to infer the most likely failure of the engine (fuel, air, compression or ignition) would be processed by an inference mechanism, resulting in a decision of what *thinking task* check to do first. In summary, a *thought process* contains thinking knowledge on which *thinking mechanisms* are employed to process it. The decisions it makes are the *thinking tasks* to be employed to obtain a determined goal.

The *thought process* can initiate a *thinking task* that executes a *thinking tool* in a proper context. A thinking tool is an aid that can be used in trying to acheive some reasoning goal. Humans typically employ thinking tools to simplify the problem domain. Thinking tools include a myriad of mathematical manipulations such as genetic algorithms, optimization techniques, statistical evaluations, calculus representation, etc. Symbolic manipulation, graphical expression, mapping and visual tools are also employed by humans. A *thought process* can determine if thinking tools need to be employed. (The execution of a thinking tool can be guided by a *thought process* or it can be external equipment such as a computer.)

Thought processes appear to be self contained entities in the human's mind. Each of these entities contain knowledge of how to seek out or reason through a problem or situation. It also contains knowledge of how to evaluate the current thinking state's proximity and trend towards the goal. A *thought process* serves a *purpose*. It has a *reason of being* and embodies an intrinsic problem or situation for which it is applicable. The process of thought can be applied to satisfy a variety of goals. Each particular goal shapes the seeking and evaluation process. A *thought process* is *aware* of the goal. (The goal of the automobile troubleshooting *thought process* example would have a goal of determining the cause of failure.) *Thought process* entities are encapsulations of the human's knowledge of how to think about a problem or a situation. The human mind is felt to contain many *thought processes* that can be applied to guide the reasoning of manifested situations and problems.

Human reasoning forms a hierarchy of *thought processes*. A current *thought process* often determines that a particular thinking action is required to be undertaken. This action itself may be a problem or a situation that must be dealt with by another *thought process*. The current *thought process* relinquishes itself to a newly summoned *thought process*. The result of executing the summoned *thought process* may be to initiate a further *thought process*, or to provide the initiating *thought process* with acquired information or a result. The reasoning process thus becomes a hierarchy of applied *thought processes*. Each *thought process* supplies some purpose for the achievement of the overall goal. The form of the hierarchy is determined dynamically, as each *thought processes* determines its needs to achieve the overall goal. Using the automobile troubleshooting example, consider that case where the ignition check concludes an ignition failure. The trouble shooting *thought process* then would initiate a *thought process* of how to trace the ignition circuitry. This *thought process* could initiate a *thought process* that troubleshoots a magneto (Figure 2.3). In general, the reasoning process begins with more deliberating *thought processes* and tends towards very specific task related *thought processes*.

Human reasoning occurs on one of at least two planes of thought. Reasoning to seek a problem's or a situation's solution occurs on the first plane. On this plane a *thought*

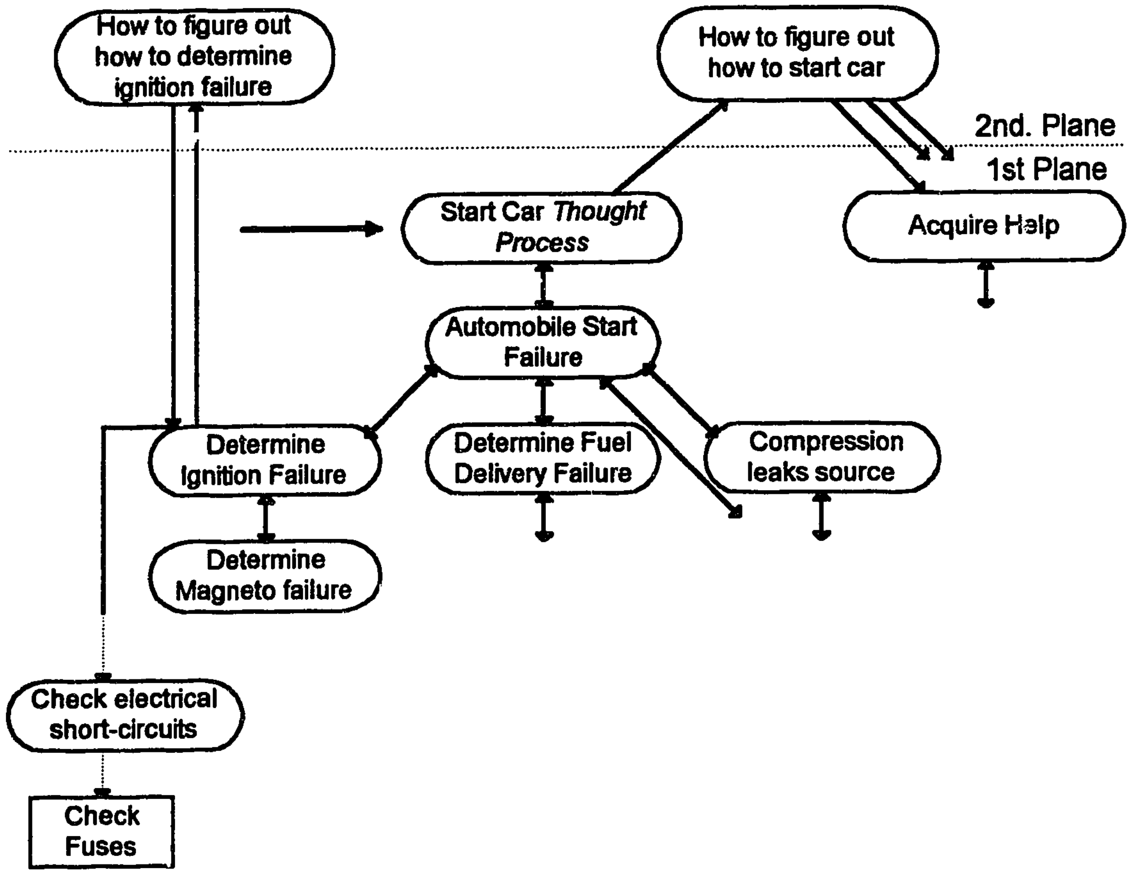

How to figure out how to determine ignition failure
How to figure out how to start car
2nd. Plane
1st Plane
Start Car Thought Process
Acquire Help
Automobile Start Failure
Determine Ignition Failure
Determine Fuel Delivery Failure
Compression leaks source
Determine Magneto failure
Check electrical short-circuits
Check Fuses

Figure 2.3: Example of Hierarchical Deployment of *Thought Processes*

process makes decisions as to which *thinking tasks* or *thought processes* should be employed on the problem at hand. The knowledge of how to go about seeking the solution is known; the *thinking methodology* is predetermined. The actual reasoning path that occurs (which tasks are employed), is dependent on the context of the problem or situation and is unknown at the onset. The path can take one of any route through a known possible reasoning task (*problem*) space. *Thought processes* at this plane are using a determined methodology to search for solutions that achieve a desired goal.

On the second plane, reasoning is being performed to determine *how to reason* a problem or a situation. Second plane reasoning uses the goal and situation knowledge embodied in a *thought process* when seeking applicable solution methodologies. Knowledge of how to go about reasoning a problem or situation is yet unknown; the thinking methodology is not predetermined. This occurs when a new problem is faced or a first plane methodology fails to produce an acceptable result. *Thought processes* at the second plane of reasoning make decisions as to which *thinking task* or *thought process* to employ. The employed task or process has the purpose of trying to determine *how* to reason the problem or situation. The results of second plane reasoning are *thought processes* (thinking methodologies) or *thinking task* to apply as first plane reasoning. The result may also be to specify an alternative goal for a problem or situation that is deemed more plausible. As an example, consider when the previous automobile troubleshooting *thought process* fails. A second level *thought process* becomes initiated to question "what to do?" This *thought process* may come up with the suggestion of phoning a friend who lives close by for help. *Thought processes* on the second plane search for methods that can seek a solution.

Experience plays a major role in reshaping the human thought process. A *thought process* dealing with a problem or situation for which experience is lacking requires more searching for successful *thinking activities* that will lead to the desired solution being found. The *thought process* involves more intermediary decision making. Experience has the effect of streamlining the *thought process* by eliminating and automating the decisions of which *thinking tasks* to employ. The thinking methodology becomes more defined and definite, discarding the ineffective methods and strengthening the successful ones. General *thought processes* are replaced by experience shaped *thought processes*. Turning to the automobile *thought process* again, a comparative experience *thought process* may, from the sound of the motor, go right to checking the fuel pump and ignoring the exploratory problem testing. The experienced shaped *thought process* develops into a predetermined methodology. The shaping process, on the other hand, makes *thought processes* more specific in purpose and narrow in their applicability. If the existing experience of a situation is minuscule then a second plane of reasoning *thought process* will probably be needed to discover a thinking methodology to apply. This will lead to a thinking methodology being suggested, applied and tested. Thus, experience creates new *thought processes* from the second plane reasoning of a problem or a situation. A second plane *thought processes* can also be shaped by experience developing more efficient methods of finding problem or situation solving methodologies.

Thought processes are applied to *generic* situations and problems. The situation or problem may vary in content but the applied thinking methodology remains the same. For example, the thinking methodology contained in the automobile troubleshooting *thought process* can be employed on a Ford that stalled in an intersection, or a Chevy that has been sitting idle for weeks, or a truck that will not start in the morning. The *thought process* embodies the knowledge of how to think; it contains no specific knowledge of the subject it is thinking about. This can be restated that the *knowledge of how to think is decoupled from knowledge of what is being thought about.* Embedded in the *thought process* is the knowledge of how and where to get subject required specific knowledge. When a *thought process* is initiated, it becomes *aware* of the subject it is thinking about. If the rudimentary goals of a situation or a problem match those serviced by a *thought process*, and the structure of the situation is similar to the applicable scenario of the *thought process*, then the *thought process* can be employed. The *thought process* acquires the necessary generic information from the context of the current situation or problem. This is especially true of domains that are very standardized, such as gas engines. An understanding of the norms of the domain is developed in the *thought process* that is used to quickly acquire the necessary information of the situation or problem. Note that searching for *thought processes* that have similar features to a problem or a situation is one possible way that second plane reasoning derives potential methodologies to apply to new situations or problems. To see this, consider an automobile mechanic faced with a troubleshooting turbine engine. He/she may deem that the turbine engine has enough similar traits to piston engines that he can try to employ his/his automobile troubleshooting *thought process* knowledge to the situation.

The initiation of human thought appears to be from a stimulus. This stimulus is referred to as a mental cue (Bellezza, 1989). Mental cues can be generated by external conditions or by other reasoning activities. The latter is demonstrated in the hierarchy concept where one *thought process* may initiate another. A process of reasoning usually begins with an external stimulus. For a human, it may be that they have *noticed* something. The stimulus provides the initiated *thought process* with contextual information about the stimulus source. For instance, the car troubleshooting *thought process* assumes the context of the car when the human notices that the engine will not start. This is how the *thought process* becomes aware of its subject. The goal can be provided with the stimulus or may be inherent in the initiated *thought process*. For instance, a quick movement initiates a *thought process* whose intrinsic purpose is to determine if there is potential danger. To begin a *thought process* there must be some mental cue.

These previous concepts on human thinking fit together to form an overall perspective which is epitomized as follows. A *thought process* to deal with a problem situation is initiated by a mental cue. The *thought process* is made aware of the desired goal and context of the problem or situation. *Thinking mechanisms* are activated by the *thought process* to process the knowledge of how to handle the situation or problem. The *thought process* makes a decision as to the next step to take. A *thinking task* or another *thought process* is initiated in the mind. Generic information and knowledge are gathered from the contextual knowledge of the particular situation. *Thinking activities* are performed for the activated *thinking task*. After the *thinking task* is complete, the *thought process* evaluates the present status versus the intended goals. If another *thought process* has been initiated, then it proceeds until it has completed its purpose or is waylaid by another *thought* *process*. This continues until the originating *thought process* terminates. (It is felt that the human mind only performs one thought process at any moment, and at most a few concurrent thought processes. However, a computable machine could execute many thought processes concurrently, or even in parallel) If the thinking of the situation or problem has reached an impasse, then a second plane *thought process* becomes initiated which deliberates on how to go about dealing with this problem. If it is successful then a new methodology becomes evident and is followed. If no methodology is forthcoming, then the target goals for the situation or problem may be restructured and the previous methodologies re-applied. Experience plays the role of focusing the *thought process* and reducing the need to debate the necessary steps. Each *thought process* manifests as encapsulated entities in the mind. The human mind is made up of countless *thought process* entities that combined to form the knowledge of "how to think."

Almost all the presented concepts on human thinking behaviour can be found within the vast body of psychology theory. Many of the concepts have been stated in a general theory of problem solving that has been developed within the body of information-processing theories of cognition (Feigenbaum & Feldman, 1963; Anderson, 1976, 1983; Newell & Simon, 1972; Newell, 1982). This theory proposes the notion that problem solving is a process of applied operators with intermediary goal evaluations that searches for a solution.

> "The application of an operator to the initial state of a problem produces a subsequent problem state, and the application of a sequence of operators constitutes a solution path. The *problem space* is thus the space of all possible states generated by the application of all operators that can conceivably be applied to each state. Problem solving, then, by definition, is the *search* of the correct sequence of operators, from the initial state to

> processes for setting goals and employing general methods for the solution of a problem, thus controlling the search process. Strategic knowledge is procedural in nature. One of the basic functions of these general strategies is to provide the problem solver with criteria to evaluate a problem-solving step. Under the control of a *means-ends analysis*, for example, the evaluation consists of comparing the current state to the goals state, and actions are selected that reduce the difference between the current state and the goal state." (Reimann & Chi, 1989).

The concept that problem-solving is a process of cognitive thinking operations has been expressed by others as well, Du Boulay (1989), Mayer (1989), Gordon (1989), Cooke (1989). A Gestalt theorist postulated a similar theory in 1945 (Duncker, 1945). Rumelhart (1975) introduced the concept of *Schemata*. A *schemata* is a structure within memory that embodies expectations about objects, situations, event and actions (Rumelhart & Norman, 1983). Chi identified schemata as the knowledge representation that encapsulates the knowledge of a problem and that these schemata could fashion themselves in a hierarchical fashion (Chi, Feltovich and Glaser, 1981; Reimann & Chi, 1989). This work verges on the concept of encapsulated *thought processes*, though the schemata were not integrated with an actual problem-solving *thought process* to form a single entity. Work done by Laird, Rosenbloom and Newell on the comprehensive SOAR goal hierarchy problem-solving architecture exemplifies the notion of a hierarchy of thinking processes. SOAR also demonstrates a reflective problem solving behaviour. (Laird, 1984; Laird & Newell, 1983; Laird et al., 1986; Laird et al., 1987; Rosenbloom & Newell, 1983). SOAR architecture is based on applying operators to move from one state towards a goal state within a problem space. A problem solver selects applicable operators for a problem space and sequentially applies them in attempt to reach a desired goal. The success of each applied operator is evaluated using a means-end analysis. Both the search for which operator to apply and the evaluation of the operator's success are problem spaces themselves. SOAR generates goals and problem spaces to handle these needs. If an operator has insufficient knowledge to meet a goal, then a subgoal and its problem space to attain the required information is automatically generated (referred to as universal subgoaling). SOAR applies learning through chunking which substitutes the efficient and successful productions of a particular problem space goal for the complex goal procession activities that normally occur. This form of learning (chunking) is an example of the discussed refining of thought processes by experience. SOAR architecture replicates many aspects of the discussed human thinking behaviour. Dealing with a problem at a higher plane of reasoning has been noted by Mayer (1989) and has been expressed by Gestalt psychologists (Katona, 1940; Wertheimer, 1959).

> "Routine problems require 'reproductive thinking'--applying already known solutions procedures to the initial problem state--while nonroutine problems require 'productive thinking'--generation of a creative or novel procedure" (Duncker, 1945).

The idea of *knowledge that* being different from *knowledge how* has its roots with Ryle (1949). Buchanan & Smith (1989) noted this when stating that there is a popular distinction between *how-to* knowledge and *what-is* knowledge. This thesis takes a further distinction in that *how-to-think* knowledge is separated from *what-is* knowledge, which translates to the decoupling of thinking from generic domain knowledge. The cueing of thought has been explored at lengths by Bellezza (1988, 1989). Cooke (1989) expressed that experts use features of a problem "to cue other relevant knowledge." The effects that experience have on shaping thinking have been explored by Fitts & Posner (1967), and later updated by Anderson (1983, 1987), in their theory of *stages of skill acquisition*. The theory views the first stage as a *cognitive* stage where domain knowledge is operated on by "*domain-general*" procedural knowledge. As competence increases, a second stage, *associative*, is reached with "*domain-specific*" procedures.

> "The need for operating on declarative knowledge gradually becomes bypassed." "The action is automatically invoked, circumventing the longer and more tedious process of retrieving declarative knowledge and applying general productions to it." (Gordon, 1989).

In the third, *autonomous*, stage the procedures become highly automatized and specialized towards particular types of situations. (Charness & Campbell, 1988). Despite extensive research by the psychology field on the behaviour of thinking, there appears to be an absence of work on the mechanisms of thinking. Pattern recognition and inferring are acknowledged but little work towards detailed exploration seems to have been done by psychologist. The field of education has provided much theory of human thinking behaviour through studies to understand how humans learn. Work by Boyd (1972) is exemplary of this. The presented perspective of human thinking behaviour closely resembles Boyd's perspective of the *process of inquiry*. Although stated in different terminology, and perhaps more general, the field of psychology concur with the concepts presented in the thesis. This thesis offers an assemblage of the concepts to depict a full perspective of the behaviour of human thinking.

Contrary to the psychology field much work has been done by the AI field to replicate *thinking mechanisms*, but little work has been done in assimilating overall thinking behaviour theory. *Thinking mechanisms* have taken technical forms in artificial intelligence applications such as inferencing by resolution, production inferencing, temporal inferencing, uncertain inferencing, evidential inferencing, fuzzy logic, forms of searches, etc. The extent of thinking behaviour in most expert system applications is a procedure or a modularization of steps to process a particular problem. Two notable differences are applications of the SOAR architecture (Laird, 1984; Laird & Newell, 1983; Laird et al., 1986; Laird et al., 1987; Rosenbloom & Newell, 1983) and the Space Shuttle with its real-time architecture (Ingrand et al., 1992). SOAR, which was discussed previously, embodies many features of human thinking behaviour. The Space Shuttle expert system architecture is similar to the SOAR architecture. (See Section 1.4 for a description of the Space shuttle expert system.) The knowledge agents of the Space Shuttle system resemble the operators of SOAR, both are employed to meet determined goals by a procedural reasoning system (problem solver). The knowledge agents are much more complex however, in that they incorporate more intelligence and take the reasoning through multiple states: operators only move to a neighbouring state. The typical knowledge agents of the Space Shuttle expert system are similar in nature to *thinking tasks*, and the meta-level knowledge agents to *thought processes*. When the Space Shuttle system is presented with a problem (goal) it calls upon meta-level knowledge agents to determine the goals that need to be processed for the problem at hand. The system then searches for applicable knowledge agents to address those goals. These knowledge agents can result in new goals that must be met resulting in a hierarchical reasoning process. This manner of reasoning exhibits some of the aspects of the human thinking behaviour discussed in this thesis.

A representation formed from the presented concepts of this section is felt to replicate the general thinking behaviour of a human. The value of it is that it acknowledges the need to reason through a problem or situation, and that many types of *thinking activities* are brought to bare when doing so. It sees the integration of thinking and external activities as a requirement of solving a problem or situation. It recognizes that a thinking process is employed to deal with unknown situations and problems. Overall, it describes a representation that can house the vast breadth of human's cognitive skills. Lastly, it surmounts one of the greatest barriers to AI development in that this representation is implementable in a computable expert system.

2.2 A New View of the Application of Human Experience

A vast portion of a human's knowledge is derived from experience. This knowledge plays a paramount role in the reasoning and decision making activities that a human performs. It was felt that such an essential aspect of the human mentality should be understood and incorporated in an expert system architecture. A study was done to comprehend the nature of experience. Among other conclusions, it was noted that there are various manners in how experience is employed by humans. These manners of experience are representable in, and employable by, a computable expert system. Together, these manifestations form a new view on how experience is applied by humans. The following discusses the concepts that were concluded on experience.

As a first step in understanding the nature of experience it was deemed necessary to examine what experience is. By definition *experience* is "knowledge or skill gained by actual observation, activity or practice in doing something" (Oxford American Dictionary, 1980). Experience can be classified as a form of knowledge different from book learned knowledge. By experiencing new situations one gains new knowledge. Experience is also seen to be knowledge which has been *refined* through repeated encounters of similar situations. Almost all knowledge, whether gained by experience or teaching, is further refined through experience. Thus a large portion of a human's knowledge takes the form of experience.

Experience knowledge has a different nature from taught knowledge. Experience tends to be more exact and specific about its subject. It is noted to be a more applicable and reliable form of knowledge.

> "Book learning is considered not to be as reliable and not as strong as knowledge gathered through direct action." "Experiential knowledge has been 'field tested', that is improved, selected, and tested in action." (Prerau, 1989).

Experience is also thought to be an automatic knowledge, circumventing the requirement of reasoning domain knowledge (Fitts & Posner, 1967; Anderson, 1983, 1987). Experience knowledge appears to be relied on unquestioningly by humans. Further, if experience is present, it usually is sought out first by a human. This can result in the problem of over generalization and errant perception. In most cases, however, experience knowledge is a more effective and efficient form of knowledge than "book learning".

There is a distinction between *expertise* and *experience*, and it is worthwhile to clarify the difference. Expertise is noted to be a knowledge that embodies more value; it is a more capable knowledge. A human who possesses expertise at something is said to be an *expert*. Expertise is developed by the *adept employment of cognitive abilities* to the refinement process of knowledge through experiences (Schank & Slade, 1991). Thus, expertise is essentially of the same form of knowledge as experience. However, though all people retain experience, only some develop significant expertise. Expertise is quality experience. Representing expertise in an expert system is done by developing a representation for experience. The acquiring of expertise is dependent on the selection of scurces during the knowledge solicitation process.

The next step was to examine how experience is employed by humans. This lead to the observation that at least four distinct types of experience are engaged for cognitive activities. These forms can be described as, experience *about situations* (**situational experience**), experience of *how to recognize* (**recognition experience**), experience of *what to expect* (**expectational experience**), and experience about *how to think* (**situation handling experience**). For example, take a car on a snowy road. From experience, a driver would have the expectation that a snowy intersection would be icy on an extremely cold day (due to temporary melting by stopping cars). This exemplifies *expectational* experience. If the car began to slip, the experienced driver would immediately recognize this from the symptoms (*recognition*). When the car was slipping, the driver would know that his car could rotate or slide across lanes (*situational*). When the car started to slide, the experienced driver would know to pump the brakes gently and not to over steer (*situation handling*). These four experience types are deemed to play significant roles in enabling a human to comprehend and function within the complexity of the real world.

Situational experience is knowledge *of* and *about* a situation of a domain. The developed knowledge of a situation was observed to be directly applied to similar situations. If a reasonably equivalent situation is identified, then the experience of and about the situation is initially assumed to pertain to the new situation. This appears to apply for partial similarities as well where the equivalent aspects of the situation are assumed pertinent. For an example in the industrial domain, regard the activities of process monitoring. Envision a process upset situation which has happened before. Previous situational experience knowledge of this upset could comprise the probable cause of the upset, how the upset develops, the effects of the upset on the domain, and the potential outcomes if the upset is not handled. This knowledge will initially be assumed valid for the new occurrence, possibly quickening the response to the upset. Situational experience plays the role of providing memory of functional knowledge about domain situations.

Recognition experience is knowledge of how to recognize something. Humans are very adept at recognition. Human's abilities span from being automatic actions, such as vision, to rational actions, such as the contemplation of similarities between two subjects. The refinement process of experience appears to develop a knowledge of the key indicative traits or symptoms to identify something. Also developed, is a knowledge of how to apply the traits towards making an identification. Often, it was observed that the salient traits are used to identify known things by directly matching them with those of the subject being identified. Recognition experience is a quicker, more accurate and more acute form of recognition knowledge. In process monitoring, recognition experience is employed to identify an upset situation. In the case of a plugged stock washer of a pulping process, the knowledge may include such indicative features as decreasing recirculation flow, increasing headbox pressure, and a high stock consistency. Recognition experience plays the role of providing an efficient means to distinguish various entities within a domain.

Recognition experience is closely coupled with situational experience. A situation is *recognized*. The knowledge to recognize appears to be embedded in the situational knowledge. Thus, situational experience encapsulates recognition experience knowledge as well as its own. Embedded recognition experience provides the knowledge to identify process upset situations.

More than one purpose of applying recognition experience was noted. Recognition is applied to perceive the *existence* of something, as has been discussed. Humans were also observed to apply recognition experience to perceive the *oncoming* of something. This is a significant notion. What this implies is that human expertise can *predict* situations, even if the basis of the prediction is not obvious. If one assumes that neural nets form the human mind's physiology, one can expect humans to be able to detect 'nearness' of situations, since this is a noted property of neural nets. However, another aspect that comes into play is temporal relationships. It was seen that a human patterns the temporal relationships of the environment to develop temporal cues to predict the occurrence of something. As an example of a temporal relationship, let's return to the icy intersection. You judge that an oncoming car will be entering the intersection. Your car begins to slide. From the temporal relationships of the vehicles' motions, you may recongize that a collision could occur. In process monitoring, predictive recognition experience is employed to detect oncoming undesirable process situations. A high stock consistency may predict, for instance, the possibility of a plugged stock washer.

Recognition of real world entities has an inherent dynamic nature. A thing being identified can remain the same, but the characterizing traits may change. A person may appear with a beard on one occasion, and without on another. A stock washer can plug, but it is dependent on the consistency of the stock, the slice opening of the headbox, fluxuations in pressure and flow, and other fluid flow and operating properties. Human recognition knowledge adapts to variations in circumstances. The knowledge of how to dynamic recognize an entity is integrated in the recognition experience.

Expectational experience comprises the knowledge of expectations a human has about a domain and its behaviours. Expectations are derived from continual exposures to a domain environment. It was observed that humans correlate patterns that are inherent in a domain into a set of *norms* or *beliefs* about the environment. If the circumstances for which the norm or belief has meaning match those perceived to be occurring within the domain, then the norm or belief is assumed valid. A salty sailor, for instance, can examine the colour of the sky and state whether to expect a storm or sunshine ("Red sky at night--sailors delight, red sky at morning--sailors forewarning"). Expectational experience plays the role of providing a means to anticipate the domain.

The anticipation aspect of expectational experience appears to be a fulcrum from which humans detect problems. If the present beliefs do not agree with the observations of the domain, then a mental alarm is set off. The alarm is a mental cueing of a thought process to begin evaluation of the significance of the discontinuity. This appears to be a major form of problem detection. This form is very evident in process monitoring. Process operators become aware of problems by mentally comparing the observed process conditions with their expectations derived from years of operating the process. A sensor reading of 3.0 Kg/l may be an expected value for stock consistency; a value of 5.0 Kg/l would indicate a problem. (Note--this is different from a process alarm which indicates values **outside** all expected operating limits.)

Like recognition experience, expectations of a domain were observed to be dynamic. A domain is not static, (though it should be in the *chaotic* sense of stability). The expectations of a domain will vary in accordance with other conditions within a domain. A process, for instance, may be expected to operate at 3.0 Kg/l stock consistency for a certain grade, where it is expected to operated at 3.6 Kg/l for a thicker grade.

Situational Handling experience is knowledge of how to handle a situation to bring about a desired goal. A situation can be a problem that requires reasoning to determine a solution or it may be a predicament that needs corrective actions to be chosen. When humans are faced with a situation they search for ways to deal with the situation in alignment with the desired goals. Through trial and error, they develop a successful methodology to process a situation. A methodology was noted to often be developed even when explicit knowledge of the domain was lacking. This methodology is an experience developed *thought process*. The *thought process* was observed to thereafter be recalled and directly employed to deal with similar situations. The requirement to cognitively search for a methodology each time is eliminated. A human is felt to amass a repertoire of situation handling experiences to deal with a myriad of situations. (*See Section 2.1 for more discussion of thought processes and their refinement through experience.*) In monitoring a process, an operator would have a developed methodology to handle the situation where the process was observed to be behaving differently from the anticipated operation. The situation handling experience may begin to deal with the upset by evaluating the upset's potential severity. It could then search for the cause, analyze if the problem can be corrected, and determine plausible corrective actions. Situation handling experience plays the role of providing a memory of knowledge of how to comprehend and manipulate a domain.

Situation handling experience is closely coupled with situational experience. Situational experience consists of the knowledge of a situation. Situation handling experience is the knowledge of how to process a situation for a desired outcome. A situation handling experience appears to be mentally cued from a recognized situational experience. For example, if the stock washer is determined to be plugged, the situation handling experience of how to unplug it is recalled. The handling methodology may consist of attempting to shake the plug loose by opening and closing the slice. If this fails, it may try to use the washer showers to dilute the plug stock. Failing this, the alternative could be to shut down the washer and take a stick and wrench to it.

Some concepts of the four experience types have been studied by the field of psychology. It is well acknowledged that there are "numerous distinctions of various types of knowledge" (Regoczei & Hirst, 1989). Further, psychologists view knowledge as serving a functional requirement; "knowledge may be distinguished according to their functional use" (Regoczei & Hirst, 1989). This can be reiterated as, "how is knowledge used", not "what type of knowledge is it". The AI contingent has not adopted this view of knowledge yet, which has raised concern among some members.

"Unfortunately, AI currently seems to have few formal or even informal characterizations of knowledge types (Kidd, 1987). This is a serious gap, one that we hope will be filled in the near future" (Chandrasekaran, 1988).

The nature of experience has also been well explored. It has been expressed as,

"Experiental knowledge is associational, providing a direct relationship between situational data and conclusions, without the explicit use of the principles of the domain. The knowledge is usually related to a narrowly defined problem" (Prerau, 1989).

The employment of experience towards situations and recognition is familiar to the psychology field. Recognition is generally viewed in the context of pattern recognition and is considered as, "processes which are needed to recognize and categorize stimuli in the environment" (Cooke, 1989). Psychologists have noted recognition experience. "Experts in a particular domain have accumulated much experience with stimuli in that domain and as a result are skilled at recognizing very domain specific patterns" (Cooke, 1989). The concept that recognition has a predictive manner, or of experience providing a reference of what to expect in a domain, does not seem to be explicitly stated in the psychology literature. Overall, most of the concepts surrounding experience have been explored to some degree by the field of psychology.

Expertise is what all expert systems endeavor to capture and represent within their knowledge base. Most present forms of representing expertise in expert system are not very natural. For example, the representation of knowledge by rules has been criticized.

"psychologists--and even experts!--often find this somewhat simplistic, if not insulting. A veteran expert may be somewhat taken aback by the suggestions that a young 'knowledge engineer' is going to 'write down' in the form of rules the fruits of his hard-earned expertise gained over many years" (Regoczei & Hirst, 1989).

Recognition experience is an exception in that it can be well represented by production rules, and is often done so. However, the recognition knowledge is usually convoluted with the rest of the domain knowledge. It is not isolated and attached with the knowledge of the entity it is recognizing. None of the present knowledge representation forms, save case-based, preserve experience in an episodic form that can be directly applied. Case-base systems, though representing knowledge in episodic form, do not dissociate forms of experience. Expectational and recognition experience have not been incorporated as cases in any present case-base systems. In general, a case encompasses all knowledge of an episode. Thus, case-base systems see experience more like a well of experience memory that knowledge can be drawn from. When a problem is at hand, a case-base system searches through its memory to find cases that are similar in circumstance to the presented problem. Relevant information is then extracted from the best cases. A *case reasoner* collates this information to develop a solution (Ketler, 1993; Kolodner, 1989; Bonissone & Ayub, 1992). If the extracted case matches the problem exactly, only then is the case knowledge directly applied. Case-base systems generally are aimed at using precedent knowledge to develop a modified solution that meets the constraint variations of a similar problem. This occurs in such problem domains as decorative design, legal systems and cuisine selection (Bonissone & Ayub, 1992; Chi et al., 1993; Kolodner, 1989; Maher & Zhang, 1991; Vossos et al., 1991), and is not as prevalent in process monitoring. Case-base is a viable and powerful reasoning method, but should be included as a *thinking mechanism* of an architecture. In summary, present AI systems do not appear to use a natural representation or direct application of human experience knowledge.

The four types of experience knowledge and their manner of employment provide humans of employment and the representation of the experience types are implementable in a computable expert system. Individually representing these experience knowledge types in an expert system replicates the natural manifestation and potency of experience knowledge in humans. It is easier, therefore, for humans to express their knowledge, as they can recount if in its natural form. The representation of recognition experience in an expert system can also bestow a predicting agent to its intelligence. Experience offers an expert system an information rich knowledge that can be directly applied.

2.3 Expansion of Object Oriented Concepts and Its Application

The object oriented paradigm is extremely potent in representing the real world. However, its roots are not from artificial intelligence soil. The parallels that exist in the psychology field are not acknowledged, nor have they transgressed into the artificial intelligence realm. As a result, the artificial intelligence theory and applications of the object oriented paradigm have taken on a programming perspective. It is felt that the concepts of the object oriented paradigm embodied by this perspective limits the scope of representation to data base objects. For an expert system, there is a very different world that needs to be represented, that being the "world of conceptualization" that exists in a human's mind. The nature of this world is irregular and dynamic, and demands additional concepts to be recognized. Several new concepts are presented in this section. The intent is that these new concepts be assimilated into the present paradigm and employed by expert systems to provide a facility to fully represent a world of conceptualization.

Although it will not be discussed in-depth, it will be noted here that any application of the object oriented concepts must be done so properly. Incorrectly or partially applying the principles results in an impotent representation. In a review of present literature, some misapplications of the object oriented concept were noted. One such misapplication that was observed is that there is a reticence to release the notion of modularity and an insistence to develop modules out of objects (Betlem & Aggele, 1994). Also noted was the attempt to employ the principles of inheritance to represent system structures (Courdier & Herin-Aime, 1989). Inheritance is a representation of taxonomy, a hierarchy of trait commonality between entities. A system on the other hand, is a ***composition*** of parts. Another improper practice that was frequently observed was the use of the object's encapsulation feature merely as a framework to construct complex data types (Bandreddi et al., 1994). For the case study of this thesis, every attempt was made to correctly follow the object oriented principles.

The following sections present proposed enhancements to the object oriented paradigm.

2.3.1 Expansion of Object Features

A function of an "object" is to encapsulate *all* information about an entity. Typically, this has been done by expressing characteristics and behaviours of the entity as variables and formulas. However, real world entities perceived by humans are dynamic and varying. Further, the nature of the perceived world is highly nonlinear and relative, both in space and in existence. More often than not, mathematical modelling of behaviours is inadequate or insufficient. Intelligence is needed to understand the context from which the entity is being addressed when relating the entity's nature. For example, the personal value of two-ton truck may be deemed *bad* by someone who is required to take a long trip. However, the same truck may be deemed *good* if the person wishes to move furniture. In a process monitoring application, viewing a chemical addition flow as an object, what would constitute a *bad* condition would depend on the context of the operation. Different grades (products) require different consumption of chemicals, as do varying operation speeds, temperatures, chemical mixes, and stock consistency. A zero flow rate obviously is not *bad* if the process is off or if that chemical is not being selected. The proposed concept is that ***intelligence should be embodied*** in an object. An object should intelligently determine its characteristics and behaviours in relation to the conditions of the domain and context of the situation (Erlenbach, 1994b).

There is a "twist" that presents itself by having objects determine their being. In a human, the reasoning about an object is done separate from the object. If a human wondered if a chair was broken, he would consider the chair and draw a conclusion. The object (the chair) is passive, lifeless. The proposition being presented here is equivalent to the human *asking* the chair if it was broken. The chair would determine itself. An object under this postulation is active--*it becomes alive*. Henke (1994), when relating the work done to develop an expert system to schedule Space Shuttle missions, mentioned this observation in passing. "These objects are not passive data, but individual intelligent entities that can be requested to perform actions on themselves or each other." The realization that objects come *alive* leads to several interesting topics and technical possibilities. These topics will not be discussed here.

Although the idea of encapsulating intelligence within an object has been comprehended by gurus of object orientation (Entisminger, 1990), there have not been many artificial intelligence implementations of it. The one exception noted is the work Henke (1994) reported. The knowledge of the various planning activities that compose the development of a Space Shuttle mission schedule are encapsulated in objects. These objects use the object oriented messaging features to execute the intelligence and the steps necessary to acheive a mission plan.

Several benefits stem from incorporating intelligence to determine behaviours and characteristics into an object. The first advantage is that all knowledge about an object can be localized. This makes the activities of modifying knowledge of an object or adding new objects much easier. The second advantage is that it satisfies a criterion for generic thinking. With all the intelligent activities about the object being contained inside the object, the reasoner (thought process) does not need to know which object it is dealing with to determine the object's characteristics. It only needs to know what *type* of object it is dealing with and requests the information generically. A third advantage is that the inconsistencies of the world can be dealt with. The intelligence within the objects adapts the nature of the object to changing conditions of the world.

2.3.2 Expansion of the Object Oriented Concept

The object oriented paradigm is an organizational scheme to represent a "world" in an equivalent manner to how the human consciously perceives and organizes it. This world is the "world of conceptualization" that exists in the human's mind. The present concepts that compose the paradigm do not fully represent all the important aspects of this world. Two additional natures of a world were observed that should be accounted for. These natures will be considered individually in the following paragraphs.

The first of the observed natures is that entities in a world effect other entities. The effect occurs through a *relationship* between the entities. A relationship can be physical, such as a boy *pushing* a cart which effects the behaviour of the cart, or it can be abstract such as Paul *loves* Jane which effects Jane. The nature of the world that abounds humans is highly structured through innumerable relationships. A chemical process, for instance, is a system of physical relationships. (System representation is the topic of discussion in Section 2.4.) A representation of a world should embody the knowledge of the perceived relationships that influence the behaviours of entities within that world. This aspect of a world can be incorporated in the object oriented paradigm through the concept of encapsulating relationship information in an object. Relationship information consists of *which* objects there is a relationship to, *what* is the relationship, and *what are the effects* of the relationship. Most relationships have reciprocating effects, thus the information should include both how *it effects* the other object and how *it is effected.* Relationship knowledge should be accompanied by intelligence to account for the nature of the world, for similar reasons as those explained in Section 2.3.1. Thus, one should be able to inquire from an object what its present relationship is to other entities (Erlenbach, 1994b).

The other observed nature is that humans create mental links between entities that are regarded by the human to have a commonality. The linked entities often are dissimilar and weakly related, and the link can be to a physical or an abstract entity This mental link is referred to in this thesis as an *association.* An association is deemed to be a thought routing mechanism that is thoroughly employed by humans. (This notion is discussed in Section 2.5.) The commonalty may be a residual of some experience that involved the associated entities. For example, an arena may be associated with the local hockey team. The object oriented paradigm can be expanded to represent association of entities by assimilating the concept of embodying association information into an object. Association information should include *which* objects it associates with, and the *context* in which the association is established. Note, that a craft fair may also be associated with the arena during a summer month. To determine the context and the applicability of the association, intelligence should reside with the association knowledge as discussed in Section 2.3.1. One should be able to question an object of its associations to other entities which presently exist (Erlenbach, 1994b).

The notion of relationships between objects is established in object oriented design theory. The relationships commonly referred to tend to be from taxonomy structures and the principles of inheritance are applied to represent them (Martin & Odell, 1992). Often system structures are misconceived as taxonomies resulting in inheritance principles being misapplied. Design theory also recognizes system structure representation and the common representation is to include link attributes in each system object (Terpstra et al., 1992; Brunet et al., 1993). This form of representation offers no knowledge of the interactions within the system. Techniques using object oriented principles to express the interactions have revolved around specifying objects that contain possible relationship knowledge. These objects are attributed to the various system objects (Arzen, 1992; Dimitrov, 1992). Associations are overlooked for the more concrete concept of relations. If associations are recognized they are considered and represented as relationships.

In summary, relationships and associations are object oriented modeling concepts. These concepts suggest that relationship and association knowledge be encompassed in an object as well as the characteristic and behaviour knowledge. The knowledge of relationships provides a representation of the interactions within a world. Association knowledge provides a representation of thought routes developed by a human. Both of these concepts play a major role in expanding the knowledge that can be represented by the object oriented paradigm.

2.3.3 Expansion of Object Oriented Application

The object oriented paradigm (OOP) views objects as the fundamental building blocks that compose a world. Every entity of a world is represented as an object and all knowledge about that entity is contained within the object. Usually in an application, the world being represented is the one which is external to the human. This world is typically represented by physical entities. In an expert system, it is the world which exists in a human's mind that is being represented. This world encompasses both the perceptions of the external world as well as other perceived realities inherent to a human. These other mental realities include *thought processes*, *experiences* and *abstract* concepts. Each of these were deemed to exist in the human mind as concrete entities that should be represented as objects. *Thought processes* were presented in Section 2.1 and experiences were discussed in Section 2.2 (Erlenbach, 1994b).

All the information about these entities is contained within the object. For the *thought process* object, the behaviour of the object is the intelligent *thought process*. Embedding intelligence into an object was suggested in Section 2.3.1. Thus, the *thought process* knowledge is intelligently processed inside the object. Cueing of another *thought process* is performed by the messaging functionality of the object oriented paradigm. The characteristics of a *thought process* include the purpose, the scope of applicability, and the inherent goals. Experience objects encapsulate the knowledge of an experience. The information about an experience is represented by the characteristics. An experience does not have a *behaviour* in general, but imbedded in the object are the prediction and identification experience behaviour. Abstract objects are similar in structure to physical objects. They have intrinsic characteristics, behaviours, relationships and associations that are encased by the object.

On one hand, the development of thought processes, experience, and abstract entities as objects could be dismissed as a design preference. However, there is a greater significance to this. What is being suggested is that these entities are consciously perceived by a human as concrete mental packages that include all the relevant information about them. The idea of encapsulating experience as an object was not observed in any of the papers on case-based systems. Even the notion of employing object oriented concepts to represent knowledge does not appear to have filtered into case-base applications yet. Many of the expert system applications incorporating the object oriented paradigm apply it only to represent data or physical entities (Betlem & Aggele. 1994; Clements & Preto, 1993; Terpstra et al., 1992; Vossos et al., 1991). Abstract concepts have been represented as objects in some applications (Bandreddi et al., 1994). *Thought processes* are a concept being suggested in this thesis. Procedurized applications of rule sets are the closest representation to *thought processes*. Most all applications using this concept form the rule sets into modules which act on some body of facts or data.

There are several values in objectizing these three manifested realities. First is that knowledge is localized and one gains the maintenance benefits that were discussed in Section 2.3.1. Another benefit is that all the entities of a world are represented using the object oriented paradigm to form one completely integrated body of knowledge. The power and efficiency from the object oriented features, such as summoning information by messaging, can be gained because of the full integration of all the knowledge forms. The structure of the paradigm is similar to human perceptions thus the transfer of thinking and experience knowledge is more direct. This facilitates the extendibility and modifiability of this knowledge. A greater complexity of knowledge can be realized.

2.4 Representation of Systems

A system is defined as "a set of connected things or parts that form a whole or work together" (Oxford American Dictionary). This can describe almost everything that is perceived. Physical entities are regarded as being made up of systems of molecules; cars, buildings, equipment, devices, and so on are all systems of components. There are ecological systems, weather systems, process systems and other abstract systems. To understand a world, it is felt that one must comprehend the systems that compose it. One must understand both the nature of the various systems, as well as the interactions within these systems, as they determine the manifested behaviours of a world. To incorporate an understanding of a world in expert systems, it is imperative that knowledge about systems be represented. The representation should comprise a macroscopic knowledge of a system, its purpose, characteristics, behaviours, etc. It also needs to embody a *deep* knowledge of the componentry and compositions of the system. A method to effectuate such a representation is suggested and discussed in this section.

The underlying concept of the proposed representation method is to use objects and their relationship knowledge to form the representation of a system. A system is composed of parts, or components, and a ***structure*** of how these components fit together to form the whole. The structure's connectivity can be expressed as a network topology. Each node of the network is a component of the system. The proposed concept is that node be an *object* that represents the system component. Encompassed within the node object is the specification of which other system components (nodes) that it effects. The specification defines links between nodes that represent the connectivity of the system's components. Thus, the nodes and links form a network topology that illustrates the system structure. The encapsulated knowledge of the characteristics and behaviours of the object describes the nature of each component of the system. The relationship knowledge embodied within an object (discussed in Section 2.3) describes how that component effects, and is effected by, the other system components. Thus, the knowledge of the components and the knowledge of the system interactions between components are represented. A system is completely represented. Further, the relationship and behaviour knowledge is intelligently determined. The knowledge about the system is available according to the context of the inquiry. Information can be drawn from the system representation through a variety of means, such as direct inquiry, searches, and simulations. Figure 2.4 depicts an example

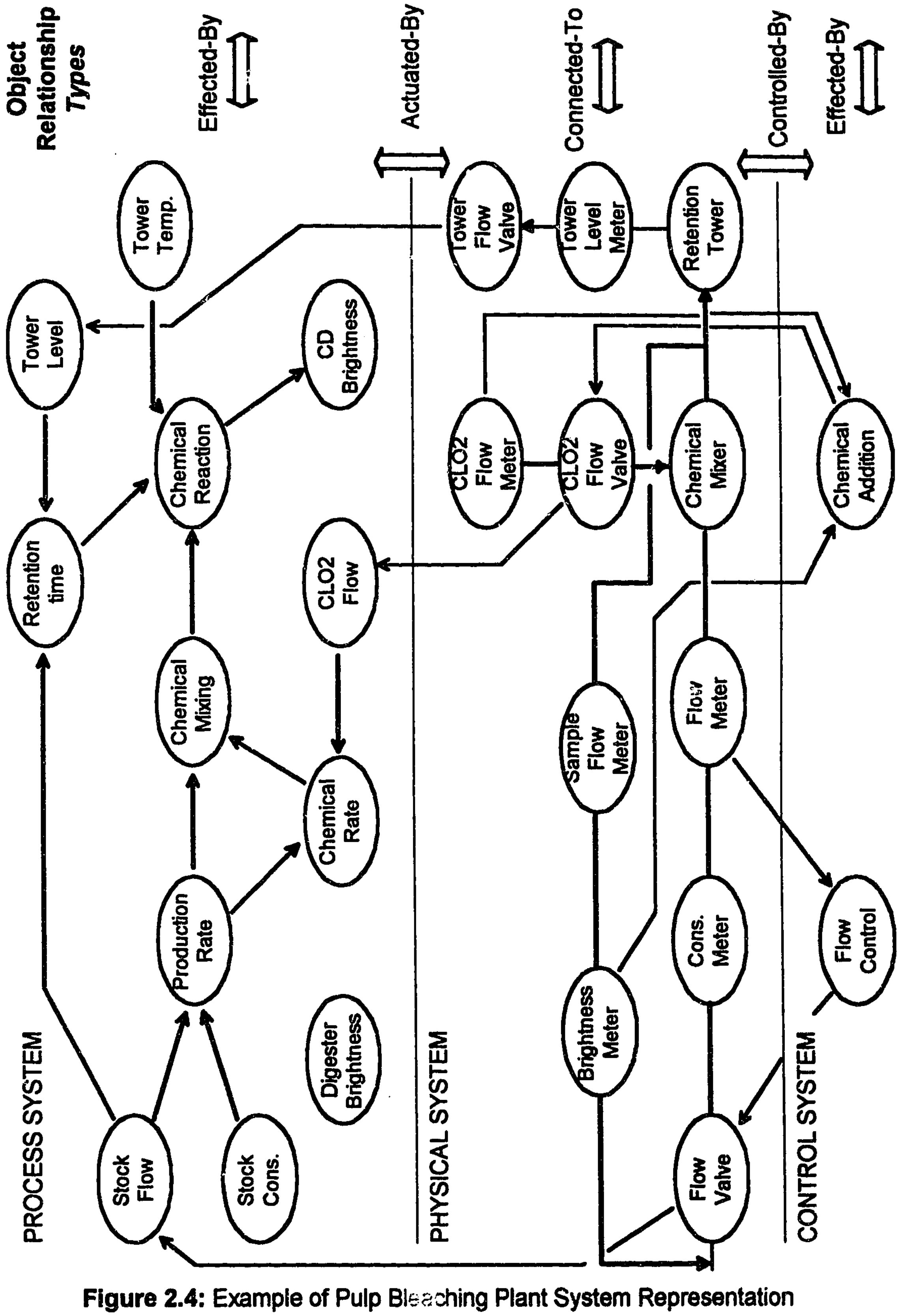


Figure 2.4: Example of Pulp Bleaching Plant System Representation

system multi-system representation of a Pulp Bleach plant. This example also illustrates interrelationship knowledge between systems, shown as relationship types.

In following true object oriented form, the system as an overall unit is represented as an object. The characteristic and behaviour knowledge contained by this object describes the nature of the system as a unit. The knowledge includes the purpose and description of the system, general behaviours of the system, and other information about the system. A heuristic model of the system's behaviour in this object is general and does not employ knowledge from the component level of a system. However, a behaviour may be to analyze itself, in which case the object could evaluate the system at the component level. The system object embodies the whole system through the *composition* knowledge. The composition knowledge comprises information of which objects the system is composed of and how the composition manifests, (such as the input and output objects of the system). The system object provides knowledge to the thought process objects about systems that make up the *known* world.

Note: Language compilers like C++ offer no assistance to coding the concept of composition. A worthy adaptation to a compiler may be to recognize a composition object and to update the object with information records for each object that is defined to be in the composition.

Many systems can be represented. An object (system component) may be a component of several other systems. For example, the stock flow in a pulp process is part of the process, the material balance, and the production flow systems. A set of relationship links and behaviours are defined within the object for each system that is part of. In addition, a component of a system may itself be described by a subsystem. The node object representing this system component is a *system object* encompassing the subsystem. For instance, the components of a material balance system may be defined as the various stages of a bleach pulp process such as the digester, the chlorine, the elemental oxygen, and the chlorine dioxide stages. However, each of these stages is a material balance system in itself. A subsystem provides more low level and detailed knowledge about the world. A thought process object can utilize these levels of knowledge to efficiently analyze a system by evaluating the higher level representation initially and focusing in on the lower level when required. In general, the knowledge of a world can be represented by this manner to any depth, granularity, or complexity as desired for the application.

There has been much research in the artificial intelligence field towards the representation of systems. The approach taken to develop the proposed concept of this thesis is quite similar to the work that has been done. The differentiating factor is the use of objects for the nodes and the incorporation of a system's interrelationships in the node object. The knowledge of the system then resides in the structure representation and not in external rule sets. The next paragraphs discuss the work done by the AI field.

The notion of representing systems by using network topology is well known.

> "One of the simplest types of knowledge that one can use in a diagnosis system is structural or connectivity information" (Milne, 1987).

Networks typical are employed to demonstrate the connectivity structure of a physical or logical system. In some cases, however, relevant information of a system is contained in the nodes of a network. This information is processed by an intelligent reasoner (Gang et al., 1988). The intelligent reasoner, called the structural reasoner, is viewed as an external body of knowledge that references the network's embodied information to draw inferences. The concept has been summarized as follows, "The knowledge encoded in a network, being declarative, is somewhat like that stored in a book: it is available for the support of intellectual activity only if there exists some outside agent that can retrieve the knowledge and apply it. This outside agent embodies knowledge about how to manipulate the information in the net" (Hendrix, 1979).

The information that can be contained in the nodes is presently seen to be attribute values and data. Further, since the structural reasoning knowledge must be applied generically to all nodes of a network, individual attributes of a node are not accommodated. A common usage of non semantic networks is in the application of fault detection (Gang, R. et al., 1988; Milne, 1987; Shiozaki et al., 1985).

Various approaches have been taken to represent the knowledge of the interrelationship of a system. One approach taken is to model the system's overall behaviours, thus capturing knowledge of the interactions (Pople et al., 1994; Alaman et al., 1992; Kramer, 1987). These models can be analytical, heuristic, or both. Models, however, encompass everything as a unit and internal system knowledge can not be accessed. Another approach is to represent the system in the form of a hierarchical structure. Each step within the hierarchy is a type of relationship that exist between the entities (Lim and Cherkassky, 1992; Cercone and McCalla, 1983; Hendrix, 1979). The true role that hierarchical structures plays is to depict taxonomies (Hendrix, 1979), though, where each branch represents a *commonality* between the entities. A system, on the other hand, is a composition of entities that *interact*. Although it has been tried, hierarchies do not fit well to represent system interrelationships. A third approach is in conjunction with a network representation. A relationship between entities of a system is referred to as a causal relation (Milne, 1987; Sembugamoorthy & Chandrasekaran, 1986; Patil et al., 1984). A network topology that represents the causal paths of a system is referred to as a causal network (David and Krivine, 1987). A causal network attempts to represent a system.

However, the approach taken is to integrate the causal relationship knowledge with the structural reasoning knowledge, which is external to the network. Because of this, the knowledge must be generically applicable to all the network nodes. Thus, the represented causal relationship knowledge is often general to the system. For specific domains and applications, such as fault diagnosis, this is acceptable. Complex systems with diverse interactions and behaviours, such as chemical processes, can not be represented in this manner. The key to representing a system of such nature is incorporate the specific causal knowledge at each node. The generic structural reasoner can retrieve the particular interaction behaviour knowledge from the nodes.

There is further advantage of representing a system through incorporating the knowledge of it into the structure representation. This allows both the ability to evaluate a system by its components as well as to perform ***mental simulation*** of the svstem knowledge. By declaring a context that the system should be viewed to be existing in and by introducing a hypothesis, the outcome of the system can be simulated. The simulation can be either heuristic or analytical determined, or both. Defining a context allows the component objects to pre determine their relationship behaviours, that apply to that context. The context, for instance, may describe a potential future condition. The hypothesis defines the various input conditions for the simulation. The structure of the system guides the simulation path.

There are some interesting implications that result from simulation using the analytical representations of the object traits. First is that the object traits, such as its behaviours, are expressed by mathematical formulae that are adapted by an intelligence layer to apply to the declared context. This amounts to intelligently adaptive models within the expert system's knowledge. Further, for an abstract system such as a pulp bleaching process, the analytical simulation ***is a simulator*** with the ability to be run predictively. Each component of the system can be an abstract component of the pulp bleaching process. The same flexible granularity at which the system is represented forms the granularity of the simulator. A comparison of measured sensor values with the simulated values could form another method of problem detection. Another implication is that the analytical expressions could be *models, estimators* or *neural networks*. It is conceivable that the same intelligence that allows humans to decide when to use an estimator, when to apply new parameters, when to train a neural network, could be embedded into the object. Thus the simulation ability could be adaptive, self pararmeterizing, and self learning. This knowledge could also form a platform for other activities, such as a supervisory level over advance control algorithms. It is worth noting that the ability to analytically simulate takes an expert system beyond the abilities of most human brains. Although no followed up work on the concepts mentioned in this paragraph was done for this thesis, these form an interesting realm of futher study.

The proposed method to represent systems presents a number of benefits. Some of them have already been discussed. One could summarize by saying that all the benefits of representing the world as objects apply to the representation of systems. The internal intelligence suggested for objects, provides a means of dealing with nonmonotonicity of a system as well. The representation is easily extensible, both in levels of composition and in expansion of system components. The component knowledge is localized, providing an ease on implementing the knowledge and maintaining it. The knowledge encompassed by the system representation can be used to evaluate the world by decomposition or can be used to mentally simulate possibilities. The method can, (and is highly recommend that it is), represent abstract system knowledge for cognitive reasoning. Perhaps the cleanest feature is that the object oriented representation embodies the system knowledge within a single homogenous body of knowledge.

2.5 Associative Memory of Experience

The term *associative memory*, in the context of this thesis, refers to that aspect of the human mind which recalls memory through the association it has with the present perceptions. An *association* is being defined as a mental link between of two or more *nonsystematic* but somehow related concepts. (Association has been introduced in Section 2.3.2.) The concepts being associated can be very dissimilar. A relationship, in contrast, is defined as the linking of two or more *correlated* concepts. Associative memory is a common trait of the human mind. To give an example consider the following object: a vase. Some people may draw the association of a *rose* to the thought of a *vase*; other people might think of an antiquity such as the *Ming dynasty*. The rose and the vase are very different concepts, yet our mind links them together. Associative memory is felt to be a definite and separable mechanism of memory recall through developing associative links. Further, it is felt that associative memory is the prevalent method by which experience knowledge is recalled. The concept of associative linking and its use to recall memory is deemed to be a powerful mechanism of the human mind that should be replicated and employed in an expert system.

An association is essentially a *hyper-linking* of concepts within the memory. A hyper-link is a direct and non contiguous connection that can link two remote objects, or in this case remote thoughts. A hyper-link forms a quick and efficient means to traverse mental thoughts. Associations appear to be developed between concepts that are commonly perceived during a time frame. There are many different ways that concepts may be brought to the awareness of humans. For instance the linked concepts may have emanated during an event or an experience, or they could have emerged through the course of a conversation. There are many diverse reasons why an association is established, ranging from commonality of purpose to esoteric, vague connections due to happenstance.

Associations often are established between various concepts that surround an experience because of their pertinence to the experienced event. For example, a methodology used to unstick a valve could be associated with the situation where a check valve at a chlorine unloading tank was stuck closed. This situation itself would probably be associated with a detected chlorine flow problem by the operator to whom the episode occurred. It is felt that over time humans develop an extensive and complex network of associations (hyper-links) that span the breadth of the concepts within their memory. This network is the key to the access of a wealth of diverse knowledge. The network itself is a knowledge form representing how a human perceives and has experienced the world. The method to represent association links in an expert system is discussed in Section 2.3.2.

Associative memory is a memory recall mechanism that retrieves information through the associative links within the memory. Each associative link embodies the context in which the link was established. The recall mechanism appears to use this information in selecting whether an associated concept should be retrieved or not. The possibility for an associative recall to occur exists if a concept which has associative links to other concepts is presently being perceived. Whether or not the associative memory activates the recall appears to depend on the similarity of the link context with the present perceptual context. At some degree of similarity, which seems to vary drastically with conditions and between people, the association recall triggers. For example, take the situation where there is a chlorine flow problem in a pulp bleaching process. The situation where the check valve at the chlorine unloading tank is stuck closed may be associated with the chlorine flow. The situation where there is a chlorine leak resulting in excess chlorine flow may also be associated with a chlorine flow problem. In addition, a situation handling experience of how to increase chlorine flow may be associated with the chlorine flow problem. Consider the perceptual context where there is concern because the pulp brightness is low due to a lack of chlorine. The associative recall mechanism bypasses the situation handling experience because the present context is to seek experience that could explain the present situation. Further, the chlorine leak situation is ignored because the context of that link is that the chlorine flow is excessive. The experience where the chlorine flow is checked is recalled given this situation. This activity of associative recall mechanism can easily be implemented in an expert system.

Associative memory is deemed to play a predominate role in the retrieval of experience knowledge. As an example of associated experience, consider a certain piece of music. When this music is played, a human often reminisces about some wonderful moment the music brings back. This moment is an experience the human had. The experience was associated with salient concepts that were perceived during the same time that the experience existed, for instance the music. The perception of these concepts at a later time recalls the associated experience to the awareness of the human. This is felt to be true in general of all significant experiences. A significant experience is one that is perceived to be having an impact on the human's world. Association with perceived concepts is a means of recalling knowledge and information about an experience. For a process monitoring problem solving point of view, a *situational experience* could be associated with perceived operating *conditions*. For instance, take the stuck chlorine flow check valve situation discussed before. When a low chlorine flow *condition* is again detected, the stuck check valve *situation* may be recalled by the operator. Further, if the situation is verified to be occurring again, the operator may then remember other situations that were associated with this situation, such as perhaps the chlorine mixer tripping off and stock plugging the chlorine inlet of the mixer. By association also, he may recall how each of remembered situations *were handled*. In terms of representation of knowledge, this is describing a complex chain of associative linking between experiences of different types. Situational experience may be associated with other situational experiences, situation handling experiences, or various entities in the world such as process concepts (process objects). Situation handling experiences have a similar nature. This concept of chaining memory recall could be explored in far greater depth than that proposed here for the purpose of process monitoring, but will not be done for this thesis. The chain itself is as valuable a knowledge as that of the experience knowledge and should be implemented in the expert system. This is achieved both by defining the associations and by developing the methodologies in *thought processes* to examine the full extent of the associative memory.

Associative memory could provide some other functions for an expert system that will be mentioned here, but not elaborated on in this thesis. It is felt that associative linking could provide a means of incorporating learning into an expert system. What is being suggested is that if an expert system is able to ascertain the information that defines an experience *type* from the occurring experience, then new learned experience objects could be generated and assimilated into the expert system by associative links. A notion of "chunking" information into an expert system has been proposed by Laird, Rosenbloom, and Newell (1984). This is a similar concept but associative learning offers a means to realizing more general learning. The work done by Laird and group was focused mainly on chunking task knowledge. Several ideas do exist of how an expert system can ascertain relevant information. The concept of association being a mechanism that is used to *learn* has been examined by the field of psychology (Atkinson et al., 1983). Associative memory could also provide a *second level* thinking method. For example if the first level thinking about a bleach brightness problem fails to reach a conclusion, then *any* associations with the concepts that define the problem could be sought out to try to find a new way to approach the situation. For instance, it may be felt that chlorine is a salient factor in the brightness quality and an examination of any associated concepts to the chlorine could be done. This may lead to the thought of home bleaching which leads to a situational experience where the bleach reacted with some element in the water and left the clothes gray. From this thinking pattern it may become apparent to check the mill water quality. If this notion is taken to the extreme, then one approaches the idea of inductive thinking where bizarre and abstract concepts could be brought into play through weak associative links. Sometimes this is a needed action to overcome the perceptual walls in the human mind.

The concept of mental associations is recognized by psychologist. It is used as a probe into the psyche of a human. The probing is performed by flashing partial illustrations at the patient and having them respond with the first thing that comes into their mind. The idea is that the vagueness of the illustrations will allow the patient to associate with it the salient frustrations that are currently manifesting within their mind. The role that associative memory plays in the human mind has been researched by a number of people. Work done by Fitts & Posner (1967), and Anderson (1983, 1987), (also discussed in Section 2.1) suggests that knowledge becomes associated in the second (*associative*) stage of human learning. Repeated application of knowledge results in "domain-specific" knowledge that has "*direct associations* between specific conditions and the resulting actions" (Gordon, 1989).

> "When conditions in the environment match the conditions of the procedural rule, the action is automatically invoked, circumventing the longer and more tedious process of retrieving declarative knowledge (Charness & Campbell, 1988)."

This is further supported by work that was performed by "Chase and Simon (1973)" on chess playing skill. They concluded that "chunks," a chunk of memory or knowledge, "are associated with particular moves" in chess playing (Cooke, 1989). The work became labeled as the recognition-association theory. The association mechanism is conceived to be a direct link to knowledge that circumvents the cognitive reasoning process.

"People becoming competent in a given domain move away from the use of symbolic or declarative knowledge and toward a reliance on perceptual, nonverbalizable procedural knowledge. This can be viewed as an *associative memory phenomenon*, in that the stimulus components triggering the use of declarative knowledge come to be directly associated with system output or actions" (Gordon, 1989).

It was necessary to define the terms *association* and *associative memory* for this thesis because present studies in artificial intelligence on memory view memory that is searched for *based on similarity* as associative (Kohonen, 1989). This notion dates back as far as Aristotle in his book entitled "On Memory and Reminiscence" (Sorabji, 1972.) The notion put forth in this thesis is that there exists associative links in memory that guide memory recall. The similarity of context is only used to select which links to activate. Further, the Artificial intelligence studies do not make a distinction of memory that is recalled, which is not directly correlated to the present context of thought (associated). In addition, there is little distinction made between association and relationships when considering a representation of the world (Kohonen, 1989). Hence, the artificial intelligence theory of associative memory does not incorporate the potent functionality of unstructured associative thought. Present designs of case-based reasoning systems originate from the associative memory perspective established within the artificial intelligence contingent. The cases, which are experience, are viewed as part of a contiguous memory. Case recall is done by searching through memory. Techniques are used to make memory recall more effective, such as indexing (Ketler, 1993), categorizing based on salient features (Bonissone & Ayub, 1992), or if possible structuring the memory based on known relationship (Ishida, 1989). In contrast, the perspective of associative memory proposed in this thesis in conjunction with the object oriented representation of knowledge, develops a hyper-link structure with direct links to cases.

Association of memory and associative links provide a fast and efficient manner in which concepts are organized and retrieved in memory. It allows complex cause and effect, and other patterns to be expressed in memory. It appears to offer a foundation for inductive and higher level thinking. Further, it is a means by which learning can be consummated. The disadvantage is that associative memory can be relied on too heavy, "always taking the first thing that pops into the mind" (anonymous). Associative memory seems to be the predominate way that humans access experience and is a method that can, and should be, replicated in an expert system.

2.6 New Knowledge Representation Architecture

The previous five sections have presented various concepts towards the representation of knowledge in an expert system. The presented concepts comprise both ideologies for how knowledge can be represented and a framework in which the representation can be realized for an expert system. These concepts merge together to form an overall knowledge representation architecture. The foundation of this architecture is felt to be generally extendible for most expert system applications. However, the presented architecture content is more indicative of the requirements identified for the purpose of representing process monitoring. *The prima , intended purpose for this architecture is to represent the activities and the world that governs process monitoring.* This section brings together the individual concepts, and presents the proposed new architecture. Figure 2.5 illustrates the overall architecture.

The architecture can be viewed as a single homogeneous body of knowledge. This body of knowledge is completely composed of conceptual objects that represent all entities and

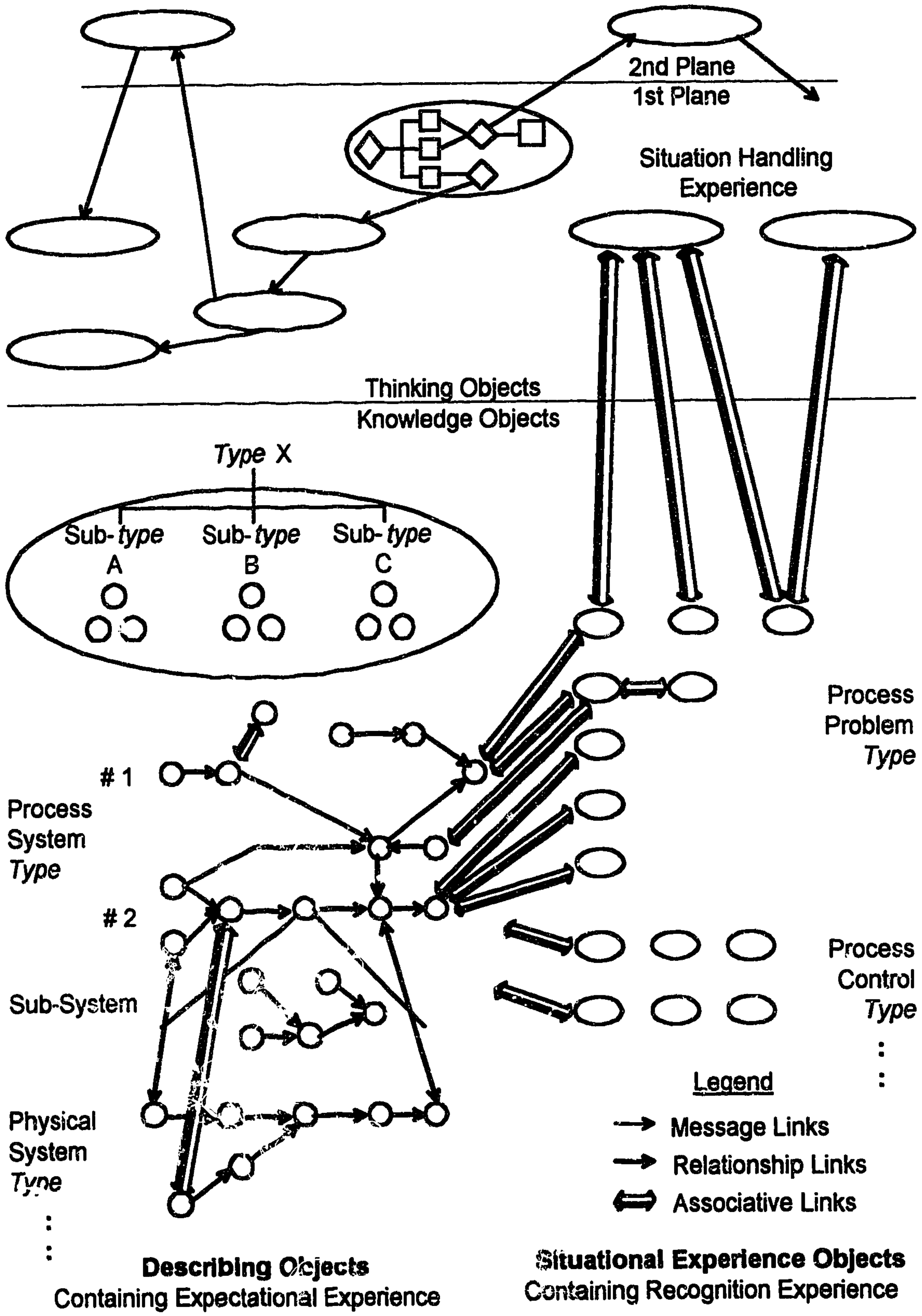


Figure 2.5: Proposed Knowledge Representation Architecture

manifestations of a known world. The objects can be split into two groups, *thinking objects* and *knowledge objects*. The *thinking objects* represent the knowledge of how to reason or think about the world. The object has an intrinsic reasoning purpose and several embedded goals that it can engage to direct its decisions. The description of its purpose and category are contained within the object as knowledge that can be accessed by other objects. *Thinking objects* encapsulate a *thought process* that is in the form of a methodology that performs *thinking tasks*. The *thought process* also embodies knowledge on how to evaluate the success at achieving a goal and make decisions of the next *thinking task* to employ. All the *thinking tasks* are imbued with knowledge pertinent to the *thinking activities* it performs. *Thinking task* can also issue external commands and acquire information from the knowledge objects. All the components of a *thought process* incorporate the knowledge of what information is available to be requested through knowing the interfaces of the *types* of object it is dealing with. Thus information can be generically access from the *knowledge objects*. The *thought process* knowledge can be either captured experience methodology (*situation handling*) or a general procedural methodology. All execution of *thinking activities* is performed using a variety of *thinking mechanisms* that are inherent to a system.

Thought processes can execute at either the first or second plane of reasoning. On the first plane of reasoning thought is processed to determine solutions to problems. The second plane of reasoning employs *thinking tasks* to evaluate *how* to reason a problem. The execution of *thought processes* towards achieving a reasoning objective often unfolds in a hierarchical form due to the initiation of sub *thought processes*. Further, this execution can jump between first and second plane *thought processes* during the overall reasoning process. A *thought process* object can associate other relevant *thought process* objects and can be associated with *situational experience* objects. The associated *thought process* objects could be alternative methods known that have been applied to the same situation. The *situational experience* objects may be situations that could manifest during the execution of the *thought process* depending on the outcome of various decisions.

The *knowledge objects* that are comprised in the body of process monitoring knowledge can be roughly categorized as *describing* and *experience* objects. *Describing* objects can be further grouped into *physical* and *abstract* objects. *Physical* objects represent tangible entities of the world; *abstract* objects represent abstract concepts existing in the world. How the world is described can be organized in several different ways. The objects of a world can be *classified* according to a commonality taxonomy using the *inheritance* principle. The objects of the world can also be *categorized* as *types*. In additions the world can be organized into *layers of systems* with the *composition* principle and *relationship* knowledge. Through these concepts and by the selection of the objects, any granularity of the world can be represented.

Describing objects encapsulate *characteristic*, *behaviour*, *relationship*, and *association* knowledge that replicate the object's nature. Knowledge of *characteristics* includes a description of the object and its *type*, system related and associated object names, and other information about the object. The *behaviour* and *relationship* knowledge describe how the object behaves and relates with other objects. The *association* knowledge describes the context for which associations can be drawn for linked concepts. *Describing* objects can have associations to other *describing* objects and *situational experience* objects. Intelligence knowledge of how the object's nature varies in accordance with the context of the world is integrated into the object. The knowledge consists of intelligent methodologies that replicate the dynamic nature of an entity in a world. The intelligent methodologies are not human thinking knowledge as in the *thought processes*. The knowledge is executed by *thinking mechanisms* and acts to effect the *characteristics*, *behaviours*, *relationships* and *associations* of an object for a given context. An entities' *expectational* experience is also embedded in a describing object. This form of experience can be employed by the behaviour knowledge to determine the expected nature of an object.

Systems are represented by the *relationship* knowledge incorporated in an object. The overall system is represented by a *system object*. In addition to the standard object knowledge, information of the system composition, the system structure, and how the system as a unit behaves are encapsulated in an object. The composition information should include a list of the composite objects. The structure of a system can be graphically depicted by linking all the composite objects of a system together. Each system object retains the name of the [illegible] directly links to. A *system* object can be an object of a more macroscopi[illegible] ing layers of systems.

The *experience* objects in [illegible] architecture represent *situational* (episode) experiences. Exp[illegible] ily organized in the architecture through *associations* with ot[illegible] objects can also be categorized by *types* of situations to allow a [illegible] process to search similar *types* of experiences. *Situational* experiences are often associated with *things* of a world, the *describing* objects. They are also associated with *situation handling* experiences that are *thought processes* of how to deal with that *situation*. *Situational* experiences can be associated with other *situational* experiences as well; one experience may recall other similar nature experiences.

The *situational* experience object contains *characteristics*, *behaviour* and *association* knowledge about the nature of a situation. The *characteristic* knowledge includes a description of the situation, the *type* of experience and a myriad of information about the situation. *Recognition* experience is inherent in a *situational* experience object.

Recognition experience is employed by the object's *behaviours* to determine if either the situation is *occurring*, or is *predicted* to occur. The *behaviours* of an experience object may also explain the nature of how a situation behaves and how it will progress.

There are a few other miscellaneous concepts surrounding the architecture ideology. The initiation of a reasoning process generally originates as a result of an external *cue*. Process data are seen as global entities that can be accessed by objects. The object's *type* dictates the object's interface, thus providing the ability for the *thought processes* to reason on generic objects. Experience plays the role in the architecture of being both a directly applicable knowledge and a fine tuner of existing knowledge.

To help clarify the concepts of the architecture, an example of the reasoning activities of process monitoring will be discussed. A pulp bleaching process will be used for the example. The example is intended to demonstrate how the architecture can represent knowledge and execute reasoning of the process monitoring activities. The example does not describe specific knowledge nor the employed thinking mechanism that would be used to develop such an application. This could be done through appropriate selection from the many mechanisms that are currently available (some have been discussed in Chapter 1.) The discussed example is only one representation of many possible perspectives of process monitoring activities, any of which can be represented by this architecture. Figure 2.6 illustrates the example of the process monitoring thinking methodolgy.

A process operator generally scans the process operation on a frequent periodic basis. This is to say that after a certain time the operator feels compelled to turn his attention back to monitoring the process. This feeling is a mental cue that initiates a process *monitoring thought process*. The *monitoring thought process* functions to detect process

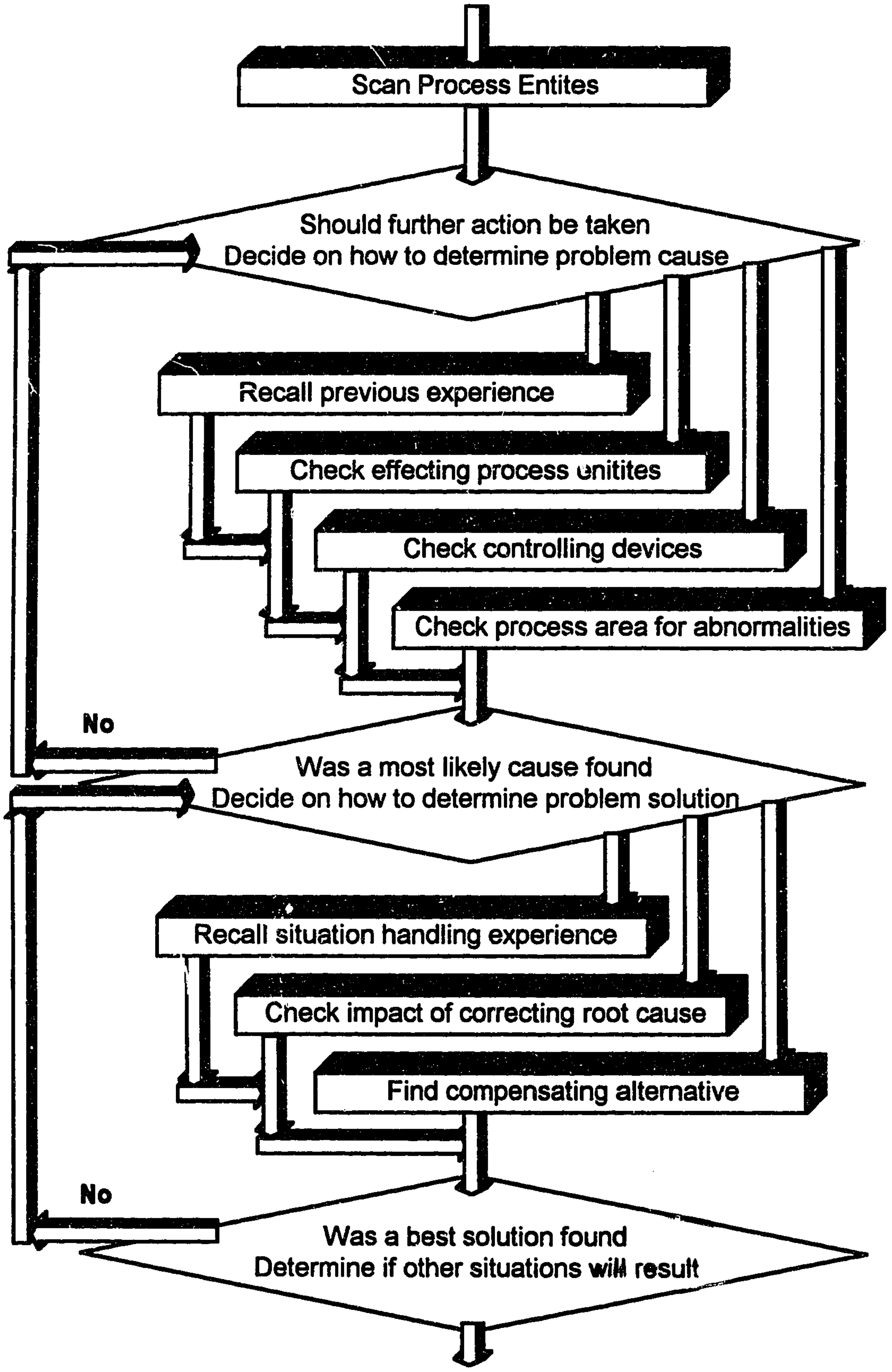

Scan Process Entites
Should further action be taken
Decide on how to determine problem cause
Recall previous experience
Check effecting process unitites
Check controlling devices
Check process area for abnormalities
No
Was a most likely cause found
Decide on how to determine problem solution
Recall situation handling experience
Check impact of correcting root cause
Find compensating alternative
No
Was a best solution found
Determine if other situations will result

Figure 2.6: Example of a Process Monitoring Thinking Methodology

problems; find the source, severity and consequences of the problem; and determine actions to take to correct or control the problem. A common way of detecting problems is to scan the measured values of process entities such as flows or consistencies. These abstract process concepts are represented as process objects. The *monitoring thought process* initiates another a sub thought process which has knowledge of how to scan the process. The *scan thought process* contains knowledge of key process points whose behaviour can indicate the state of the overall process operation. For instance, the operator may look at the KAPPA number of pulp out of the digester, the chlorine flow, the stock consistency, etc. An operator has in his memory an expected behaviour of each key process entity that has been established through experience. The *scan thought process* functions by inquiring in memory if the measured values of these key points are what is expected given the present process conditions. This would be done in the architecture by sending a message to the process object to determine its expected behaviour status. It then decides if further action should be taken at that time. For this example perhaps the stock consistency is at 4.6 kg/l^3 and it is expected to be no more than 3.9 kg/l^3 in which case there is suspected problem.

If a scanned process entity is indicating an undesirable behaviour, the operator through his *monitoring thought process* decides how best to uncover more information of why this behaviour is happening or what could happen due to it. Several choices for finding more information are available to him: recalling previous experience memory, tracing root of the problem through the process system knowledge, examining the behaviour of the physical devices that manipulate the suspect process entities, or examining other process points in the physical vicinity to see if there are other unexpected things happening. Typically the first activity that comes to mind is the memory recall. The *thinking task* for this consists of summoning all the experiences associated with the process entity that fit the context of the situation. In this example there may be experiences of the pulp washer plugging and of a faulty consistency sensor associated with the stock consistency process entity being higher than expected.

The next action taken by the *thinking task* is to determine if any of the recalled experiences explain the situation. For the plugged washer experience, the recognition indicators may be low re-circulation flow along with high headbox pressure and high consistency. Through the recognition experience knowledge, the situational experience can be declared to not be happening again, to be happening again, or that it is predicted to happen. A feeling of confidence of the similarity of the symptoms to the present situation emanates out of the recognition. Perhaps through experience it is recognized that there is a 70 percent likelihood that the washer could plug, given the present consistency, stock and re-circulation flows.

The *monitoring thought process* then must decide if the recalled experience most likely explains the present situation causing the unexpected process entities' condition. It also must decide if all the possible outcomes given the present conditions are accounted for. If the experience seems to fit, then the experience knowledge of the severity and consequences is assumed for the current situation. Certainly one consequence of a plugged washer may be that the production line will have to be shut down while the washer is unplugged.

For the plugged washer experience there may be *associated situation handling* experiences of how to deal with the situation. Typically in process monitoring the goals are the same for any occurrence of an undesirable process situation, that being to try to keep salable production happening and prevent any destruction of equipment or lives. The associated *situation handling* experience for a potential plugged washer may be to increase water content or open the washer slices more in a predictive upset situation.

Either of these actions may have other associated *situational* experiences that the *monitoring thought process* must address. For example, the opening the washer slice has resulted in uneven matting and or poor chemical extraction. The *situation handling* experiences to handle these new situations are associated with the *situational* experience knowledge. The *monitoring thought process* continues reasoning until all know conditions are met and acted on.

If there is not enough confidence that the cause of the present condition has been found, then the *monitoring thought process* may decide to try another approach to determine the problem root. A second approach may be to trace the process using the process *system* knowledge to find a root cause. The *thinking* task for this activity would examine the condition and the relationships of the suspect process entity and determine which other process entities could be effecting it. It would then check the expect behaviour of the potentially effecting process entities. Let the example be changed to the case where the pulp brightness entity is low. The brightness can be effected by the chlorine content, the retention time of the chemical addition, the residual lignin in the pulp, the content of chlorine dioxide, etc. Assume the process to be running as elemental chlorine free, in which case the brightness is not effected by the chlorine content. The other process entities that could possibly be effecting the brightness would be checked. From this perhaps only the chlorine dioxide content entity is behaving unexpectedly. The *monitoring thought process* would turn its attention to the chlorine dioxide content entity. It could make a decision to recall experiences to explain this process entities' situation or possible outcomes. Alternatively it could check the condition and the relationships of this process entity to determine which other process entities could be effecting it. In the latter manner the process operations would be searched through knowledge of the process system. In doing so several different system layers and different represented process systems could be accessed.

A search of such nature would continue through a process system until no effecting process entities can be found. It then would be highly suspected that the root cause emanates from the last process entities of the system search that shows unexpected behaviour. In this example it may be discovered that the chlorine dioxide flow is very low. The *monitor thought process* of the operator is now is faced with a reduced problem that centers about a known root process problem. They may now choose to examine the expected behaviours of the controlling physical device system to see if there are any questionable device activities. Another choice is the recall of experience knowledge about the chlorine dioxide controlling devices. By examining the experiences the operator may conclude that the manual excess flow valve has probably become stuck in the closed position.

The presented knowledge representation architecture and its operation are very different from the approach taken presently by expert system designs for process monitoring. The proposed architecture is a homogeneous structure that houses multiple forms of reasoning and knowledge representations. Present process monitoring expert systems either are homogeneous structures of a single form of knowledge representation, or a few forms of representation grouped into units that are modularity applied. Formentor (Pennings & Saussais, 1993), is an example of the former case. All the knowledge and reasoning activity are represented in the form of a decision tree. The MIP project (Alaman et al., 1992) is an example of the latter case in that four forms of representation are hierarchically employed. EAGOL (Pople et al., 1994) sits in between the other two, using only one form of knowledge representation, but grouped into modular units that are procedurally applied.

Another difference is that the proposed architecture decouples thinking and information knowledge, and encapsulates the thinking behaviour in thought process to perform the reasoning. Present knowledge architectures convolute the thinking and information knowledge in their representations, and reasoning processes are in almost all cases a fixed predetermined sequential procedure. The MIP project (Alaman et al., 1992) and EAGOL (Pople et al., 1994) are developed using rule-based knowledge. An implementation of a rule, by its nature, convolutes of both how, and on what information, to make an inference. At a higher reasoning level, these projects both apply fixed sequential or layered deployment of reasoning activity modules. Formentor (Pennings & Saussais, 1993) uses decision trees that represent stepwise decision. The decision tree knowledge is applicable only to the specific domain it is created for. The NASA Space Shuttle system (Ingrand et al., 1992) is an exception to fixed reasoning in that it hierarchically deploys knowledge agents to address confronted goals and problems. The system can reason through a situation in a non predetermined manner. However, the *knowledge agents* consist of procedures that embody specific knowledge of the process structure. The information knowledge is not decoupled and hence a *knowledge agent* is limited in the problems it can be applied to. Courdier and Herin-Aime (1989) also proposed a system architecture which is similar to the Space Shuttle. This system faces the same differences to the proposed architecture as the Space Shuttle expert system. The SOAR architecture (Laird, 1984; Laird & Newell, 1983; Laird et al., 1986; Laird et al., 1987; Rosenbloom & Newell, 1983) is another exception, being a representation for very open reasoning processes. SOAR employs a full range of goal directed thinking processes that generate sub task thinking processes to deal with unknown problems. However, the philosophy of SOAR is to construct a machine that develops its own thinking patterns from trial and error rather than represent the thinking expertise of humans. One supplies it with task knowledge in the form of operators of a problem domain. SOAR is a general problem solver in the sense that it can be applied to any problem if one defines the possible operators for the problem domain. SOAR then learns how to solve the problem by applying the possible operators at each solution state, until the gaol state is reached.

SOAR offers little benefit as a process monitoring expert system architecture because it would be difficult to identify and describe the process domain in terms of operators and states. However, many of the concepts inherent in SOAR are extremely potent and would provide much functionality for other architectures.

The proposed architecture emphasizes the principle of representing and reasoning about the world abstractly through use of abstract objects and system representations. Almost all present applications of expert systems to process monitoring represent the process world as physical components and systems. The MIP project (Alaman et al., 1992), for example, encompasses a knowledge level that represents the process by equipment components and their interconnections. Other examples of process monitoring systems that represent the process domain by the physical components can be found in Betlem & Aggele, 1994; Gamble, 1993; Lee et al., 1992; Thomson, 1988; Rong et al., 1988. A few research systems have taken the approach of abstract process knowledge representation. Courdier and Herin-Aime, 1989 develop objects which "describe an abstract and concrete element of the real world." The NASA Space Shuttle system (Ingrand et al., 1992) is another that expresses the use of abstract entities, "RCS-Controller takes an abstract view of the domain; it deals in pressures and valve positions."

The proposed architecture specifies and uses a true object oriented knowledge base where objects encapsulate domain *knowledge* and *intelligence*. Present object oriented expert systems for process monitoring employ an object oriented database in which only the attribute information of an object is represented. This is true with all the leading vendors such as Gensym, Talarian, and Comdale.

The proposed architecture represents and applies episodic experience and other forms of experience in a natural manner. Experience knowledge with most present architectures is piece wise extracted from the human's knowledge and implemented in convoluted reason and information knowledge representations. Experience is seen as part of the knowledge and no differentiation is distinguished. Neither episodic experience nor thought process experience are directly applied by present process monitoring systems.

A further difference is that the proposed architecture represents knowledge of the internal interrelationships of a system as well as the structure of it. Present process monitoring expert systems do not capture this knowledge but represent systems through input-output models or by causal observation knowledge of a system. Both MIP (Alaman et al., 1992) and EAGOL (Pople et al., 1994) are examples of this. The MIP project uses quantitative mathematical models to represent the behaviour portions of the process system. EAGOL uses causal models to predict the behaviours of a process system.

Lastly, the proposed architecture represents natural thinking routing between various conceptual entities within the knowledge. Such routing (associations) can link situation experience knowledge to situation handling experience knowledge. Present process monitoring expert systems to not represent *thought process* behaviour, thus natural thinking routes can not apply. The NASA Space Shuttle system (Ingrand et al., 1992) is built with similar concepts to *thought processes*. It does not represent, however, natural forms of experience nor a homogeneous knowledge body in which to mentally route through.

In general the proposed architecture presents a more natural manner of representing human knowledge in a computable system. On the other hand, present process monitoring knowledge representations tend more to force the user to have to fit pieces of knowledge into computer like forms. The proposed architecture takes these hard to fit, but necessary to represent knowledge forms, and reintroduces them in appropriate places in a structure that is similar manner in which a human consciously perceives his knowledge.

There are many benefits offered by the proposed architecture. A homogeneous knowledge base is more flexible making it easier to modify and effectuate changes. Further, the system can be more powerful because the strengths of the object oriented representation structure can be applied globally. The implementa[illegible] thought processes captures and applies the human's ability to thoroughly reason t[illegible] a problem or face new situations. The decoupling of reasoning and information knowledge provides the capability of applying problem solving methodologies to a variety of similar situations. It also makes it easier to identify and implement the necessary and desired private knowledge of a human. Through the design of encapsulating intelligence in objects, a homogeneous object oriented knowledge representation could be achieved. In addition, the representation can account for the inconsistencies of a world. The representation of episodic experience in natural form allows the full potency of it to be directly applied to situations. Representing the internal interactions of a system provides the ability to access knowledge of any portion of a system. Using the object oriented representation furnishes the ability to represent layers of systems within systems, thus defining varying depths of knowledge. The architecture captures the human mental mechanism (associations) that links conceptual entities within the mind. This mechanism establishes effective and efficient links from current problem contexts to experience knowledge, and chains together further cause-and-effect relationships known through experience. The associative mechanism provides a mental routing through knowledge that reasoning can explore to search for solutions. The architecture in representing many aspects of a human's world of conceptualization in a manner similar to the natural form facilitates the ability for a human to relate their knowledge and implement it in an expert system. It also offers the ability to represent and benefit from the powerful mental activities of a human.

3

Case Study: Bleach Plant Expert Process Monitor

Although the proposed new knowledge representation architecture offers an intuitive completeness to an examiner, the practical test of its worthiness comes from its success when applied in the real world. Towards this end a project was arranged with industry to provide them with an expert system process monitor, and that would give the university the opportunity to test new expert system innovations in an industrial environment. This project formed an arena for a case study of proposed new knowledge representation architecture. A prototype expert system based on the architecture was developed and applied to the monitoring of a bleach process of a pulp mill. The performance of the system and other targeted features were observed to meter the achievements of the proposed architecture. The following three chapters discuss the case study project.

3.1 Project Overview

The project came into being by a joint agreement among the Peace River pulp division of Daishowa-Marubeni Int., MoDo Chemetics Int., the Canadian Alberta Partnership in forestry, and the Intelligence Engineering Lab at the University of Alberta. Daishowa-Marubeni Int. is the owner of a Kraft pulp mill at Peace River. MoDo Chemetics is a supplier of an industrial process management information software system. The agreement was formed around the development of a prototype expert system to monitor the process operations of the bleaching process in the Peace River pulp mill. For their part of the agreement, the mill provided partial funding for the project and access to their process for university personnel to test and apply a prototype system. MoDo Chemetic's interest in the project is to develop the prototype into a commercial product that can be offered to other customers, and to supply ongoing support for the system installed in the mill. MoDo Chemetics donated a DEC^{TM} computer system and their MIS software for the development of a prototype system. The Intelligence Engineering Lab is executing the project, providing the technology, supplying the expert system development tool, Metacoop, and a user interface system, INTEMOR, for the project. The rest of the funding is being provided by the Canadian Alberta Partnership in forestry.

The scope of the project was narrowed to constraints that were felt achievable as a first step in developing a pulp process monitoring expert system. These constraints came largely from Daishowa-Marubeni who wanted to ensure the initial success of the system in their mill. The purpose of the expert system was defined to detect oncoming or occurred incidents, to give warning of them to the operations personnel, and to supply on-line information of how to avoid or handle the incidents. Knowledge for the expert system was to be drawn from the mill in the form of *incident* reports that are compiled by mill personnel. This constraint was relaxed during the knowledge acquisition phase to include the personal experiences of mill operators. It was found that the personal experiences were more accurate and practical than the administrative reports. Employing the incident reports implied that the prototype system was to derive its intelligence from previous experience knowledge, thus requiring the expert system to represent experiences in the form of episodes. A suggested limit of twenty incidents was proposed for the initial installation.

The restricted scope imposes some limitations on the ability to evaluate the full knowledge representation architecture. In particular, the implementation of *system* knowledge is not included in the scope. Further, only one thinking methodology towards determining the source of a process problem and of what action to take is employed by the defined scope, that being to recall applicable previous experiences. It is felt that the full knowledge representation architecture presented in this thesis is needed to accurately replicate all the process monitoring intelligent activities. However, only the portions of the architecture that align with the scope of this project were implemented and practically tested. In regard to the *system* representation, an initial bench test prototype system was constructed. A test process system representation was implemented in this prototype. The prototype included a *thought process* to select which methodologies to utilize to determine the source of a process problem. This prototype system was successfully executed in a lab setting. It was not included in the project development prototype as process system representation was not going to be used. It is felt that a further project is needed to test this and to test advance thinking behaviour concepts of the architecture, such as the second level thought processes. Note that these concepts are deemed to be important and integral components of an overall knowledge representation architecture, and should not be viewed in isolation. To meet the speed requirement of executing the project's prototype system in real time, the process *monitoring thought process* has been partially coded into a C routine. This was done after the methodology was implemented and tested as a knowledge base *thought process* object. The final prototype system is a practical expression of most of the aspects of the knowledge representation architecture.

The history of the prototype development is the following. The conceptualization of the knowledge representation architecture occurred during the summer and fall of 1993. The initial prototype, which represents a system, was constructed and tested during late winter and early spring of 1994 at the University of Alberta. The development of the mill prototype system to monitor the pulp bleaching process was carried out over the summer of 1994 at the Peace River site. Also performed at site were the knowledge acquisition and the incident report verifications. Computers and office space at the site were provided by Daishowa-Marubeni. Late fall saw a return for two weeks to clean up loose ends in the system and to review the knowledge base. At this point the system was complete and left to execute in real time. Another follow-up trip to site in early February of 1995 was made to monitor the prototype system performance and to train the mill personnel on how to expand the systems knowledge. Some system bugs were corrected and several modifications to the knowledge base were made. This trip saw the installation of the system into the operations control room as an operational aid. The system is presently in operation at the Daishowa plant.

3.2 Millwide System Overview

The mill data and information system consists of DEC computers and PCs sitting on an computer network. The DEC computers are clustered using the network protocol DECNet™[1]. The PCs operate under Windows operating system and run PCSA™[2] for communication with the DEC equipment. The DEC computers all operate under VMS™[3]. The PCs have local hard disks and can also use the DEC hard disks for file storage.

The main software components composing the process information system are the DCS, MOPS, EDE/2, and an assortment of PC based utilities such as Excel.

DCS: The DCS is the mill distributed process control system. The DCS employed by Daishowa is a Honeywell TDC-3000™[4] system. The system comprises proprietary control computers, data highway and operator consoles. Plant process data is fed into the DCS.

[1] DECNet is a registered trademark of Digital Equipment Corporation
[2]PCSA is a registered trademark of Digital Equipment Corporation
[3]VMS is a registered trademark of Digital Equipment Corporation
[4]TDC-3000 is a registered trademark of Honeywell Corporation

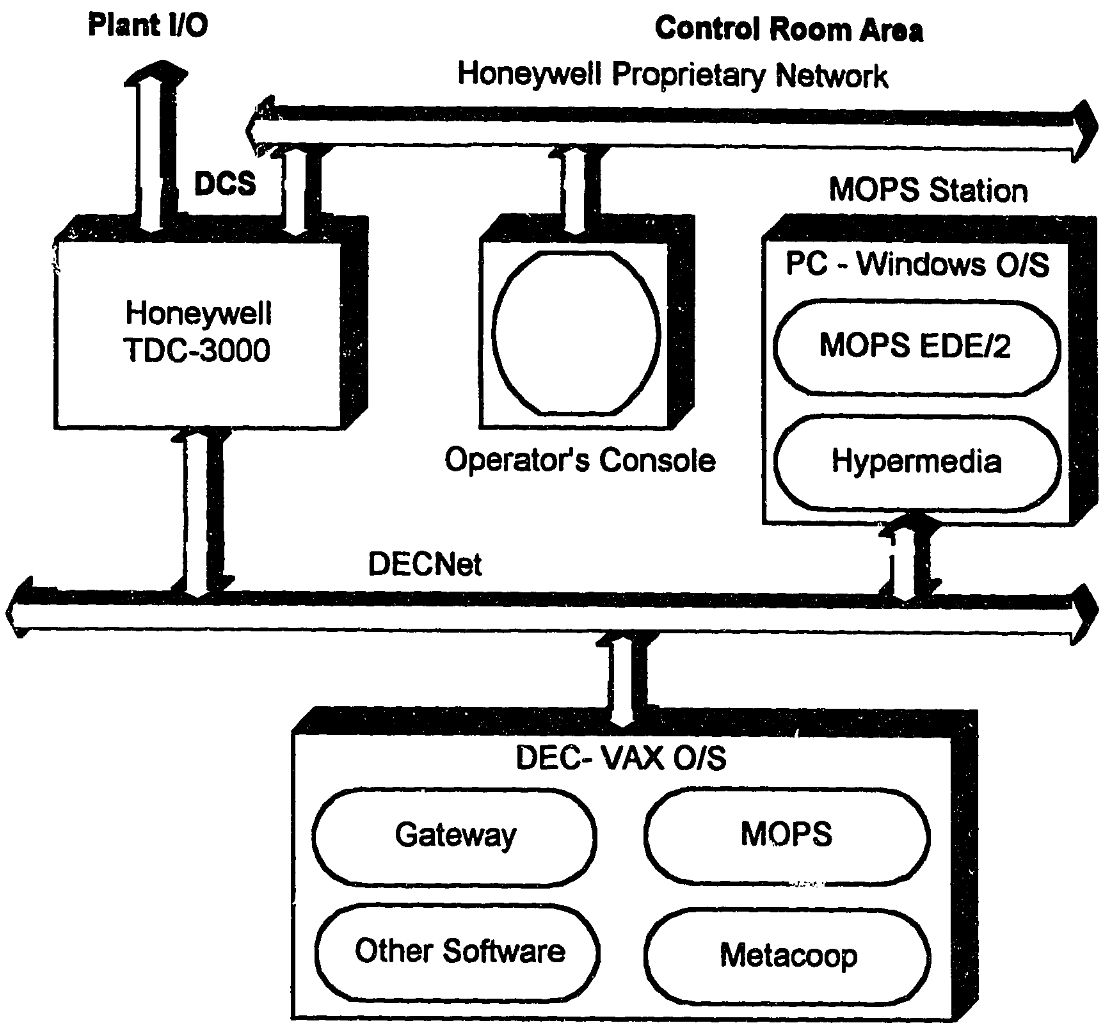


Figure 3.1: Millwide Computer and Information System

MOPS: MOPS is a management information system developed by MoDo Chemetics. MOPS collects data and information from various mill sources and provides enhanced data and information to operations and management personnel. The information is presented in graphical form at MOPS stations or as printed reports. MOPS compresses and stores data into a historical database (HDB) for later analysis. Current data is retained in DEC internal RAM memory in a current value database (CVD). MOPS offers a library of API routines that can be embedded in user developed programs to access the data. MOPS routines execute in a DEC VAX™ cluster environment.

EDE/2: EDE/2 is the MOPS user workstation software that resides in a PC. EDE/2 communicates with MOPS via DECNet™. EDE/2 offers graphical information such as dynamic process graphics, trends, and report display. Display screens are developed for EDE/2 through a display development utility. All MOPS data is available for trends, control of graphic entity, and for display. EDE/2 screens can also be configured with user input variables. Presently at Daishowa, EDE/2 is being used for the lab data input. EDE/2 workstations are remotely placed throughout the mill for process operators and management personnel.

The process data and information flow originates with the DCS acquiring field transmitter data. MOPS polls for DCS data through a software communication link called Gateway™[5] at frequent periodic time interval. This data is written into records in the CVD. Lab input and other data are also written into the CVD. A calculation handler software process is activated periodically to perform calculations on the data. The CVD data is scanned and stored historically at intervals. For the pulp and paper industry, intervals between scans are around 30 seconds. When a user calls up a screen, a request for that screen data is sent to MOPS. A minimum size display information vector is sent to EDE/2 over the DECNet™ The screen information is displayed. The display is automatically refreshed frequently while it is active.

[5]Gateway is registered trademark of Hoeywell Corporation

3.3 Implementation of the Knowledge Representation Architecture

The expert system development tool chosen for this project was Metacoop, a system developed by the Intelligence Engineering Lab of the University of Alberta. Metacoop is a frame-based system designed after object oriented concepts of *class* structure, *inheritance* and *messaging*. Each frame consists of *value* slots, *rule* slots and *methods*. A frame is an *instance* of a class frame. Slots can be inherited if defined as *memberslots*, or over written if defined as *ownslots*. Metacoop supports multiple inheritance of other frames. Data can also be stored external to the frames in *global* variables that are universally accessible. All data is internally represented in ASCII with the data type information carried for converting to externally pass values. String, real, and integer data types are recognized. The knowledge base is pseudo compiled into binary trees to store the frame attributes, rule sets, methods and global variables. The rules are typical of other procedural based expert systems. They are combined into rule sets that can be executed by a method. Methods are procedures that are similar in syntax to programming languages. A method is initiated by sending a message requesting the method to a frame that contains that method. A message encodes the frame name, the desired method to be activated and parameter data for the method. Both the rules and the methods can call external routines. The source code for Metacoop is written in C. The major reason that this system was selected was due to its openness. All the source code was available to develop the functionality and mold the system to meet the project's needs.

Metacoop has much the look and feel of the object oriented paradigm, and it can provide a foundation to implement the proposed knowledge representation architecture. Mapping an object oriented structures over a frame-based system has been done in the past and is not an uncommon move (Vossos et al., 1991). Metacoop frames can be shaped into objects by observing the principles of the paradigm. The *characteristics* of an object are represented by the value slots, and the *behaviours* of an object by methods and external programs. Ownslots of a frame satisfy the concept of polymorphism of objects, allowing the custom expression of a unique nature for a particular object. *Intelligence* can be internally realized in a frame through the incorporation of rule sets and calls to external intelligent routines in a procedural method. Efforts can be made to ensure that the frames *encapsulate* all information of the object it was representing.

Thought process objects can be developed out of frames. The *thought process* itself can be represented by a method. Rules and external intelligence routines can be called upon by the method for intermediary decision making. Note that rules are the only built in *thinking mechanisms* of the Metacoop system. *Thinking activities* for *thinking tasks* can be realized as other methods within the frame. A method can summon other methods by sending a message; a *thought process* can activate other *thought processes* or *thinking tasks*. The execution of a reasoning process can be hierarchical. Implementing second level *thought processes* is same as implementing first level *thought processes*, only the contents is different. Reasoning using generic objects is accomplished through passing the generic object's name to a *thought process* object (frame) in the initiating message. The methods of the *thought process* use the passed name when necessary to access or request information from the generic object. The interface of an object *type* is defined by the slot and method names, and each *type* (class) of objects has a defined interface which is known to the *thought process* objects. By sending an appropriate message to the named generic knowledge object, an internal method can be executed to determine the behavior of that object. The method of this object can summon other internal methods to set parameters, or other requirements that adjust for the present context of domain, and thus deals with its nonmonotonic nature.

Situational experience consists of information and behaviours that can be represented by value slots and methods. In addition, for this project external hypertext files are linked (associated) to the object for futher detailed information. Recognition can be implemented in Metacoop as rule sets that pattern match the data with recognition rules. The rule sets can be embodied in a recognition method inside a frame (object). The behaviour method of an experience object can summon the recognition method. If required, external routines such as neural networks could be called from a recognition method to perform other manners of pattern matching. Expectational experience can also be realized by rule sets. The rules capture knowledge in the form of "rules of thumb" about the nature of a domain. For the project, expectational experience was implemented through the combination of rules and an external routine. The external routine, call **Stat_set**, forms a *thinking mechanism* that can be employed by intelligent activities to evaluate expectational experience.

Associations, relationships and object composition can all be implemented through the same techniques as discussed above. Association, relationship and composite object names can be recorded in value slots. Methods with rules and external routines can intelligently describe the associations, relationships and system behaviours. Thought process objects can send messages to the knowledge objects to obtain information of associations and relationships.

3.4 Integration of Metacoop into the Millwide System

It is necessary to integrate Metacoop into the Daishowa plant computer system and data flow environment to be able to develop a real time process monitoring expert system. Metacoop was originally available for the PC, thus the source code had to be transported to work under the VMS™ operating system. The code was mostly standard C and directly transportable, however, modifications were made such as to reduce the amount of stack verses static memory used. The Main routine was rewritten to include a shell that would run the expert tool continuously. This shell utilized VMS™ operating system features to execute more efficiently in the VAX™ computers. VMS™ logicals were employed in header files to define all directories relative to a root, thus making the code independent of hard drive location. COM files were created to define user friendly commands for compiling, for using the expert system, and for viewing log files. A COM file was created to run the expert system in detached mode. This file was interwoven into the VAX™ system startup and system monitor files. The system monitor watches for dead processes and attempts to restart them. As part of the integration, two directories were set up. One directory is for the real time executing expert system, the other is for a testing expert system that executes in a one shot mode. The testing system does not impact the continuous system and can be run simultaneously.

The integration of the Metacoop into the process information flow was achieved through MOPS. MOPS API library routines were used to access data and pass results back into the MOPS database. A routine, MOPSData, was created to fetch a snapshot of the process data from the MOPS CVD every time the expert system executed. The expert system results are written into the MOPS CVD by a routine called Report. Report routine also records the events into a disk file. An error logging routine was written to record run time errors into a disk file. Both of these files can be viewed at any time by the user. The distribution of the results to the users is done through MOPS workstations. An EDE/2 display was developed to show a summary of the expert system results. This display is integrated with a hypermedia system, INTEMOR, that can be seamlessly called from the EDE/2 display. The MOPS workstations are well distributed through out the mill.

Main routine can either be run to compile the knowledge base or to execute the expert system. The compilation of the knowledge base coverts the configured text files into a binary tree file that can be loaded for execution. The execution of the expert system begins by calling MOPSData and Report routines to initialize. The knowledge base is then loaded into memory. The program now begins a loop of the following activities. The MOPSData routine is called to capture a snapshot of MOPS process data. The expert system core tasks are then executed through the process monitor *thought process* which is partially code as the Operator routine. Upon completion of the expert system tasks the Report routine is called to report all events to MOPS and to the log file. The system then goes into hibernation for 35 seconds while the alarm signal communication from MOPS to the DCS is allowed to occur. The operating system wakes up the main program that calls the Report routine again to recheck the active events to see if any are no longer active. The time taken for the ongoing program execution is kept. When the Report routine completes, the difference between the execution interval for the expert system and the elapsed execution time is calculated. The expert system goes into hibernation for the calculated time period.

MOPSData routine has two modes, initialization and execution. The initialization mode creates a link list of pointer address to MOPS and Metacoop memory for data fetches. A text configuration flat file defining MOPS points is opened. An error listing file is created and opened. A channel to access to the MOPS database is created by a called MOPS API routine. The MOPSData routine creates the link list by reading each configured MOPS point name, allocating a new link structure, and fetching the MOPS point database pointer. The program then creates a global variable in Metacoop and retains the pointer for that. Both pointers are stored in the link structure and the program continues on to the next specified point. If errors are encountered the allocated structure is removed and an error message specifying which MOPS point is written to the error listing file. The final success of the initialization is recorded at the top of the error listing file. For the continuously executing expert system, any MOPS initialization errors will terminate the expert system's start-up. In the execution mode of the MOPSData, the routine uses the created link list to quickly fetch MOPS data. The program steps through each link using the pointers to read the MOPS data and store it in the Metacoop variable. The MOPS data is converted into ASCII before it is stored. Any errors encountered while fetching data are recorded in the run time error file.

Report routine has three modes of operation, initialization, report and recheck. In the initialization mode the program creates global variables where active incidents are to be recorded by the expert system. *Note that only incidents which change state from inactive state are recorded by the expert system.* The program also obtains the MOPS CVD database record pointers where the results will be written. These pointers are stored in arrays that are dimensioned to number of incidents that can be shown on the EDE/2 user display. The event log file is created, opened, and a time stamp header is written into it. The file is then closed to allow user access. In the report mode the global variables are checked for new incidents. These incidents are the experience object names. The information that is to be reported is fetched from the object and stored in arrays. The information is written into the arrays in cyclical manner. Any incidents about to be over written that are still active are removed and inserted with their information into a link list structure that is dynamically allocated. When all new incidents are processed the result information in the arrays is written into the MOPS CVD records, from the newest to the oldest incidents. On the display screen the results appear to *push down* previous results. The report mode ends by writing an alarm signal, if applicable, into the MOPS CVD which is transferred to the DCS to set an audible alarm. The recheck mode of the Report routine steps through all the active incidents in the arrays, and along the link list of removed incidents. The program sends a *message* to the incident experience objects and requests them to determine if they are recognized to be inactive. The program sets the display array incidents inactive or removes the incident structure from the link list if they are found to be inactive. The program also sends a message to check the status if the incident experience is still active. If the incident is determined to have *increased* its status, such as going from a warning state to an occurred state, the MOPS database is updated. If the increased status occurred for a removed incident on the link list, that incident is pushed back into the display arrays and reappears on the user display. The recheck mode of the Report routine is terminated by writing an alarm reset signal in MOPS to reset the DCS alarm. The DCS audible alarm will continue, however, until it is acknowledged by the operator.

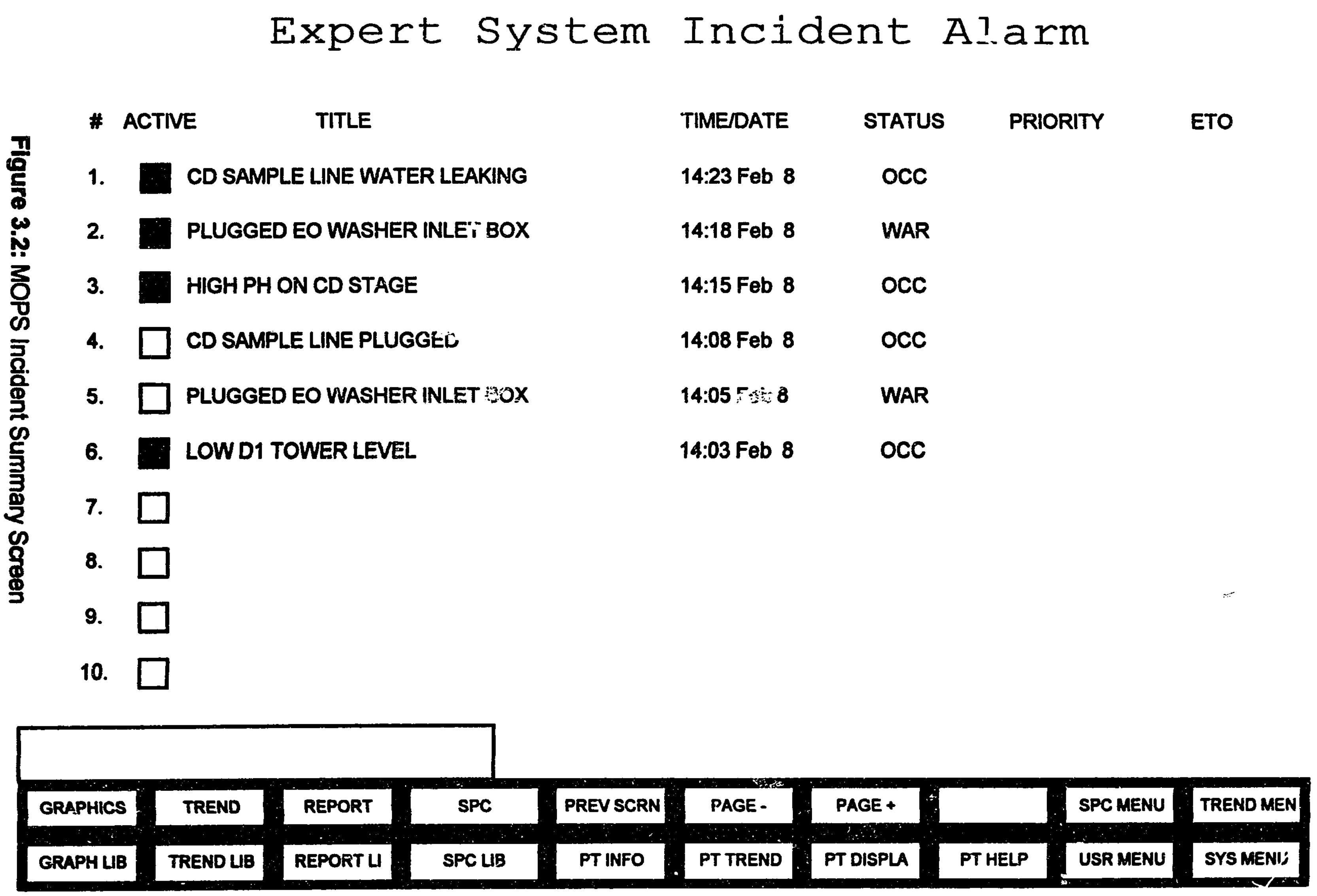
Error Logging Routine records run time error and message information into a log file that the user can view. The run time error routine initially creates and closes a file to establish it presence. Each time a message is passed to the routine it fetches the current time, attaches that in front of the message, and writes the message to the file. The file is opened to allow the write, then it is closed to allow user access to it. Error messages include the inability to fetch MOPS data (data has been mark invalid in MOPS) or a request for variables that has not been defined. Special debug messages, such as the time taken for each execution of the program, are also written into this file if the DEBUG flag is set in the Main header file.

3.5 User Interface

The purpose of the user interface is to supply the expert system results to the operator. The user interface is strongly tied in with the MOPS system. When a process monitoring result has occurred, the expert system writes an alarm signal into the MOPS which is transferred to the DCS. An audible alarm is heard by the operator and an alarm message is appears on the console that indicates that there are new expert system results on the MOPS workstation. A dedicated key is configured for the MOPS workstations that brings up the incident summary screen (screen showing expert system results). This EDE/2 screen lists recent incidents and summary information about them (Figure 3.2). This information includes the title of the incident, time of occurrence, status of the incident and whether the incident is active or inactive. The active or inactive state of the incident is depicted by a box beside the title. The box is displayed in red if the incident is active, green if inactive. The status specifies whether the incident is seen as a warning of possible occurrence, a warning of immanent occurrence, or an alarm that the incident has occurred. For the project implementation the ten most recent incidents are shown. The screen information is all drawn from the MOPS CVD where the expert system writes the results. The operator can select an incident title on the screen and by clicking a menu box, can call up the associated INTEMOR screen. The hypermedia system is automatically initiated if it is not presently running. The operator is presented with detailed textual information about the cause, possible consequences, and preferred action that should be taken (A Hypermedia incident information screen can be seen in Figure 3.3.) Displayed also on the screen is a snap shot of selected relevant process data from the time the incident was reasoned to be active. This data provides the operator with a means of verifying the expert system results. The INTEMOR display can have hyper links to other topics (screens) that supply further information. Such a display could be of equipment operating procedures, of mill equipment tagout locations. Editing the displayed incident knowledge is done separately from the Metacoop knowledge base configuration.

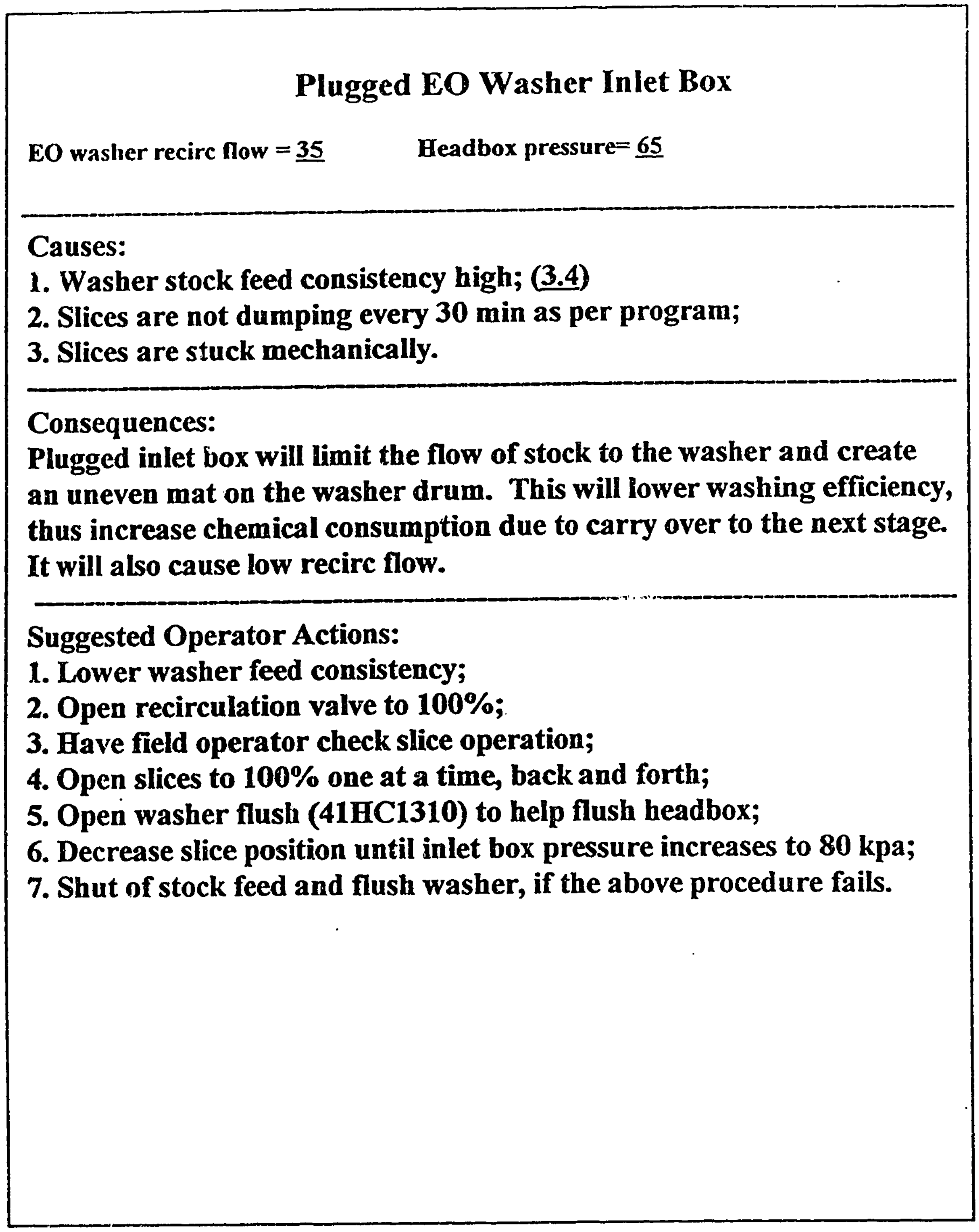
INTEMOR is being developed through the Intelligence Engineering Lab of the University of Alberta. The INTEMOR development uses Borland C++ with Windows™[6] API. The

[6] Windows is a registered trademark of Microsoft Software Corporation


Expert System Incident Alarm

#	ACTIVE	TITLE	TIME/DATE	STATUS	PRIORITY	ETO
1.	■	CD SAMPLE LINE WATER LEAKING	14:23 Feb 8	OCC		
2.	■	PLUGGED EO WASHER INLET BOX	14:18 Feb 8	WAR		
3.	■	HIGH PH ON CD STAGE	14:15 Feb 8	OCC		
4.	□	CD SAMPLE LINE PLUGGED	14:08 Feb 8	OCC		
5.	□	PLUGGED EO WASHER INLET BOX	14:05 Feb 8	WAR		
6.	■	LOW D1 TOWER LEVEL	14:03 Feb 8	OCC		
7.	□					
8.	□					
9.	□					
10.	□					

Figure 3.2: MOPS Incident Summary Screen



Plugged EO Washer Inlet Box

EO washer recirc flow = 35 Headbox pressure= 65

Causes:
1. Washer stock feed consistency high; (3.4)
2. Slices are not dumping every 30 min as per program;
3. Slices are stuck mechanically.

Consequences:
Plugged inlet box will limit the flow of stock to the washer and create an uneven mat on the washer drum. This will lower washing efficiency, thus increase chemical consumption due to carry over to the next stage. It will also cause low recirc flow.

Suggested Operator Actions:
1. Lower washer feed consistency;
2. Open recirculation valve to 100%;
3. Have field operator check slice operation;
4. Open slices to 100% one at a time, back and forth;
5. Open washer flush (41HC1310) to help flush headbox;
6. Decrease slice position until inlet box pressure increases to 80 kpa;
7. Shut of stock feed and flush washer, if the above procedure fails.

Figure 3.3: INTEMOR Incident Information Screen for a Plugged EO Washer

screens have the look and feel of Windows^{TM} programs. The software system sits over top of a Metacoop expert system foundation. Hyper links between screen entities and frame attributes can be made. This has the advantage of being able to have externally updated variables embodied into a display. The communication of the external variable data is accomplished by embedded routines in the Metacoop portion of INTEMOR. Hyper links can also be made to methods which can invoke intelligent activities to occur. INTEMOR resides on the PC MOPS workstations.

The hypermedia system was integrated with MOPS EDE/2 to provide a fluid user interface. Most of the integration development work was performed by MoDo Chemetics, however, routines were written by the Lab to fetch data for INTEMOR. When more information about an incident is initiated at the EDE/2 display, a request for the CVD data for that incident is sent off to the VAX^{TM}. The VAX^{TM} resident MOPS sends back the information in the form of a text record . The Report routine constructs the text record for each incident. The record includes the name of the INTEMOR executable program, the INTEMOR document file name to be opened, the incident name, and data in the form of the variable name followed by the value (Figure 3.4). The topic name for a INTEMOR screen is given the same name as the experience object (incident) in the expert system. EDE/2 captures this returned record and writes it into the Windows clipboard. A terminate-and-stay-resident program is initiated that parses the program and document file name and activates the program in the Windows foreground. INTEMOR, when called forth, executes a routine that reads the clipboard and parses the rest of the message. The appropriate INTEMOR display (topic) is summoned and data is written in the matching variables. The user now is viewing the hypermedia screen.

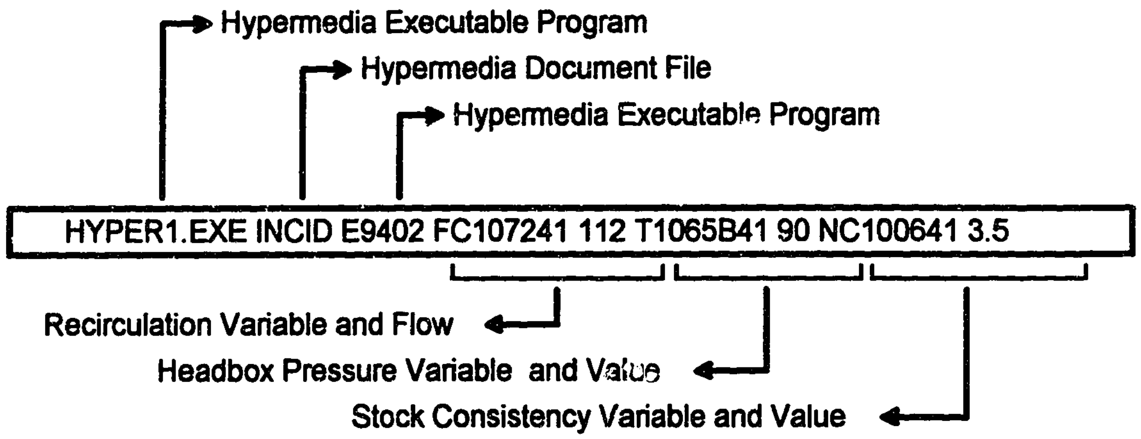

Hypermedia Executable Program
Hypermedia Document File
Hypermedia Executable Program
HYPER1.EXE INCID E9402 FC107241 112 T1065B41 90 NC100641 3.5
Recirculation Variable and Flow
Headbox Pressure Variable and Value
Stock Consistency Variable and Value

Figure 3.4: INTEMOR Communication Record

3.6 Engineering Interface Enhancements

Several enhancements and modification were done for the engineering interface. The Metacoop knowledge-base compile error messages were examined and edited for clarity. Several new messages were created to more precisely locate compilation errors. More work could be done on the compilation error detecting. A compile message listing routine was developed to list knowledge base compilation activities and errors into a disk file. The compile messages within Metacoop were modified to pass their messages to the new compiler log routine. The log routine opens up a file when it is initiated. The name of the log file is the same as the knowledge base configuration file being compiled, except the file extension of LIS is used. The routine writes a header into the file with the knowledge base file name and time of compilation. All the passed messages are written into the file. These messages include what slot, rule set, method, and frame is being processed. Error messages are listed in the file in appropriate sequence to the portion of the knowledge base being compiled. When compilation is complete, the routine records the success of the compilation at the top of the file, and closes the file. The file provides a crisp organized listing of the knowledge base entities and detected compilation errors.

Another enhancement was the development of a testing system. The expert system is identical to the real time system except the test system executes once only and the results are written to test CVD records. The test system can use the same MOPS CVD process data as the real time system. There is no impact on the real time system by the test system. A second EDE/2 screen was developed for the test system, thus creating a full testing environment. A special thought process knowledge base object was designed that would display, and store in files, status information about the execution of the reasoning. Information is stored for in files for each object that is processed. The supplied information is helpful to verify that the system is reasoning as anticipated.

4

Case Study: Expert Process Monitor Implementation

The implementation of the expert system process monitor mainly comprises the development of the proposed knowledge architecture into the prototype system. This is accomplished through developing a knowledge base structure from the concepts of the knowledge representation architecture. This structure then provides a repository for knowledge to be assimilated. For the proposed architecture, both the thinking knowledge and the information knowledge structure must be developed. In addition, for the project's prototype, the development of a thinking mechanism was required. This chapter discusses the developed knowledge base structure and its components. The chapter also presents the knowledge acquisition process and the development logistics of the project.

4.1 Process Monitoring Thinking Methodology

For the project, the thought behaviours employed in monitoring a pulp bleaching process needed to be ascertained and implemented in accordance with the proposed knowledge representation architecture. To identify the process monitoring thought behaviours, mill operators were observed and questioned. From this, an overall process monitoring methodology was determined for the process operations at the Daishowa Peace River Pulp mill. This methodology is depicted in Figure 2.5. It was found that the thinking knowledge consisted mainly of *situation handling* experience. Also found was that the operators depend heavily on and utilized *situational* experience knowledge when analyzing the process conditions. The reported incidents from which the mill wished us to draw the expert knowledge formed part of the body of all recalled experiences. Recalled situations included personal experiences as well. The implementation scope was expanded to encompass these. The experience recall aspects of the overall process monitoring methodology concurs with the scope constraints established for this project. Thus, the methodology implemented in the prototype system was the portion of the overall process monitoring methodology that involved recall of experiences. The methodology for the prototype system is depicted in Figure 4.1. (It was observed that the more skilled operators often tend to reflect on technical process system knowledge when making decisions about actions to take, or when diagnosing and predicting problems.) The following recounts the implemented bleach plant process monitoring thinking methodology.

A process operator through experience develops a selection of process points that they monitor. With the present state of technology, the monitoring is usually performed through observing measured values for the process points on a graphical display. The operator has an expectation of what those values should be, given the present operation conditions of the plant. If any of the points appear to be in an unexpected state, he will recall his process experience associated with that process point, based on the present condition of the process. The scenario of each experience will be recalled. They will then observe measured values of other process points that are involved or indicative of the experience occurrence. They would have expectations to the values of these process points as well. From the observed conditions of all the involved process points he would draw a conclusion of whether the experienced situation is, or will be, occurring. A level of certainty about the conclusion would manifest. If the situation will or has occurred, they will recall knowledge of the implications of the situation and base the immediacy of their action upon that. The knowledge of how to act on the situation also comes into mind, as well as any situations that may arise from the actions. Once necessary actions for a situation are executed, the operator will continue monitoring the key process points.

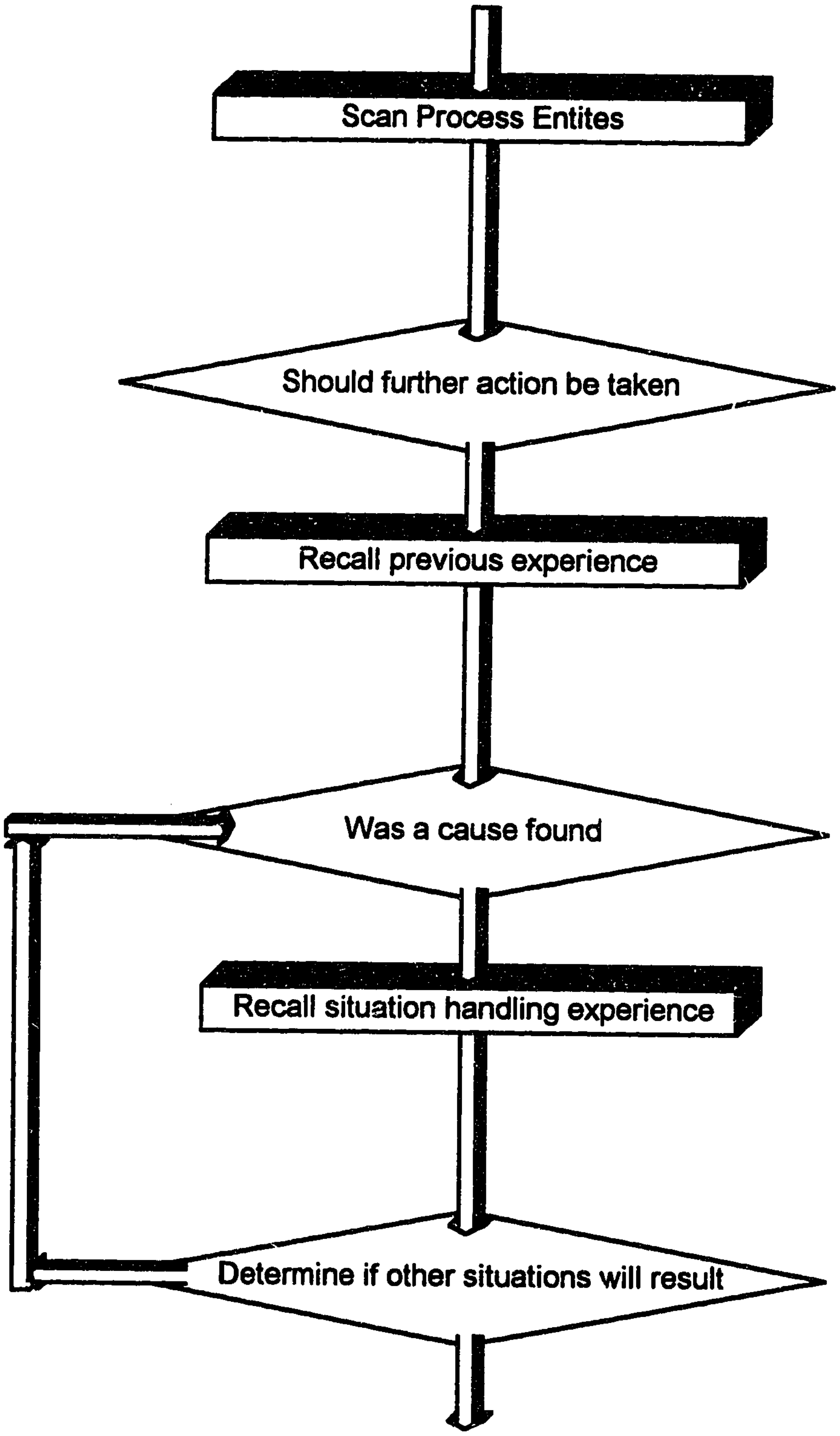

Scan Process Entites
Should further action be taken
Recall previous experience
Was a cause found
Recall situation handling experience
Determine if other situations will result

Figure 4.1: Protoype System Process Monitoring Thinking Methodology

This process monitoring thinking methodology was implemented as a *thought process* object in the prototype system. As a *thought process* object, the system was executed and was successful in its ability to reason. However, the system was felt to be too slow for the real-time mill environment due to the software structure of Metacoop. Since the was content with the present implementation, expansion of the process monitoring thinking methodology would not occur for some time. Thus, it was deemed practical for the prototype system to convert the thought process to a hard coded C routine. The routine was given the name Operator. This routine will be discussed in detail next. Figure 4.2 depicts the program flow of the prototype process monitor expert system.

Operator Routine:

The Operator routine was developed to implement the process monitoring thinking behaviour. The routine begins by opening up a process object scan file. This file represents the knowledge of the key process points that a mill operator surveys. It is a flat text file of the process objects and can be organized so the more important objects are monitored first. The routine reads each process object in turn and sends a message to that object to determine its status. The status of the process object is read. If the status is *bad*, (that which constitutes *bad* is discussed in Section 4.2.), then the routine reads from the object the name to the associated experience (incident) objects. The routine accesses the associated experience objects in turn. The applicable conditions description for the experience are fetched and compared with the status conditions of the process object. Applicable experiences are selected from the group of associated experiences. For instance, the chlorine flow process object may determine that its value is ***high*** and out of an expected operating range. This status information of the process point would be compared with the description of its associated experiences. Such incidents could be a plugged chlorine mixer inlet or a water leak in the brightness analyzer sample line. The

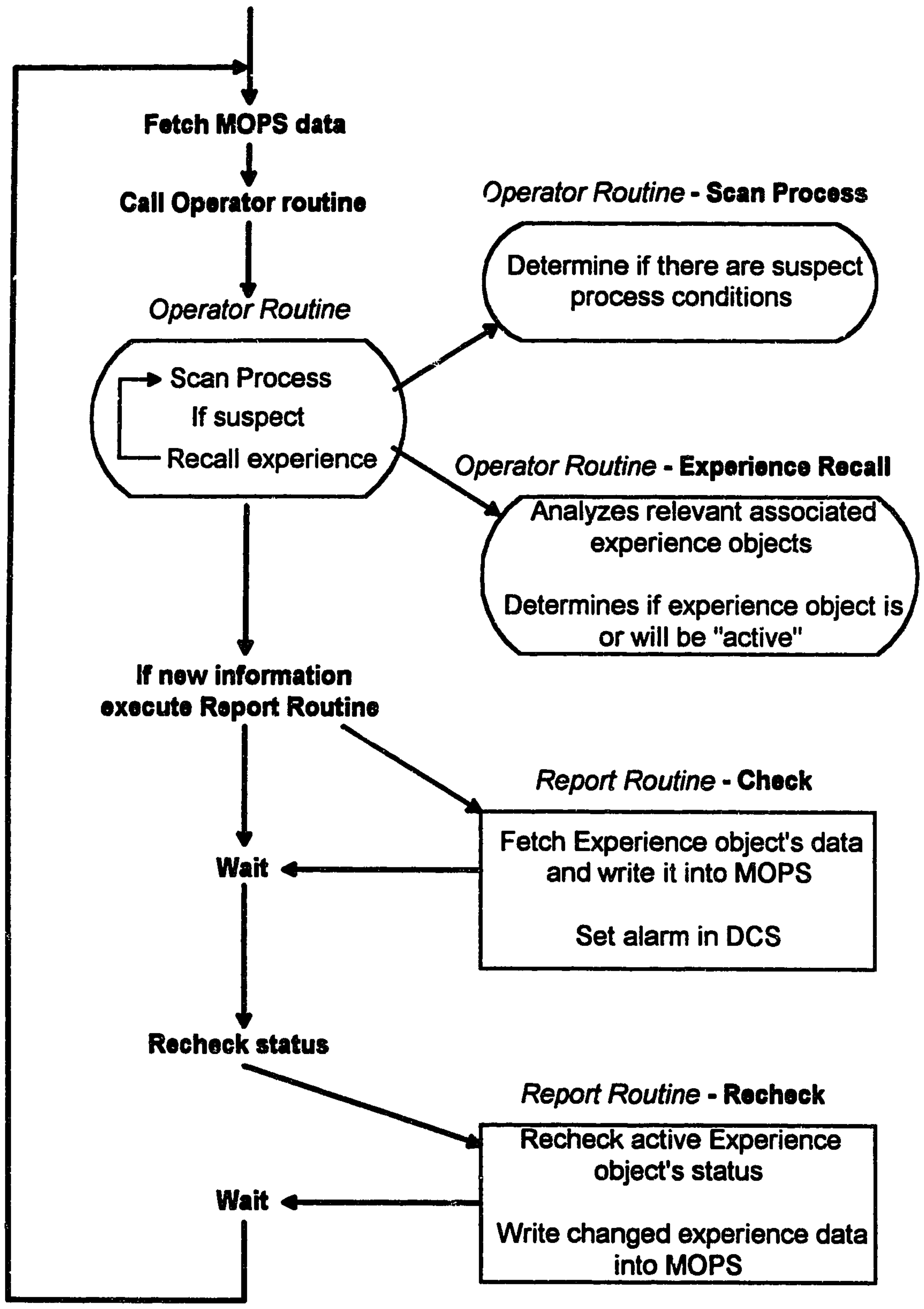

Fetch MOPS data
Call Operator routine
Operator Routine
Scan Process
If suspect
Recall experience
Operator Routine - Scan Process
Determine if there are suspect process conditions
Operator Routine - Experience Recall
Analyzes relevant associated experience objects
Determines if experience object is or will be "active"
If new information execute Report Routine
Report Routine - Check
Fetch Experience object's data and write it into MOPS
Set alarm in DCS
Wait
Recheck status
Report Routine - Recheck
Recheck active Experience object's status
Write changed experience data into MOPS
Wait

Figure 4.2 : Prototype System Flow Diagram

plugged chlorine mixer incident would be eliminated from the possibilities as it is applicable when the chlorine flow is low. The sample line water leak is applicable, and is selected out. Experiences can also be selected by which operation mode of the mill they apply to, such as startup, production, or shutdown mode.

The routine then sends a message to the applicable experience objects requesting them to check their status. The experience check status behaviour summons the recognition experience knowledge and concludes if the experience is or will occur. The Operator routine reads the status of the experience object. If the experience is or will be active it records the experience name to be reported by the Report routine. The routine can also read any other associated experience objects that could come from handling the detected experience. Again, the conditions are compared. In this case the associated experience may only be applicable when the detected experience has *occurred*, but not if it is predicted to happen. The routine continues until all the scan file objects are dealt with. It is worth noting that the Operator routine thinking behaviour is applied on generic objects, the selection of which object is provided by the scan file.

4.2 Knowledge Base Configuration

The configuration of the knowledge base for the prototype system comprises the objects necessary to accomplish the previously discussed process monitoring thinking behaviour methodology. The information included in these knowledge base objects would be greatly expanded for a commercialized system. A more inclusive set of object *types* would also be developed. Figure 4.3 shows the knowledge representation architecture for the prototype system. The prototype knowledge base is composed of *device*, *process*, *experience*, and *thought process* objects. The *thought process* object, Operator, is a *situation handling* experience object and was discussed in Section 4.1. The other objects and their inclusion

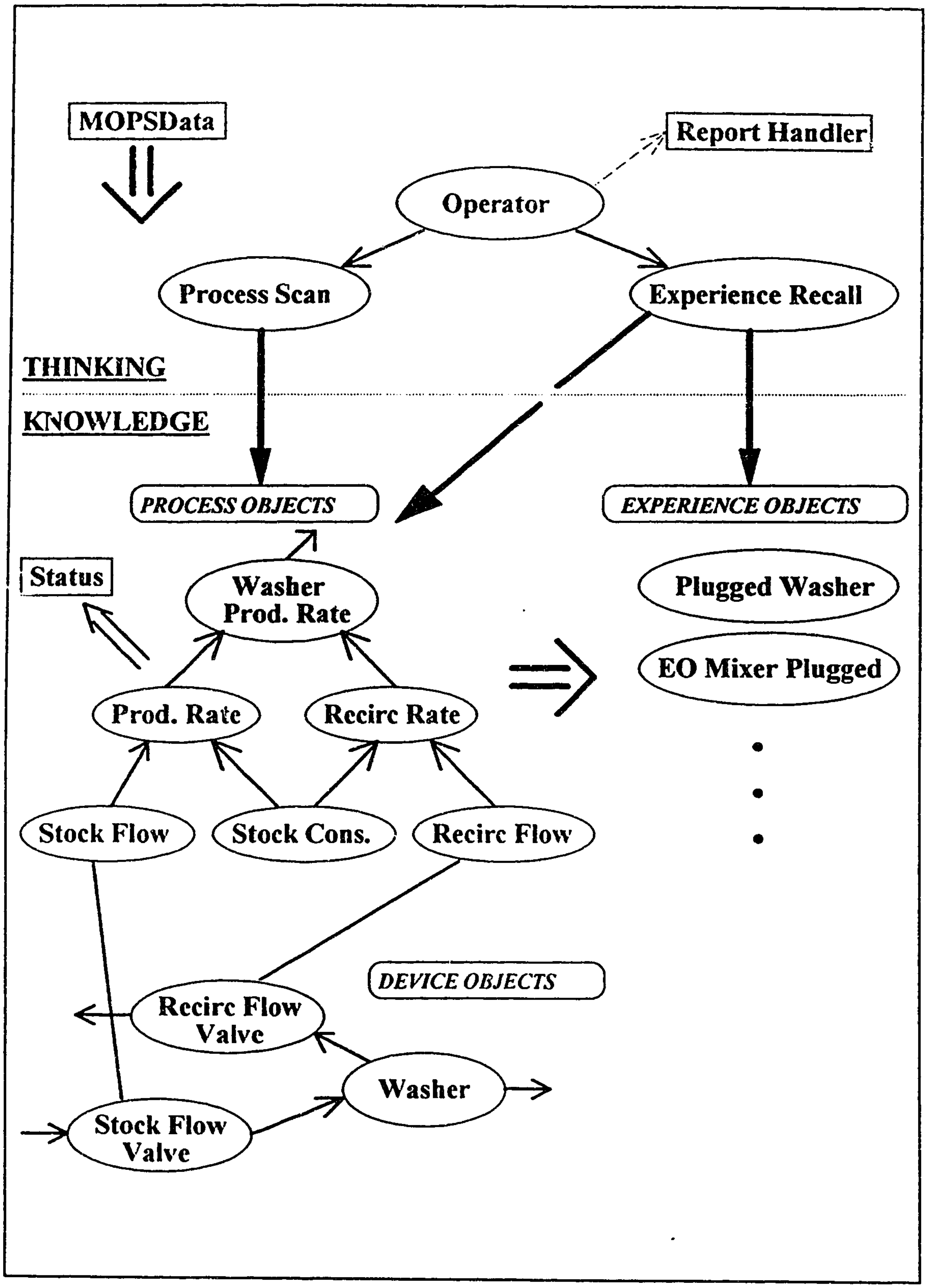


Figure 4.3: Prototype Knowledge Representation Architecture

into the knowledge base in accordance to the proposed knowledge representation architecture are discussed in this section.

The knowledge base design can be summarized as follows. *Device* and *process* type objects exist to describe the physical equipment and the process knowledge of the pulp bleaching plant. These object *types* contain *expectational* experience that is used by a status method (behavior) to determine the status of an *instance* of a *process* or *device* object *type*. The *expectational* experience uses an external intelligence routine (*thinking mechanism*) to evaluate its knowledge. The user can overwrite (polymorphism) the use of the external routine in any *instance* of an object *type* and specify its own status determining manner. The status method of these object types is called **stat**. (See Appendix D for details of the various methods employed.)This method initially executes reasoning to determine a present quantitative value for the object. This value is used to conclude heuristic qualitative attributes that describe the status of the object. Associated with these object *types* are *experience* objects. The *experience* object *types* contain *recognition* experience knowledge that is used by the **check** and **reset** methods. The **check** method uses *recognition* experience knowledge to identify if the experience is or will occur. The **reset** method uses knowledge that will recognize when the experience is no longer occurring. The *experience* object types also embody knowledge to link (associate) to INTEMOR topics. A INTEMOR topic contains the *situational* experience knowledge in the form of editable textual information. The topic also enshrines the *situation handling* knowledge in textual form. The *situation handling* knowledge is intended to guide the mill operators in handling the situation. The ideology of an expert process monitoring system is to keep the operator "in the control loop". Experience knowledge of a situation and how to handle it is updated through editing the text with knowledge learned through repeated situation encounters. The following paragraphs discuss the main attributes of each of the objects.

Process objects are created for each pertinent process entity of the pulp bleaching process, or for a combinations of process entities that can be unified. (An example of a configured process object is included in Appendix B.) The object encapsulates, through frame variables, status and value characteristics of the object. A method, called **stat**, determines these characteristics. When configuring the object, the user specifies either the MOPS data point name, a value, or a blank in the frame's **value** variable. The method reads the variable and responds accordingly. If the specified entity is a MOPS data point name then the method fetches the present value from the global variable for the MOPS point. If there is a value specified for the object's value variable then the method uses that value. If the variable is left blank then the method calls upon a rule set, **vset**, which the user can configure to perform a necessary calculation to determine a value. The rule set can activate external routines if required. After the method has ascertained a quantified value for the object it then proceeds to evaluate a status for the object. The status can be determined by utilizing a thinking mechanism that works in conjunction with expectational experience, or it can be left up to the user to configure a method. If a frame variable, **ustat**, is specified as yes, then the thinking mechanism is employed. The thinking method is an external routine called **Stat_set** that will be discussed further in this section. If **ustat** is specified no, then the method calls upon a rule set, **sset**, which the user can configure. Both methods conclude heuristic values that describe the present status of the object. These value reside in a set of frame variables. A rule set, **rset**, is used by the expectational experience methodology to adjust for the present process operation conditions when specifying the expectations for the process object. This rule set can also be employed by user configured methodology as it is activated by the **stat** method before the **ustat** value is checked. **Rset** is discussed later in conjunction with **Stat_set**. Other characteristics and behaviours could be added to the design of the object at the user's discretion. The associated experience objects of the process object are identified by frame variables labeled **case_#**. The number attached after the underscore can be any value as long as the numbers are contiguous. The thought process method takes a nonexistent number in the sequence as the end of the list, thus the number of **case** variables does not have to be pre-dimensioned.

Device Objects have the same interface as process objects for the needs of this prototype. The major difference is that there is usually not a quantifiable value that can be attached directly to a device object. The technique employed to quantify the device object's behaviour is to relate data surrounding the device's operation in some fashion. For instance one manner of evaluating the status of a process control valve is to look at the difference between the setpoint and the measured value signal, especially if the valve output is at maximum. One can use the expectational experience on an indicating quantified value. However, it is simpler to configure a rule set that quickly concludes the status of the device. Device objects have associated process incident experiences as well, and the associations are specified in the same manner as the process objects above.

Experience objects are created for each situational experience or process incident. (An example of a configured experience object is included in Appendix C.) These objects contain a set of description values that are configured into frame variables. The description values describe the applicability of the object. The object also contains a method to determine if the experience is, or will be, active. This method sets a status frame variable, **stat**, to either active or inactive. The **stat** variable is set through the method executing a rule set, **check**, which is recognition experience in the form of rule pattern matching. The user configures both predictive and existence recognition experience in the **check** rule set for each object If the recognition experience knowledge is dependent on the status of a process object in the knowledge base then a **stat** message is sent first by the rule set to that process object to update its status. The process object's status values can then be used. The inferred result of the rule set is put in a frame variable,

cond, which has the cardinality of *warning*, *imminent*, or *occurred*. Before the **check** rule set is executed, the method calls upon a rule set, **vset**, in which the user can configure adjustable variables. These variables are adjusted for the present domain context and are used in the inferencing of the object's **stat** and **cond** values. The object embodies a **recheck** method as well to determine when the experience is no longer active. This method calls upon a rule set, **reset**, in which the user configures the required recognition experience. The **reset** and **set** conditions for an experience should be configured to provide enough leeway to prevent cycling. The situational experience knowledge for the object is located in an INTEMOR text file. The system links the experience and the INTEMOR file through the object name. The data display information for the INTEMOR file and the MOPS summary screen are configured in the frame variables, **title** and **disp_epn_#**. Like the **case_#** variable, the **disp_epn_#** forms a non-dimensioned array by incrementing the attached number in a contiguous manner.

Class Objects are employed to enhance the robustness and ease of knowledge base expansion. The interface for each of the three object *types* discussed above is the same for all instances of object. The object *types* are object classes. Much of the functionality and definition that makes up an object is standard and not user configurable. When creating a new object there is no need to be concerned with many of the attributes. To reduce confusion, class objects were designed that contain the complete necessary body of attributes to execute the prototype system. The attributes are given default values. It is possible to create a new object solely by defining it as an instance of the class and it will function within the system. It would, however, contain no practical intelligence. To implement intelligence and specifications for an object, the user configures *ownslots* for the attributes that need to be specified. The core methods are developed to execute behavioural reasoning in the manner specified by the scope of the project. The user of the system is not required to develop these. The user influences the behaviours of an object and the system through configuring a select number of rule sets and specifying some key frame variables. This is practical for an industrial application where the users generally do not have the time to delve into the intricacies of expert systems. They require the functionality of a system for a particular application with an ease to make it work. If the user wishes to gain control of a method he can launch off of a rule set by embedding an external routine or a summoning a user defined method. By doing this, the rule set determines which *thinking tasks* to employ and is the decision making step of a *thought process* that determines an object's behaviour. Some of the transparent activities and variables that the class object encompasses, but are not required to be dealt with by the user, are those that deal with runtime information. Each method of an object records the last runcode (an incrementing value) of when a particular message was received for it. The stored run code is checked against the present system run code every time a method request is received. If the two are equal, the method is not executed, thus saving redundant processing. This is especially true of the status request which could be generated several times within a single execution of the system. Also recorded is the last grade and operation mode at the time the method was run. The grade and mode are checked before the rule sets are summoned and compared with present conditions. Two variables are set, **xgrade** and **xmode**, which can be used by the user to control the rules set activation. These features are all aimed at focusing and reducing the necessary processing activities of the system. (The prototype configuration of the class object is included in Appendix D.)

To ease user knowledge base development confusion, the configurable knowledge base entities were separated from the default. The class objects are grouped into a file call Class.fra in the executable directory and are transparent to the user. The process and device objects are contained in a file called Process.fra, the experience objects are in a file called Incident.fra. Both of these files are in a work directory for the user to develop.

The files have example object configurations to guide the user. At the top of the file the user specifies the instance of the class. The rest of the file is the definitions for each instance of the ownslots to configure.

At the present state of the project there are no configured examples of *situational* experiences associated with *situational* experiences. As well, there are no *process* objects being associated with other process objects. These lacking features are due solely to the level of knowledge acquisition and level of thinking behaviour achieved at this step of the implementation of the prototype system. Associated *process* objects are one way of evaluating a suspect process status further. The associated *process* object could be a combination of process entities which more succinctly define a process condition. The status of such an object could be executed when a single process point is suspect, and thus narrow in on the process condition. Another concept is to embed in the *process* object the knowledge of how to deal with a suspect condition. The *thought process* can utilize the object encapsulated knowledge to directed how it reasons the situation. The advantage is that each object can be unique in how it is dealt with and that the knowledge is localized in the object. Many other techniques can be employed if the proposed knowledge representation architecture is established as the knowledge base design. (Erlenbach, 1994a)

Stat_set Routine:

Stat_set routine is a *thinking mechanism*. **Stat_set** derives heuristic values which describe a process entities' status based on a quantified value and on an operator's expectation experience of that process entity. The expectation experience is represented as a linear range, demarcated by expectation limits (Figure 4.4). These limits express expectation boundaries that an operator has of a process entity's value. By comparing the quantified process value with the range limits, semantic values for several heuristic

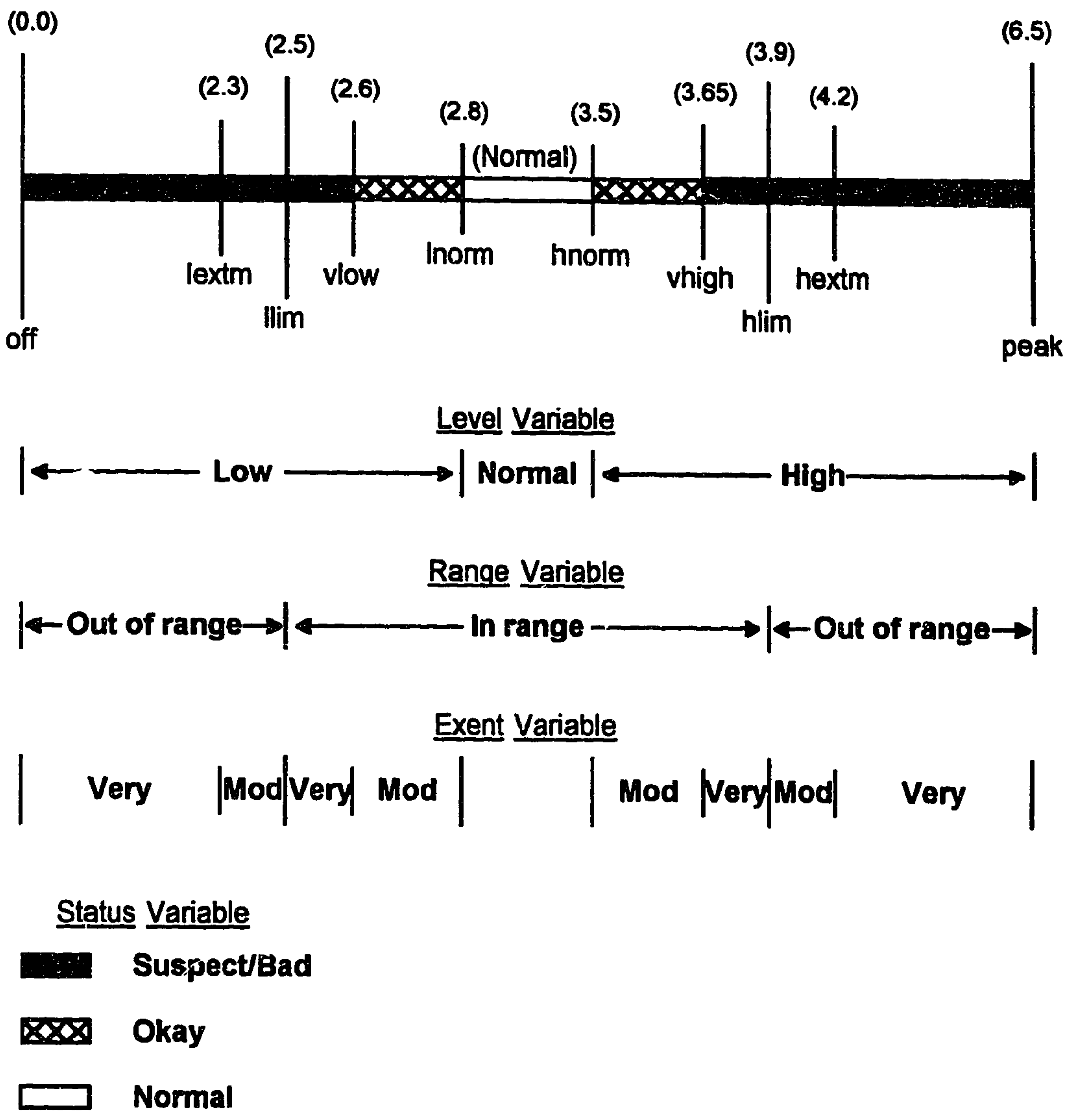


Figure 4.4: Expectation Experience Range and Variables

variables are determined. These values describe the process entities' status relative to an operators' expectations. The specifications of the range limits are retained in frame variables. These variables are: **off**, **lextm**, **llim**, **vlow**, **lnorm**, **hnorm**, **vhigh**, **hlim**, **hextm**, **and peak**. **Off** and **peak** demarcate the full extent of the possible values. **Llim** and **hlim** represent the maximum expected range that the process value would take. *Note that this is not an alarm limit but an expectation limit.* **Lnorm** and **hnorm** specify the expected range that the process value would normally take. **Vlow**, **vhigh**, **lextm**, and **hextm** express the extent that the process value lies outside of the normal and outside of the expected range. These limits can also be regarded as boundaries where the operator becomes watchful, and extremes that the process value can take before drastic measures need to be undertaken. The values for these limits are dynamically determined through a rule set, **rset**, that is summoned before **Stat_set**. This rule set allows the user to configure reasoning to adjust the limit values for the present process context. Establishing the value of the process entity was discussed previously under **process object**. The expectation limits permit at least three heuristic variables to be concluded, **range**, **level** and **extent**. The range variable describes whether the process value was found *in* or *out* of expected range. The level variable expresses if the process value is *higher*, *lower* or *normal* to expectations. The extent variable adds extra information stating whether the process value was *very* or *moderately* beyond the normal or beyond the expected range. The **Stat_set** routine compares the process value, determines the appropriate semantic values, and stores the information into the frame variables. **Stat_set** is called by the status method after the process value and the expectation range limits have been established. The **Stat_set** can be seen as a *thinking mechanism* that is applied to *expectation* experience knowledge which is represented by limit values. The determined status information for the process entity is used in reasoning by the *thought processes* and by other objects

The **Stat_set** routine has some other features to it as well. It is not expected that operations personnel will supply values for all the mentioned limits. In some cases not all the limits apply. The routine will calculate or infer the missing limits for these situations. A minimum of three, but often four specifed limits are necessary to generate a range map. If the minimum number of specified limits do not exist, then an error message is logged in the run time error file. At least the **llim**, **hlim** and one of the **lnorm** or **hnorm** limits are require. The routine will use the available limits to conclude the status values that it can.

In the header file for the **Stat_set** routine, one specifies a percentage split for the *moderate* and *very* extents. This is a default value that is used when other information is lacking. The **Stat_set** routine sets the **stat** frame variable as well. **Stat** can take on values of *ok* and *bad*. This variable is a general indicator of how the process entity fairs up to expectations. It is used by the Operator thought process to cue further thought action to be taken. No action is taken if the value of **stat** is *ok*. If time series analysis was able to be performed on the measured process data then a semantic value of *potential* could also be assumed by **stat**. In order to develop a predictive nature for the determination of a *bad* status, and thus activate reasoning activity, the definition of *bad* was decreed to be any status beyond **vlow** or **vhigh**. As a result, whenever the process value becomes very high or very low, but still in range, it is considered suspect and reasoning actions are initiated. The definition of *bad* is configurable in the **Stat_set** header file.

4.3 Knowledge Acquisition

The knowledge acquisition was focused around a fill-in-the-blank form. This form was developed to allowed the operations personnel to express the required knowledge to configure the expert system. The form provides a media for human knowledge to transcend into the expert system. The expert system knowledge representation under the proposed architecture already takes on a similar knowledge form as is consciously perceived by a human. Therefore, what is necessary for knowledge acquisition is to focus on the specifics of the required knowledge. The form was used towards that end to facilitate and automate the acquisition of human knowledge for the system. (Knowledge acquistion form is included in Appendix E.)

The project's knowledge acquisition process started with an initial a stack of mill incident reports that were submitted as potential pulp bleaching process experiences. This stack amounted to around two hundred reports. These reports went through a pre-selection process based on useful content. The reduced selection, about fifty, reports were then reviewed with some of the mill engineering staff. These talks did not offer much new information, but did generate much uncertainty about the content of the reports. Much information was lacking or incorrect. The next step was to review the reports with several mill operators. These meetings proved very informative and the omissions in the report were revealed. From the stack of reports, twenty of the incidents that had practical value were identified. The evaluation meetings also had another effect, that being of demonstrating that there is a wealth of practical experience knowledge with operators. Further, this experience knowledge is easily represented by the proposed architecture. At this point in the knowledge acquisition process, the forms were introduced to the operators. The operators, in light of the discussed reports, independently completed the forms, supplying the required detailed information. The forms were returned and the content was examined. There were only a very few items that required clarification. From the form's information, the prototype's knowledge base was directly developed, both the objects and the INTEMOR topics. The most difficult aspect of this work was the tracking down of MOPS data point names. The MOPS system was going through a version upgrade at the same time as the expert system project.

Once the first prototype system was developed and tested, and compile errors eliminated, the knowledge base was reviewed. Some issues engaged in during the review were the layout of the INTEMOR displays, and the specifying of the expectation ranges. It was found that the operators tend to have different preferences on how to operate the process. This difference could be resolved by a series of group meetings to clarify each operator's reasoning, or by correlating the more successful operation methodologies. It is worth noting that the difference can often emanate from such reasons as a fear or hesitation by an operator to run the process in any thing but a very liberal safe operating conditions.

Over three review sessions the knowledge base was tuned and refined. During this time our own development of the concepts of the architecture stabilized. The knowledge base modifications were performed with the operator on hand. Through this work the operator developed the ability to understand and configure the knowledge base. The mill management has decided that a technical support person should work in conjunction with the operations personnel in developing new knowledge. It was clear, however, that the operator could perform all aspects of implementing knowledge himself. *An operator could directly implement their knowledge into the prototype system.*

For the end product, two of the twenty reports were abandoned as physical equipment could eliminate the problem. Six operator experiences were adopted. Twenty-four incidents are presently configured in the prototype system.

4.4 System Testing, Training and Startup

The testing of the system was done by executing the system on line in a back-room environment. Each added knowledge base incident was compiled and tested separately, and then incorporated into the overall knowledge base. All software development went through a typical compile and debug state. Real-time data was accessible throughout the testing stage. The system output screens were not directed to the process control room. The DCS alarm was not enabled either. The screen outputs were viewable only on the computers used for the development. This provided a safe development environment that did not impact mill operations. After the prototype system was installed, the system's activities and behaviours were monitored for a period of several months. The logged events were saved. From the run time testing, several knowledge base and program bugs were encountered. The corrections were implemented. In early February 1995, the expert system was deemed sufficiently stable to be introduced in the control room. By this time, several of the operators were completely familiar with the system and were to instruct others on operating the user interface. During the period of final modifications, a member of the technical staff worked on performing the knowledge base changes. This provided them with the confidence of having the ability to execute changes. A system problem still exists which can be rectified and is being attended to. The system user interface is now loaded into the control room and the system is running on a twenty-four hour basis.

5

Case Study: Results

The results from the pulp bleaching process monitor prototype system can be split into three different categories. The first category regards the results from the implementation and integration of the system. The second looks at results that manifested from the acquisition and implementation of knowledge. The last category is concerned with the results from the operation of the prototype system. These results demonstrate the success from different perspectives of the prototype system as well as of the proposed knowledge representation architecture.

The results from the implementation and integration of the prototype system indicate the ability to apply the system in an industrial setting. Questions answered by the results are, "can this system be implemented in an industrial plant," or "can the system be operated by industrial personnel." The results for this category are discussed in Section 5.1.

The results from the acquisition and implementation of knowledge for the prototype expert system reflect the success of the proposed knowledge representation architecture and its goal to mirror the consciously perceived representation of knowledge by a human. The aim of the architecture is to provide an expert system knowledge base structure that is understandable by a human, thus promoting the ability for it to be expanded, modified, and revisited. Section 5.2 discusses the knowledge acquisition and implementation results of the project.

The results from the prototype system's operation indicate the usefulness of the system as a process operations aid. This is also an indication of the effectiveness of the proposed knowledge architecture to replicate and execute valid human knowledge. The results answer such questions as, "did the system work; did it produce valid and useful conclusions," or "was it unable to produce conclusions that it should be able to." These results are presented in section 5.3.

5.1 System Evaluation

From a system evaluation point of view this project resulted in an expert system fully integrated into the control systems and the operations structure of the Daishowa pulp mill. The expert system is fully embedded into the mill's computer and information system, and it generates insignificant loading on the computer systems. The expert system is executing on-line and is continuously monitoring the pulp bleaching process. The user interface has been seamlessly interwoven with the mill's management information system's user interface, EDE/2, in the PC environment. The expert system detects and provides early warning monitoring of process incidents, and supplies a simple and user friendly operational aid to handling the incidents.

> "The screens are easy to read and give the operator inside knowledge that he may not be privy to. The options that he is given to react to certain incidents allows him to make educated decisions." (Dan Bowes, Operating Technician I, Daishowa-Marubeni Peace River Pulp Mill)

The final installed prototype system consists of twenty-four incident objects, thirty-eight process objects, and sixty process data points. The prototype system has been viewed and approved by both the mill manager and the production superintendent of the Daishowa Pulp mill. A letter by the production superintendent acknowledging the project is attached in Appendix F.

In addition, the prototype work achieved the defined scope of the project. The system is based on operator's experience and accumulated mill incident reports. The operator's process monitoring thinking methodology experience has been assimilated into the expert system to guide the reasoning process. The ability to *predict* incidents based on experience knowledge has been realized. The knowledge base of the prototype expert system is maintainable by the mill personnel.

Lastly, the implementation of the proposed knowledge representation structure was accomplished. The installed expert system has been developed in a way that represents and executes the major aspects of the proposed architecture. The implemented concepts include:

- an object oriented structure for all knowledge
- associative linking of knowledge
- a thought process object to represent thought behaviours
- decoupling of thought and information knowledge
- employment of situational, expectational, and situation handling experience
- application of reasoning on generic objects
- intelligence incorporated in objects
- intelligent determination of the nature of an object
- conceptual expression of the process rather than a physical viewpoint
- more than one thinking mechanism is employed.

The configuration of the knowledge base by the user was simplified through utilizing the capabilities of the architecture's structure to create user transparent default objects.

Overall, the prototype system is successful in providing a functioning application of an expert system in a real world industrial environment, and to be a test system to explore the concepts of the proposed knowledge representation architecture. This is a very difficult task as often the needs of the two perspectives conflict. The complexity of the knowledge base had to be limited to an accepted practical scope for the industrial system. This prevented all aspects of the proposed knowledge representation architecture from being tested. Also, a certain level of system robustness is necessary for an implementation of a software system into industry. This demanded that much of the development time be spent programming. The prototype system, however, does pull together both sides of the project. It implements the proposed knowledge representation architecture. All observations and activities of the prototype are reflections of the architecture. The expert system is seamlessly integrated into the pulp mill's control and information systems. This is a notable achievement itself.

5.2 Knowledge Acquisition and Implementation Results

The results of the process monitoring knowledge acquisition revolves around observations of the facility in which valid knowledge can be identified, specified and implemented. In addition to these results, there were some general observations noted from the process of acquiring knowledge. The observations were of expected, but are often overlooked truths that manifest in the process of acquiring knowledge. It was found that the mill technical staff were not familiar with the details of the process operations concerning the incident reports. (Though they were suggested to be the first personnel to meet with.) The technical staff tended to have general experience concepts and analyze situations from a theoretical point of view. Often their memory of an incident, or knowledge about how the process is supposed to be run, was very different from that of the operations personnel. The senior operators tended to regard the process in terms of the operating the equipment, and often seem to visualize the equipment when analyzing a situation. Many experienced operators portrayed a depth of knowledge when deliberating on process operations in that they could mentally chain sequences, and see different levels, of *cause and effect* that would occur. These operators showed much knowledge as to what will not work, and what will happen if an action is taken even if the action is *theoretically* correct. The operators, understandably, had a much better awareness of the limits and idiosyncrasies of the equipment. On the other hand, many of the operators did not seem as quick to grasp the concept of temporal thinking, which seemed second nature to the technical staff. The idea that an upset at the Digester needs to be addressed at the Chlorine stage six hours later, and that preparations should be made for that often are not perceived. (It became apparent that applying an expert system towards this end would to be a very useful application.) Another observation, which quite commonly occurs, is that the knowledge acquisition process uncovered some organization communication deficiencies. The knowledge acquisition process provided an information exchange between organizational levels of the mill. Several times the participating personnel were quite surprised by what was uncovered and were left wondering why the proper information had not reached the appropriate persons. In a few cases, by simply talking about the incident it became apparent that there were simple remedies. These observations are rather typical, but they should not be overlooked when planning the knowledge acquisition phase of an expert system implementation project.

The results of ability for process monitor knowledge to be identified and related for the prototype system were very positive. Technical and operations personnel became quickly familiar with the knowledge representation concepts, such as process objects, experience objects, recognition experience, associations, etc. Both groups found it easy to express and verbalize their experience and knowledge in these terms. The brainstorming and review sessions were all comfortably carried out by expressing the knowledge in terms of the knowledge base structure. Knowledge base modifications were done directly from the discussions. The operators were very receptive to the concept of expressing recognition and expectational experience in relation to varying process conditions. They felt that this was a missing ingredient in many systems. The determined and implemented process monitoring reasoning methodology was well heralded.

> "It is a commonsense approach, I do not think that there is another realistic approach. The expert system brings a more humanistic approach to dealing with day to day problems. When configuring these incidents one takes a logical and systematic approach to come to viable solutions." (Dan Bowes, Operating Technician I, Daishowa-Marubeni Peace River Pulp Mill)

> "The approach this system uses is very logical. It's a common sense way to monitor the process: watch various tags and when they are out decide if there are any effects. It makes configuring simple; the only information needed is how do you know a tag is out and how do you know an incident is occurring (or about to occur)." (Katrinka Scott-Burns, Process Technician, Daishowa-Marubeni Peace River Pulp Mill)

The ease at which knowledge could be specified definitely led the participants to be encouraged and with the system and comfortable with the new technology.

The development of a form to guide the knowledge acquisition process, was very successful. Technical and operations personnel found it easy to relate their knowledge on the form. From the form, knowledge was directly implemented in the expert system.

> "The incident report form is clear and relates directly to the logic needed for the process and incident variables. This form makes it easy to get the necessary information to configure a new incident." (Katrinka Scott-Burns, Process Technician, Daishowa-Marubeni Peace River Pulp Mill)

The operations personnel had similar comments though added the caution when filling out the form that, "people must realize when filling out these forms that they must be total and complete with information" (Dan Bowes, Operating Technician I, Daishowa-Marubeni Peace River Pulp Mill). The implications of these results are that the proposed knowledge representation architecture is similar enough to the human's that knowledge can be passed and implemented in an expert system without the need to have it translated. This has several ramifications which were observed. The first is that the need for specialist to translate knowledge is eliminated. As a result knowledge can originate from the operations personnel. In fact, it was found that once the operator's became involved in expressing their knowledge of incidents they seem to pour forth with very useful personal experiences that they would like to see implemented. This is possible and is facilitated with through using the knowledge acquisition form. Also, the process of specifying and implementing knowledge can be automated, and does not need the supervision of a specialist. Knowledge can originate and be recorded on the form by an operator. This knowledge can be reviewed and discussed among other pertinent personnel. It then can be passed to the appropriate staff to effectuate its implementation.

(*It has been suggested before that operating personnel are hesitant to share their knowledge, however, this was not noticed. For this project the operating personnel appeared proud that their knowledge formed part of the expertise of the system. The hesitation may have been alleviated by the fact that it was clearly realized that the system is an aid and that the operation depends heavily on personnel at the controls.*)

The results of the ability to implement knowledge were also very supportive of the thesis goals. The configuration of knowledge in the prototype system was felt to be very straightforward by the mill technical staff.

"Anyone with *any* programming or MOPS configuration experience will find configuring the expert system straightforward. Rules are easy to add and to understand" (Katrinka Scott-Burns, Process Technician, Daishowa-Marubeni Peace River Pulp Mill)

It took only a day of training before they were able to configure knowledge. The implementation of knowledge in the prototype was accomplished by small simple local decoupled rule sets, and by a few attributes. Recognition experience was configured through small pattern matching rule sets. Expectational experience was also configured by small rule sets that specified expectation limits with respect to different process conditions. It was found that expectations could effectively be implemented for various process conditions or modes of operations. Association links were quickly configured as attributes. Although the knowledge base changes were enacted by ourselves and by the technical staff, the operator working on tuning the knowledge base often *read* the configured knowledge, and pointed out changes he would do. If was fairly evident that he could have configured the knowledge directly. This has the implication that knowledge could be transferred directly by operations personnel into the prototype system, if given access to doing so.

Overall, the exercise of acquiring and configuring knowledge for the prototype system showed the system to be very understandable, and to be easy to implement knowledge in. The ease at which the system can be worked with is a reflection of the knowledge representation architecture that it is structured after. The prototype system demonstrates a successfulness of the architecture towards overcoming the difficulties of knowledge base understandabilty, maintainability and expandability that current expert systems exhibit.

5.3 System Trial Results

Almost the moment that the prototype system was fully integrated and configured, it was producing results. The execution of the knowledge representated by the proposed architecture detected unexpected process values and predicted possible process incidents. An operations person was questioned about the detected situations. The reply to the question by Dan Bowes (Operating Technician I, Daishowa Pulp Mill) was, "these are all valid incidents". Appendix G includes printouts from the prototype system's activity log. During the initial startup the expert system was brought up and down several times to make final corrections, thus some of the durations shown on the printouts are short. These logs depict several detected activities.

The activities on February 7, Figure G-1, show *warnings* being issued for a plugged EO washer inlet box. During those times the Elemental Oxygen stage was operating at a very high consistency and the EO washer was in danger of plugging. Why the EO stage was being operated at such a high consistency, was due to the addition of three drying sections on the pulp machine. The mill can now produce more pulp per day. It is now, in fact, producing up to fourteen hundred tons per day. The mill was designed for a capacity of only one thousand tons per day, thus the washers are having to be run at the edge of their capabilities. Higher consistencies are necessary to push the pulp through.

The CD sample line leak, shown in Figure G-2, was one of the incidents that were eventually removed. The sample line is a small parallel pipe to the stock flow where pulp is drawn off. A set of on-line process analyzers sits on this line. These analyzers directly effect the control of bleaching chemical addition. In order to prevent plugging, the sample line has a water flush system. The entry valve for this system has been known to not close completely. The result is that the analyzer measurements are "watered down." It was very difficult to isolate the water leak situation from other possible scenarios based on the measurable process data. A simpler solution of installing a positive flow switch on the line was suggested instead.

On February 8, Figure G-3, a low D1 tower level was detected. The comment by the operator, Dan Bowes, who came in to work with us that day was, "They're looking for trouble." The operations personnel brought the tower level back up some time after 14:25 that day. The mill was running a peak capacity, so again the plugged EO washer inlet box *warning* was detected. A high pH on the CD stage was picked up the expert system. It was being recorded as occurred. The operator was questioned about that. He informed us that there has been a work request issued long ago to calibrate the pH probe in the CD stage. The expectation limits that were supplied for the system were valid in anticipation that the pH probes would eventually be calibrated. I feel that the expert system produces a valid result in case such as this. It brings problem equipment to the attention of the mill management staff.

During the time of these results, work was being done to tune the recognition experience that resets the incident. The limits were not completely configured at this point and were being updated. Thus, the cycling of the incidents that are displayed in the log printouts are now dampened. Unless the indicating process values return to *normal* expected ranges the incident will remain being viewed as active.

The discussed incidents are presently all of the ones that have been detected. This concurs with the mills present operations. There has not been an incident for quite a while. The mill is running exceedingly well at present. This is rewarding for the mill, however, until a further incident occurs results will not be forthcoming.

In summary, the prototype system is performing the function of process monitoring. The system is detecting valid possible incidents and is resetting them. The results are properly being displayed and logged to a disk file. The hypermedia links are functioning to bring the experience information to the operator. Run time messages are being generated. The knowledge representation architecture in the prototype system is suc ssful in its ability to implement a process monitoring system for a pulp bleaching plar.

6
Conclusions

6.1 Thesis Evaluation

This thesis proposes a knowledge representation architecture for expert systems designed to represent all types of knowledge process domain knowledge and to execute process monitoring thinking knowledge. The architecture attempts to better the present state of the technology in the sense that it hopes to reduce or overcome several problems that present applications face. These problems, as well as the objectives for the architecture, were discussed in Section 1.5. A prototype system was developed to test the worthiness of the architecture by applying it to a pulp bleaching process. Evaluation criteria for the prototype system were suggested (Section 1.5.4) to judge the worthiness of the proposed architecture. These criteria identify measurable aspects of the prototype's operation and character which can be directly attributed to the architecture. The degree of success of these aspects indicates the worthiness of the proposed architecture. The following discussion is an evaluation based on the cited criteria.

For the proposed architecture to have worthiness it must be applicable. The first criterion questions this aspect. The results of the prototype installation (Section 5.1) clearly show that it can be applied. The prototype is as an expert system that implements a subset of the architecture. This system is fully integrated into the pulp mill's computer and control systems, and interfaces with real-time software and data. The prototype operates on-line and continuously monitors the process. The system is not producing nuisance results and is, as commented by an operator, friendly to work with.

An expert system built after the proposed architecture for process monitoring must work. The results of chapter five indicate that the prototype system is detecting valid undesirable process situations. The validity of the detected situations has been supported by mill operators. How meaningful the supplied information for a detected situation is has not been fully tested due to the good operating conditions of the Mill. It is felt by operators that the supplied information will be useful, and this was commented on in Chapter 5.

For an application of process monitoring, an expert system needs to be able to predict oncoming occurrences of an incident before production is interrupted. The prototype system definitely achieved this as shown by the warnings. These warnings were verified by mill operators to be potentially bad situations.

An expert system also needs to be able to adjust for the inconsistent nature of a process, if it is to be applied to process monitoring. The proposed architecture was created to be able to represent the inconsistent nature of a world. This representation was utilized in configuring the process monitoring knowledge in the expert system. However, few inconsistencies were implemented in the limited scope. Full testing of this criteria could acheived.

The aim of the architecture is to represent and apply episodic experience. The prototype system was configured using mill incident reports by directly installing the knowledge of these situations. This episodic knowledge is being presented to the operator when a similar situation is occurring.

The success of the architecture to represent knowledge in a manner similar to a human is indicated by the ease of configuring the prototype. Section 5.2 contains several comments that indicated that the system was understandable and easily configurable. This section also expresses that the acquisition and transfer of knowledge were straightforward. Knowledge for the prototype system could be related freely by the operations personnel.

The architecture was aimed at being able to replicate the thinking behaviour of a human. This was achieved in the prototype. The knowledge of how an operator monitors the pulp bleaching process was ascertained and then represented in the prototype. The response from both technical and operations personnel of how the system went about monitoring the process was extremely complementary. They commented on the common sense of the approach (Section 5.2).

The results of the case study, and the comparison of them with the evaluation criteria, demonstrate that the architecture is a bettering expert system design for the purpose of process monitoring. The objectives of reducing the difficulty of being able to fully represent a process domain, and of the difficulty expressing knowledge in the form that is required by an expert system have been accomplished in theory and in practice. From the success of the prototype and from the discussions in appendix A on potential cost savings, it can be concluded that this technology offers strong benefits to pulp bleaching process monitoring. The benefits can be seen as both cost savings and customer satisfaction by a pulp mill. It is felt, therefore, that applying an expert system which is designed with the proposed architecture to the monitoring of a pulp bleaching, or any other, process, is a practical, worthy, and a cost reducing venture.

6.2 Summary

A knowledge representation architecture has been conceived which addresses various problems and deficiencies of present expert system technology in the activities involving the monitoring of industrial processes. This architecture meets the intended objectives laid out in Section 1.5.2. A summary of the key results of the architecture is as follows:

1) The architecture aptly represents knowledge in the form that humans consciously perceive it by:

   - representing all domain entities, physical and abstract, as *objects.*

   - representing all concepts and entities employed by the human mind for the purpose of intelligently interacting with a domain, such as thought processes and episodic experiences, as *objects.*

   - localizing all knowledge about an entity within the representing object.

   - representing a world not only as a composite of entities, but also as numerous systems of which the entities are part of. These systems can be abstract, conceptual, or physical in nature. Knowledge of how the entities interact in a system is encapsulated in each entity's object representation.

   - representing the associative linking between perceived concepts within the human mind. These links form a human like, and efficient method, to recall conceptual entities and episodic experience knowledge which pertain to the present thinking context. The associative links map complex, context dependent, routing for thinking to follow when dealing with a problem or a situation.

2) The architecture provides a representation to captures and execute intelligent thinking and thought processing abilities of a human by:

   - decoupling thinking knowledge (knowledge of how to think about something) from information knowledge (knowledge about the domain).

- representing thinking as an intelligent process of applying thought activities that can act on generic information.

- capturing a library of thought processes which can be summoned and applied to confronted situations. Any thought process can summon other thought processes to fulfill a manifested requirement. A thought process can be either derived theoretically, or be developed and refined through experience.

- acknowledging and representing thought processes for the purpose of determining the manner in which to perform thinking, as well as thought processes that apply the thinking method to knowledge about a domain.

3) The architecture is capable of representing experience in a close to natural form by:

   - representing experience in distinct forms in which they are utilized by humans. (Experience is seen as more than just domain knowledge that has been refined by personal experience. It is also seen as being *directly* applicable knowledge. This knowledge is applied to provide knowledge of situations, expectations of a domain, and how to recognized something.)

   - understanding that recognition experience provides a predictive trait for humans, and employing it to this end.

4) The architecture creates a facilitated knowledge development environment by:

   - representing all knowledge homogeneously as objects.

   - represents knowledge in the form that humans consciously perceive it.

   - localizing and decoupling all information into self encapsulating entities.

- representing a domain to any degree of granularity and complexity without impeding the ability of the system to perform intelligent activities. The only effect is the depth and diversity of solutions in which the system can ascertain from the implemented knowledge.

- providing a practical means for a user to incorporate the nonmonotonic nature of a domain.

- embodying intelligence into an object so that the nature of an object can be intelligently expressed in relation to a domain context. Embodying intelligence also provides a means for capturing and representing generic thought behaviours.

It is felt the architecture addresses the problems detailed in Section 1.5.1 to provide an:

- Ability to accurately represent all the necessary aspects of a domain.

- Ability to handle novel situations.

- Ability to easily implement knowledge.

- Ability to comprehend and understand the knowledge base after the fact.

This architecture was implemented in a prototype system that was applied to a practical industrial situation. The system was evaluated by criteria tto meter the success of the architecture (see Sections 1.5.4 and 6.1). It is concluded from the evaluation, and from the intuitive completeness of the architecture's concepts, that this proposed knowledge representation architecture offers a better and more capable means to represent a process monitoring domain and process monitoring intelligent activities. Further, it is also concluded that this technology, if applied for the purpose of process monitoring of a pulp bleaching process, is very beneficial and cost saving for the operation of this process.

6.3 Future Improvements

This architecture offers much promise as a platform for the research and development of advance technologies such as learning, simulated thinking, optimal control, parallel processing of intelligence, etc. However, on a more immediate level some future concerns are:

- Developing a scheme utilizing intelligence to determine a priority for a goal, that takes into account the present temporal state of a situation, the ability to do something about the situation, the ease at which it can be done, and the impact if action is not taken.

- Developing an intelligent confidence determination scheme for recognition experience which is embedded in situational experience knowledge. When recognition experience is applied, it should generate a *feeling of confidence* with its conclusions.

- Develop the concept of a vocabulary and of a sentence structure. This is especially crucial because of the intelligence embodied in an object. As discussed in Section 2.3.1., an object becomes an intelligent entity. Hence, as humans share information, and issue commands and requests among each other by a language, the same must be developed for objects. A domain must have an established vocabulary so that all objects representing that domain can understand what is being communicated to them. A sentence structure is necessary if a totally generic architecture is to be developed for a domain. The structure does not have to be complex; even a structure of five words could suffice for a process domain. More complex domain representations would need more complex syntactic structures. The generic nature becomes developed by the ability of an object to formulate commands and requests to other objects, and to understand non explicit replies to inquiries, in the same manner as does a human.

- For the prototype system, a future step would be to develop more thinking mechanisms, such as mulit-value logic or evidential reasoning, and intelligent tools, such as statistical analysis that can be applied by thought processes.

The next step for the architecture should be to form a fu, ~~~~ project. This project should be designed with the intent to develop and test a representation of a process monitoring domain, completely using the all the concepts of proposed architecture. Such a project should challenge the advanced thinking behaviour that is proposed by the architecture, and a detailed representation of a variety of systems that make up a domain.

References

Alaman, X.; Romero, C.; Aguirre, P.; Serrahima, R. M.; Lopez,. V.; Dorronsoro, J.; de Pablo, E. (1992). "Knowledge-based systems for real-time process control: the MIP project", *Artificial Intelligence in Real-Time Control, IFAC/IFIP/IMACS Symposium*, Delft, Netherlands, 391-396.Anderson, J. R. (1976). *Language, Memory and Thought*, Erlbaum, Hillsdale, NJ.

Albright & Wilson, (1988). *Erco Bleaching Workbook*, Albright & Wilson Americas, Pulp Chemicals group, Islingnton, Ontario, Canada.

Anderson, J. R. (1983). *The Architecture of Cognition*, Harvard University Press, Cambridge, MA.

Anderson, J. R. (1987). "Skill acquisition: compilation of weak-method problem solutions", *Psychology Review*, **94**, 192-210.

Arzen, K. E. (1992). *"A survey of commercial real-time expert system environments"*, IFAC Artificial Intelligence in Real-Time Control, Delft, 483-490.

Atkinson R.L.; Atkinson R.C.; Hilgard E.R. (1983). *Introduction to Psychology*, Harcourt Brace Jananovich Inc., USA

Bandreddi, E. P.; Perraju, T. S.; Uma, G.; Umarani, P. (1994). "An expert system shell for aerospace applications", *IEEE Expert*, August, 56-64.

Bellezza, F. S. (1989). "Mnemonics and expert knowledge: mental cuing", *The Psychology of Expertise*, **R. R. Hoffman, ed.**, Spinger-Verlag, New York, NY, 1992.

Bellezza, F. S.; Buck D. K. (1988). "Expert knowledge as mnemonic cues", *Applied Cognitive Psychology*, **2**, 147-162.

Betlem, B. H. L.; Aggele, R. M. V. (1994). "An object-oriented framework for production control", *Control '94*, March, 1411-1416.

Bonissone, P. P.;Ayub, S. (1992). "Similarity measures for case-based reasoning systems", *Advanced Methods in Artificial Intelligence. 4th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems,* **IMPU '92**, 161-172.

Boyd B. (1972). *Thinking About Inquiry*, Mcgraw-Hill Ryerson Limited, Toronto, Canada.

Brunet, J.; Cauvet, C.; Meddahi, D.; Scmmak, F. (1993). "Object-oriented analysis in practise", *Caise 93 Advanced Information Systems Engineering 5th International Conference*, Paris, France, June, 293-308.

Buchanan, B. G. & Smith, R. G. (1989). "Fundementals of expert systems", *The Handbook of Artificial Intelligence*, **A. Barr, P. R. Cohen, E. A. Feigenbaum, eds.**, Addison-Wesley, Reading, MA, 149-192.

Boullart, L. (1988). "Artificial Intelligence and expert systems: next generation tools", *Industrial Process Control Systems Reliability, Availability and Mantainability. Proceedings of the IFAC Workshop*, Bruges, Belgium, 45-52.

Cauvin, S.; Braunschweig, P G.; Glaize, Y. (1992) "Model-based diagnosis for continuous process supervision: the ALEXIP experience", *IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in REal Time Control*, 515-521

Cercone, N.; McCalla, G. (1987). "What is knowledge representation", *The Knowledge Frontier: Essays in the Representation of Knowledge.*, Springer-Verlag, New York, NY.

Chandrasekaran, B. (1988). "Generic tasks as building blocks for knowledge-based systems", *The Knowledge Engineering Review*, **3**, 183-210.

Charness, N.; Campbell, J. I. D. (1988). ""Acquring skill at mental calculations in adulthood: a task decomposition" *Journal of Experiemental Psychology: General*, **117**, 115-129.

Chase, W. G.; Simon, H. A. (1973). "The mind's eye in chess", *Visual Information Processing*, **W. G. Chase, ed.**, Academic Press, New York, NY.

Chi, M. T. H.; Feltovich, P.; Glasser, R. (1981). "Categorization and representation of physics problems by experts and novices", *Cognitive Science*, 5, 121-152.

Chi, R. T.; Whinston, A. B.; Kiang, M. Y. (1993). "Case based reasoning to model building", *Proceedings of the Twenty-Sixth Hawaii International Confernece on System Sciences*, **3**, 324-332.

Clements, B. R.; Preto, F. (1993). "Evaluating commercial real time expert system software for use in the process industries", *Expert Systems for the Process Industries Symposium 43rd Canadian Chemical Engineering Conference*, Oct, 107-114.

Courdier, R.; Herin-Aime, D. (1989). "Object oriented knowledge and reasoning designed for diagnostics", *IFAC Advanced Information Processing in Automatic Control*, Nancy, France, July, 251-255.

Cooke, N. J. (1989). "Modeling human expertise in expert systems", *The Psychology of Expertise*, **R. R. Hoffman, ed.**, Spinger-Verlag, New York, NY, 1992.

CPPA (1989). *A Manual of Kraft Mill Operating Practices*, Technical Section Canadian Pulp and Papare Association, **Martin Mcleod, ed.**, Canada

Dimitrov, I. I. (1992). "Systems-based knowledge representation: relations and methods", *Artificial Intelligence V: Methodology, Systems, Applications*, 203-212.

Du Boulay, B. (1989). "Nonadversary problem solving by machine", *Human and Machine Problem Solving*, **K. J. Gilhooly, ed.**, Plenum Press, New York, NY

Duncker, K. (1945). "On problem solving", *Psychological Monographs*, **58**, **(5)**, (whole No. 270).

David, J-M.; Krivine, J-P. (1987). "Three artificial intelligence issues in fault diagnosis: declarative programming, expert systems, and model-based reasoning", *Proceedings of Second European Workshop on Fault Diagnosis, Reliability, and Knowledge-Based Approaches*, **Apr**, 19-27.

Entsminger, G. (1990). *The Tao of Objects*, M&T Books, Redwood city, CA.

Erlenbach, S.; Ying, Y.; Rao, M.; Danielson, K. (1994a). "An expert system for on-line incident detecting and reporting", *1994 CPPA Western Branch Fall Conference*, Grande Prairie, Canada, Oct. 25-26.

Erlenbach, S.; Ying, Y.; Rao, M.; Danielson, K (1994b). "An intelligent system for bleach plant incident warning and reporting", *44th Canadian Chemical Engineering Conference*, Calgary, Canada, **Oct**, 1-3.

Fiegenbaum, E. A.; Feldman, J. (1963). *Computers and Thought*, McGraw-Hill, New York, NY.

Fitts, P. M., Posner, M. I. (1967). *Human performance*, Brooks and Cole, Belmont, CA.

Fox, M. S. (1990). "AI and expert system myths, legends, and facts", *IEEE Expert*, Feb, 8-20.

Frost, R. A. (1987). *Introduction to Knowledge Base Systems*, Collins Professional and Technical Books, London, England.

Gamble, K. (1993). "NOxSMART Real time expert system for process and polulution control", *Expert Systems For The Process Industries Symposium, 43rd Canadian Chemical Engineering Conference*, Oct, 39-43.

Gang, R.; Wang, S-Q.; Wang, J-C. (1988) "Building a knowledge base of a fault diagnostic system for chemical processes", *IFAC Industrial Process Control Systems*, Bruges, Belgium, 67-73.

Genesereth, M. R.; Nilsson, N. J. (1987). "Logical Foundations of Artificial Intelligence", Morgan Kaufmann, Palo Alto, CA.

Gordon, S. E. (1989). "Implications of cognitive theory for knowledge acquistion", *The Psychology of Expertise*, **R. R. Hoffman, ed.**, Spinger-Verlag, New York, NY, 1992.

Hardeveld, T. V. (1992). "Smart monitoring for compressor stations", *International Gas, Turbine and Areospace Congress and Exposition*, Cologne, Germany, June, 1-8

Hendrix, G. G. (1979) "Encoding knowledge in partitioned networks", *Associative Networks*, Nicholas V. Findler, editor, Acedemic Press, New York, 51-92.

Henke, A. (1994). "Scheduling space shuttle missions", *AI Expert*, Mar, 16-24.

Ingrand, F. F.; Georgeff, M. P.; Rao, A. S. (1992). "An architecture for real-time reasoning and system control", *IEEE Expert*, 7, (6), 34-44.

Ishida, Y. (1989). "A framework for dynamic representation of knowledge: a minimum principle in organizing knowledge representation", *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 170-179.

Jackson. P. C. (1985). *Introduction to Artificial Intelligence*, General Publishing Co., Toronto, Canada.

Katona, G. (1940). *Organizing and Memorizing*, Columbia University Press, New York, NY.

Ketler, K. (1993). "Case-based reasoning: an introduction", *Expert Systems With Applications*, 6, 3-8.

Kidd, A. L. (1987). *Knowledge acqu on for expert systems*, Plenum, New York, NY.

Kolodner, J. L. (1989). "Selecting the best case for a case-based reasoner", *Program of the Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, MI., 155-162.

Kolodner, J. L.; Riesbeck, C.K. (1986). "Experience, Memory, and Reasoning", Lawrence Erlbaum Associates,Hillsdale, New Jersey, 1-13.

Kohonen, T. (1989). *Self-Organization and Associative Memory*, Spinger-Verlag, New York, NY.

Kramer, M. A. (1987). "Malfunction diagnosis using quantitative model with non-boolean reasoning in expert systems", *AIChE Journal*, **33**, **(1)**, 130-140.

Krijgsmann, A. J.; Jager, R. (1992). "DICE: a real-time toolbox", *IFAC Artificial Intelligence in Real-Time Control*, Delft, 503-507.

Laffey, T. J.; Cox, P. A.; Schmidt, J. L.; Kao, S. M.; Read, J. Y. (1988). "*Real-time knowledge based systems*", AI Magazine, Spring, 27-88.

Lee, I. B.; Kim, M.; Jung, J.; Chang, K. S. (1992). "Rule-based expert system for diagnosis of energy distribution in steel plant", *IFAC On-Line Fault Detection and Supervision in the Chemical Process Industries*, April, 239-243.

Liard, J. E., (1984). "Universal subgoaling", Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University.

Liard, J. E.; Newell, A. (1983). *A universal weak method*, Carnegie-Mellon University, Computer Science Dept., Pittsburg, PA.

Liard, J. E.; Newell, A.; Rosenbloom, P.S. (1987). "SOAR: an architecture for general intelligence", *Artificial Intelligence*, **33**, 1-64.

Liard, J. E.; Rosenbloom, P.S., Newell, A. (1984) "Towards chunking as a general learning mechanism", *Proceedings AAAI*, Austin ,Texas, 188-192.

Liard, J.; Rosenbloom, P.; Newell, A. (1986). *Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies*, Kluwer Academic Publishers, Hingham, Ma.

Lim, E-P.; Cherkassky, V. (1992). "Semantic networks and associative databases", *IEEE Expert*, 7, **(4)**, 31-40.

Maher, M. L.; Zhang D. M. (1991). "CADSYN: using case and decomposition knowledge for design synthesis", *Artificial Intelligence in Design*, 137-150.

Martin, J., Odell, J. (1992). *Object-Oriented Analysis & Design*, Prentice-Hall Inc., Englewood Cliffs, NJ.

Mayer, R. E. (1989). "Human nonadversary problem solving", *Human and Machine Problem Solving*, **K. J. Gilhooly, ed.**, Plenum Press, New York, NY

Milne R. (1987). "Strategies for Diagnosis", *IEEE Ransactions On Systems, Man, and Cybernetics*, **SMC-17**, **(3)**, 333-339.

Minsky, M. A. (1975). "A framework for repesenting knowledge", *Psychology of Computer Vision*, New York, McGraw Hill

Newell, A., Simon, H.A. (1972). *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ.

Newell, A. (1980). "Reasoning, problem solving and decision processes: The problem space as a fundemental category", *Attention and Performance VIII*, **R. Nickerson, ed.**, Erlbaum, Hillsdale, NJ.

Nii, H. P. (1986a). "Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures", *The AI Magazine*, Summer, 38-53.

Nii, H. P. (1986b). "Blackboard systems: blackboard application systems, blackboard systems from a knolwedge engineering perspective", *The AI Magazine*, August, 82-106.

Oxford American Dictionary (1980), Avon Books, New York

Patil, R. S.; Szolovitz, P.; Schwartz, W. B. (1984). "Causal understanding of patient illness in medical diagnosis", *Readings in Medical Artificial Intelligence: The First Decade*, Addison-Wesley, Reading, MA.

Pennings, R.; Saussais, Y. (1993). "Formentor: Real-time safety oriented decision support using a goal tree-success tree model", *Thirteenth International Conference: Artificial Intelligence, Expert Systems, Natural Language Proceedings*, **2**, 91-100.

Pople, H. E.; Spangler, W. E.; Pople, M. T. (1994). "EAGOL: an artificial intelligence system for process monitoring, situation assessment and response planning", *Proceedings of the 10th Conference on Artificial Intelligence for Applications*, 298-304.

Prerau, D. S.; Alder, M. R.; Gunderson, A. S. (1989). "Elicting and using experiental knowledge and general expertise", *The Psychology of Expertise*, **R. R. Hoffman, ed.**, Spinger-Verlag, New York, NY, 1992.

Reeve, D. W. (1989) *Pulp and Paper Manufacture: Volume 5: Alkaline Pulping*, **T. M. Grace & E. W. Malcolm, ed.**, CPPA, Montreal, Canada.

Regoczei, S. B.; Hirst, G. (1989). "Knowledge and knowledge acquisition in the computational context", *The Psychology of Expertise*, **R. R. Hoffman, ed.**, Spinger-Verlag, New York, NY, 1992.

Reimann, P.; Chi, M. (1989). "Human expertise", *Human and Machine Problem Solving*, **K. J. Gilhooly, ed.**, Plenum Press, New York, NY

Rivera, D. E.; Desai, A.; Beaumariage, T.; Roberts, C. (1993). "A knowledge-based framework for controller structure selection in intelligent process control.", *Proceedings of the 1993 American Control Conference*, **(2)**, 1890-1894.

Rosenbloom, P. S.;, Newell, A. (1983). "The chunking of goal hierarchies: a generalized model of practice", *Proceedings of the 1983 Machine Learning Workshop*.

Rumelhart, D. E. (1975). "Notes on a schema for stories", *Representation and Understanding*, **D. G. Borbrow & A. M. Collins, ed.,** Academic Press, New York, NY.

Rumelhart, D. E.; Norman, D. A. (1983). *Representation in Memory*, San Diego:University of California, Cognitive Science Lab., CHIP.

Ryle, G. (1949). *The Concept of Mind*, Hutchinson, San Francisco, CA.

Sassen, J. M. A.; Jasper, R. B. M. (1992). "Designing real-time knowledge based systems with PERFECT", *IFAC Artificial Intelligence in Real-Time Control*, Delft, 491-496.

Schank, R. C.; Slade, S. B. (1991). "The future of artificail intelligence: learning from experience", *Applied Artificial Intelligence*, **5**, 97-107

Sembugamoorthy, V.; Chandrasekaran, B. (1986). "Functional representation of devices and compilations of diagnostic problem-solving systems", *Experience, Memory, and Reasoning*, **J. L. Kolodner & C. K. Riesbeck, ed.**, Lawrence Erlbaum Associates,Hillsdale, New Jersey, 47-73

Shiozaki, J.; Matsuyama, H.; O'shima, E.; Iri, M. (1985). "An improved algortihm for diagnosis of system failures in the chemical process", *Computers and Chemical Engineering*, **9**, **(3)**, 285-293.

Smook, G. A. (1990). *Handbook for Pulp and Paper Technologists*, Angus Wilde Publications, Vancouver, Canada

Sorabji, R. (1972). *Aristotle on Memory*, Brown University Press, Providence, Rhode Island.

Stein, L. (1992). "Integration of expert system technology within the process monitoring and analysis system", *Electronic Progress*, **32**, **(1)**, 9-12.

Terpstra, V. J.; Verbruggen, H. B.; Hoogland, M. W.; Ficke, R. A. E. (1992). "A real-time, fuzzy, deep-knowledge based fault-diagnosis system for a CSTR", IFAC *On-Line Fault Detection and Supervision in the Chemical Process Industries*, April, 26-31.

Thomson, A. C. (1988) "Real-time artificial intellegence for process monitoring and control", *Artificial Intelligence in Real-Time Control*, Swansea, U.K.

Vossos, G.; Dillon, T.; Zeleznikow, J.; Graeme, T. (1991). "The use of object oriented principles to develop intelligent legal reasoning systesm", *The Australian Computer Journal*, **23**, **(1)**, 2-10.

Wagenaar, G.; Schrijnen, L. M. (1988). "Autopes, the developemtn of a knowledge based system for process control", *Industrial Process Control Systems: Reliability, Availability and Maintainability, Proceedings of the IFAC Workshop*, Bruges, Belgium, 59- 65.

Wertheimer, M. (1959). *Productive Thinking*, Harper & Row, New York, NY.

Wilikens, M.; Nordvik, J.P.; Poucet, A. (1993). "FORMENTOR: a real-time expert system for risk prevention in complex hazardous environments: a case study", *Control Engineering Practice*, **1**, **(2)**, 323-328.

Williams, D. J.; West, A. A.; Hinde, C. J. (1991). "AI for process control and monitoring", *ICCIM 91 Proceedings of the International Conference on Computer Integrated Manufacturing*, 579-582.

Wood, W.A. (1983). "What's important about it knowledge representation", *Computer*, **16**, **(10)**, October.

Yao, Y (1994). "A petri-net model for temporal knowledge representation and reasoning", *IEEE Transactions on Systems, Man, and Cybernetics*, **24**, **(9)**, 1374-1382.

Appendix A

Benefits of Process Monitoring System

Scenario #1 (Equipment failure - early warning)

The expert system will allow operators to respond to early warning alarms quickly and efficiently. The system not only informs operators of impending problems with the process, but also instructs them on what to do before the problem becomes irreversible and causes a production upset. For example, take a high Inlet Box alarm on one of the four C.B. Bleach Washers. If the operator does not react quickly or follow the instructions on how to remedy the incident, a production interruption will occur. Take into account that pulp prices are at $750.00 per ton, that the mill produces one ton of pulp per minute, and that a downtime or production interruption duration can be thirty to fourty-five minutes until the system is back up to full production again. This would calculate to between $22,500.00 to $33,750.00 in production losses.

It is important for our operators to maintain a uniform and uninterrupted operation. The expert system will definitely give our operators an informed advantage and the skill to advert costly incidents and downtime.

Dan Bowes,

Operating Technician I,

Daishowa-Marubeni Peace River Pulp Mill

Scenario #2 (Process variable off target - early warning)

The expert system will allow operators to respond to early warning alarms quickly and efficiently. The system not only informs operators of impending problems with the process, but also instructs them on what to do before the problem becomes irreversible and causes a production upset. For example, take a low EO Washer ph alarm. If the operator does not react or follow the instructions on how to remedy the incident, an off-grade problem will occur in the final product. The mill's market is generated towards the sale of high quality product to its customers. Off-grade pulp can sit in our warehouse for extended periods of time until a customer can be found. If demand is low for an off-grade product, the price is considerably lower than that of an on-grade product. Take into account that pulp prices are at $750.00 per ton, that the mill produces one ton of pulp per minute, and the off-grade pulp can be devalued by $50.00 t0 $200.00 per ton depending on the extent of the defect. One hour of low EO Washer ph can expand into two hours of off-grade due to mixing in the latter stages. This would calculate to between $6,000.00 to $24,000.00 in sales losses. This production loss also puts a strain on ensuring that we do get the on-grade pulp to our customers on time.

It is important for our operators to maintain a uniform and consistent operation. The expert system will definitely give our operators an informed advantage and skills to advert costly incidents.

Dan Bowes,

Operating Technician I,

Daishowa-Marubeni Peace River Pulp Mill

Appendix B

Example Configuration of a Process Object

```
/*************** CL2 FLOW TO MIXER OBJECT ***************/
/* Definition of CL2 Flow to Mixer Process object */

Unit: cl2mix_obj in DMI.kbs;
  Superclasses: Process_obj;

Memberslot: Value From cl2mix_obj;
  Inheritance: Override;
  Valueclass: String;
  Values: "fc104341";
End Slot;

Memberslot: case_1 From cl2mix_obj;
  Inheritance: Override;
  Valueclass: String;
  Values: "R930757";
End Slot;

Memberslot: case_2 From cl2mix_obj;
  Inheritance: Override;
  Valueclass: String;
  Values: "R931573";
End Slot;

/* Variable definition section */
Memberslot: vset From cl2mix_obj;
  Inheritance: Override;
  Valueclass: Rules;
  Values: { }
End Slot;
```

```
/* Define rules to set operating range variables */
Memberslot: rset From cl2mix_obj;
  Inheritance: Override;
  Valueclass: Rules;
  Values: {
   RULE 1     fact   xgrad=1
              and    _grade="hw"
              and    n1042b41<50
              then   _frame.off=0.0;
                     _frame.llim=300;
                     _frame.lnorm=400;
                     _frame.hnorm=1000;
                     _frame.hlim=1100;

   RULE 2     fact   xgrad=1
              and    _grade="hw"
              and    n1042b41>=50
              and    n1042b41<90
              then   _frame.off=0.0;
                     _frame.llim=200;
                     _frame.lnorm=250;
                     _frame.hnorm=1000;
                     _frame.hlim=1100;

   RULE 3     fact   xgrad=1
              and    _grade="hw"
              and    n1042b41>90
              then   _frame.off=0.0;
                     _frame.llim=0;
                     _frame.lnorm=0;
                     _frame.hnorm=0;
                     _frame.hlim=0;

   RULE 4     fact   xgrad=1
              and    _grade="sw"
              and    n1042b41<50
              then   _frame.off=0.0;
                     _frame.llim=500;
                     _frame.lnorm=650;
                     _frame.hnorm=1600;
                     _frame.hlim=1750;
   RULE 5       fact   xgrad=1
                and    _grade="sw"
                and    n1042b41>=50
                and    n1042b41<90
                then   _frame.off=0.0;
                       _frame.llim=350;
                       _frame.lnorm=400;
                       _frame.hnorm=1500;
                       _frame.hlim=1700;
   RU    6      fact   xgrad=1
                and    _grade="sw"
                and    n1042b41>90
                then   _frame.off=0.0;
                       _frame.llim=0;
                       _frame.lnorm=0;
                       _frame.hnorm=0;
                       _frame.hlim=0;
          }
  End Slot;

/* define rules to set operating status varialbes */
Memberslot: sset From cl2mix_obj;
  Inheritance: Override;
  Valueclass: Rules;
  Values: {
         }
End Slot;
End Unit;
```

Appendix C

Example Configuration of a Experience Object

```
/*************** E9402 OBJECT(11)***************/
/* Definition of CD WASHER INLET BOX object */

Unit: E9402_obj in DMI.kbs;
  Superclasses: Incident_obj;

/*******************************/
/* Variable definition section */

Memberslot: title From E9402_obj;
   Inheritance: Override;
   Valueclass: String;
   Values: "PLUGGED CD WASHER INLET BOX";
End Slot;

Memberslot: prior From E9402_obj;
   Inheritance: Override;
   Valueclass: String;
   Values: "  ";
End Slot;

Memberslot: disp_epn_1 From E9402_obj;
   Inheritance: Override;
   Valueclass: String;
   Values: "fc107241";
End Slot;

Memberslot: disp_epn_2 From E9402_obj;
   Inheritance: Override;
   Valueclass: String;
   Values: "t1065b41";
End Slot;
```

```
Memberslot: disp_epn_3 From E9402_obj;
  Inheritance: Override;
  Valueclass: String;
  Values: "nc100641";
End Slot;

/****************/
/* Description variables */

Memberslot: grade From E9402_obj;
  Inheritance: Override;
  Valueclass: String;
  Values: "both";
End Slot;

Memberslot: mode From E9402_obj;
 Inheritance: Override;
  Valueclass: String;
  Values: "prod";
End Slot;

Memberslot: cond_1 From E9402_obj;
  Inheritance: Override;
  Valueclass: String;
  Values: "all";
End Slot;

Memberslot: qual_1 From E9402_obj;
  Inheritance: Override;
  Valueclass: String;
  Values: "any";
End Slot;


/****************/
/* Rules (logic) definition section */

Memberslot: vrset From E9402_obj;
  Inheritance: Override;
  Valueclass: Rules;
  Values: { }
End Slot;
```

```
Memberslot: check From E9402_obj;
  Inheritance: Override;
  Valueclass: Rules;
  Values: {
  Rule 1      fact   1=1
               then   send("stat") to "cdbox";   /* 41FC1072/41FC1103 */
                      send("stat") to "cdn01";   /* CD consistency*/

  Rule 2      fact   cdbox.range="in"
               and    cdbox.level="low"
               and    cdbox.extent="very"
               then   _frame.stat="act";
                      _frame.cond="WARN";

  Rule 3      fact   t1065b41>90
               then   _frame.stat="act";
                      _frame.cond="WARN";

  Rule 4      fact   cdbox.range="out"
               and    cdbox.level="low"
               and    t1065b41>90
               and    cdn01.range="out"
               and    cdn01.level="high"
               then   _frame.stat="act";
                      _frame.cond="OCC";
        }
End Slot;

Memberslot: reset From E9402_obj;
  Inheritance: Override;
  Valueclass: Rules;
  Values: {
  Rule 5      fact   cdbox.level="norm"
               or     cdbox.level="high"
               and    t1065b41<85
               then   _frame.stat="inact";
        }
End Slot;
End Unit;
```

Appendix D

Class Object Configuration

```
/*****************************************************************************/
/*                                                                           */
/*      FILE  : CLASS.FRA                                                    */
/*      AUTHOR      : Sam Erlenbach                                          */
/*      DATE : 27/6/94                                                       */
/*                                                                           */
/*****************************************************************************/
/*      DESCRIPTION:                                                         */
/*        This file contains the default class objects that                 */
/*      contain the base structures for each objects. These                  */
/*      class objects are inherited by each individual class                 */
/*      object.                                                              */
/*                                                                           */
/*****************************************************************************/
/* Global definition section */
GLOBE
  xgrad: integer;
  xmode: integer;
END
/*******************************/


/******************** PROCESS OBJECT ********************/
/* The following is the root Process object */

Unit: Process_obj in DMI.kbs;

/*******************************/
/* Variable definition section */

Memberslot: value from Process_obj;
  Valueclass: String;
  Values: 0.0;
End Slot;

/****************/
```

```
/* Run code variables */

Memberslot: src from Process_obj;
  Valueclass: Integer;
  Values: 0;
End Slot;

Memberslot: rrc from Process_obj;
  Valueclass: Integer;
  Values: 0;
End Slot;

/*****************/
/* Status variables */

/* Possible values are: ok/bad/pot */
Memberslot: stat from Process_obj;
  Valueclass: String;
  Values: " ";
End Slot;

/* Possible values are: yes/no */
Memberslot: ustat from Process_obj;
  Valueclass: String;
  Values: "yes";
End Slot;

/* Possible values are: in/out */
Memberslot: range from Process_obj;
  Valueclass: String;
  Values: " ";
End Slot;

/* Possible values are: norm/high/low */
Memberslot: level from Process_obj;
  Valueclass: String;
  Values: " ";
End Slot;

/* Possible values are: off/mod/very/peak */
Memberslot: extent from Process_obj;
  Valueclass: String;
  Values: " ";
End Slot;
/*****************/
```

```
/* Range variables */

Memberslot: off from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;

Memberslot: lextm from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;

Memberslot: llim from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;

Memberslot: vlow from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;

Memberslot: lnorm from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;

Memberslot: hnorm from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;

Memberslot: vhigh from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;

Memberslot: hlim from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;
```

```
Memberslot: hextm from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;

Memberslot: peak from Process_obj;
  Valueclass: Real;
  Values: " ";
End Slot;

/*****************/
/* Range check variables */

/* Possible values are: hw/sw */
Memberslot: pgrad from Process_obj;
  Valueclass: String;
  Values: "init";
End Slot;

/* Possible values are: start/stop/prod */
Memberslot: pmode from Process_obj;
  Valueclass: String;
  Values: "init";
End Slot;

/********************************/
/* Method definition */

/* Method to check the status of the Process object */
Memberslot: pm1 from Process_obj;
  Valueclass: METHODS;
  Values: pm1;
End Slot;

/* Method to set the operating ranges of the data point */
Memberslot: pm2 from Process_obj;
  Valueclass: METHODS;
  Values: pm2;
End Slot;
End Unit;
```

```
/******************** INCIDENT OBJECT ********************/
/* The following is the root Incident object */

Unit: Incident_obj in DMI.kbs;

/****************/
/* Run code variables */

Memberslot: src from Incident_obj;
  Valueclass: Integer;
  Values: 0;
End Slot;

Memberslot: rrc from Incident_obj;
  Valueclass: Integer;
  Values: 0;
End Slot;

/****************/
/* Status variables */

/* Possible values are: act/inact */
Memberslot: stat from Incident_obj;
  Valueclass: string;
  Values: " ";
End Slot;

/* Title of incident to be sent to Hypermedia */
Memberslot: title from Incident_obj;
  Valueclass: String;
  Values: " ";
End Slot;

/* Possible values are: warn/imm/occ */
Memberslot: cond from Incident_obj;
  Valueclass: String;
  Values: " ";
End Slot;

/****************/
/* Description variables */

/* Possible values are: hw/sw */
Memberslot: grade from Incident_obj;
  Valueclass: Integer;
```

```
   Values: " ";
 End Slot;

 /* Possible values are: start/stop/prod */
 Memberslot: mode from Incident_obj;
   Valueclass: Integer;
   Values: " ";
 End Slot;

 /****************/
 /* Range check variables */

 /* Possible values are: hw/sw */
 Memberslot: pgrad from Incident_obj;
   Valueclass: String;
   Values: "init";
 End Slot;

 /* Possible values are: start/stop/prod */
 Memberslot: pmode from Incident_obj;
   Valueclass: String;
   Values: "init";
 End Slot;

 /*******************************/
 /* Method definition */

 /* Method to check the status of the Incident object */
 Memberslot: im1 from Incident_obj;
   Valueclass: METHODS;
   Values: im1;
 End Slot;

 /* Method to reset the Incident object's status */
 Memberslot: im2 from Incident_obj;
   Valueclass: METHODS;
   Values: im2;
 End Slot;

 /* Method to set the ranges of the Incident object's variables */
 Memberslot: im3 from Incident_obj;
   Valueclass: METHODS;
   Values: im3;
 End Slot;
 End Unit;
```

```
/****************** METHODS FOR CLASS *****************/
/* The following are the Methods for the Thinking file */

/***** METHOD PM1 *****/
/* Method to check the status of the Process object */
METHOD pm1(stat:keyword)
VAR
  stat: keyword;

BEGIN
  if _frame.src<>_rc then
    Begin
     send("range") to _frame;
     reason(_frame,"vset");
     status(_frame);
     reason(_frame,"sset");
     _frame.src=_rc;
  End;
END.

/***** METHOD PM2 *****/
/* Method to set the operating ranges of the data point */
METHOD pm2(range:keyword)
VAR
  range: keyword;

BEGIN
  if _frame.rrc<>_rc then
    Begin
     xgrad=0;
     xmode=0;
     if _frame.pgrad<>_grade then
       Begin
        xgrad=1;
        _frame.pgrad=_grade;
       End;
     if _frame.pmode<>_mode then
       Begin
        xmode=1;
        _frame.pmode=_mode;
       End;
     reason(_frame,"rset");
     _frame.rrc=_rc;
  End;
END.
```

```
/***** METHOD IM1 *****/
/* Method to check the status of the Incident object */
METHOD im1(stat:keyword)
VAR
  stat: keyword;

BEGIN
  if _frame.src<>_rc then
    Begin
     send("range") to _frame;
     reason(_frame,"check");
     _frame.src=_rc;
    End;
END.

/***** METHOD IM2 *****/
/* Method to reset the Incident object's status */
METHOD im2(reset:keyword)
VAR
  reset: keyword;

BEGIN
  send("range") to _frame;
  reason(_frame,"reset");
END.

/***** METHOD IM3 *****/
/* Method to set the ranges of the Incident object's variables */
METHOD im3(range:keyword)
VAR
  range: keyword;
BEGIN
  if _frame.rrc<>_rc then
    Begin
     xgrad=0;
     xmode=0;
     if _frame.pgrad<>_grade then
       Begin
        xgrad=1;
        _frame.pgrad=_grade;
       End;
     reason(_frame,"vrset");
     _frame.rrc=_rc;
  End;
END.
```

Appendix E

Knowledge Acquisition Form

INCIDENT REPORT

Species: ______________

Incident

Title: ____

Number: ____

Description: ____

____

____

Consequences

Quality variables effected:

______________ ______________ ______________

Tagnames of the quality variables:

______________ ______________ ______________

Consequence classification:

____

____

Consequence explanation:
(Briefly explain how the incident effects the quality variables and the "classified" consequence.)

____

____

____

____

Causes

Observed Causes:
(Specify the cause of the problem that was observed when the incident occurred.)

____

____

____

Root causes:
(Specify the root cause of the problem and all observed causes after analyzing the incident.)

____

____

____

Tagnames of process points involved in causes:

______________   ______________   ______________

______________   ______________   ______________

Determine the cause:
(Specify how to determine from process data value if the cause is occurring: include limit values.)

Warning: ____

____

Imminent: ____

____

Occurred: ____

____

____

Return to Normal:
(Specify how to determine that the incident is no longer occurring. Allow enough lenience to prevent alarm oscillation.)

____

____

____

<u>Actions</u>
(Describe suggested action to take given the following situations.)

Early warning: ______________________________
(Could occur given the present ______________________________
process trend)

____

____

Imminent: ______________________________
(about to occur within the next ______________________________
5 minutes)

____

____

Occurred: ______________________________
(has occurred already) ______________________________

____

____

Displayed Process Point Tagnames:
(List process point tagnames useful to be displayed on the incident report screen.)

Appendix F

Project Acknowledgment Letter

April 28, 1995

To Whom It May Concern;

Daishowa Marubeni International Ltd. Peace River Pulp Division (PRPD) started up in July 1990, and is now at design production (about 1000 t/d). PRPD is known for its high quality pulp, particularly its hardwood (aspen), which makes up over 70% of the mill's production. The hardwood pulp is used for high quality paper requirements such as printing and photographic paper. The remaining 30% is softwood pulp (spruce or pine) used for tissue and towel products. The mill wide process control facility is with a Honeywell TDG-3000 DCS. MoDo Chemetics process-optimization software runs in the TDC-3000 to ensure optimum digester and oxygen reactor performance. Four stages of chlorine and CLO_2 bleaching are regulated by a BTG brightness control system.

About 30 PC-compatible computers serve as MOPS user terminals. Any terminal on the system can access the mill computer via the VAX local area network. All managers, supervisors and planners have terminals, as do the maintenance shop and control room.

To fully use these advanced facilities and increase "intelligence" to solve more complex problems such as process quality control, operation optimization, and troubleshooting; is a growing concern to the pulp and paper industry.

The current process control and information systems deal mainly with normal situations and provide process operation information. But how to use and manage this information and make decisions for handling process operation problems still rely heavily on process engineers/operators' experience. An experienced operator may still be unable to sufficiently handle operations such as quality control, grade transition, troubleshooting and so on.

DMI and the University of Alberta are currently working on a joint project using Expert Systems to resolve some of the previously discussed issues. Todate there are no expert sytems on the market that focus on the needs of the pulp and paper industry. The university has worked with DMI to identify mill needs and project scope for an Expert system that could be integrated with DMI's process control and information system as well as adapting to the mill's incident reporting procedures. The system is designed to receive operator knowledge, based on the analysis of process incidents, and alarm the operator when a similar incident has the potential of reoccurring.

To date the project is in the mill evaluation stage. Both operations and technical personnel are successfully using the application. Once the prototype has been finalized we will be looking at other areas of the mill where this technology can be applied.

With the level of automation implemented at the Peace River mill, tools are required to assist the operator in evaluating process information for decision making. This project reduces the data overload that is common with new process control and information systems. We believe that this type of research is critical to improving operator effectiveness and productivity.

Keith Danielson

Pulping Superintendent

KD:LH

Appendix G

Process Monitor Activity Logs

```
                    *** POMCS ACTIVITY LOG FILE ***
                       Created: Feb  7 13:04

Feb  7 13:06 - E9403     : Plugged EO Washer Inlet Box          * Detected *
                         : STAT: warn    Prior:       ETO:
Feb  7 13:06 - R930260   : High PH on CD Stage                  * Detected *
                         : STAT: occ     Prior:       ETO:
Feb  7 13:23 - R931697   : CD Sample Line Water Leaking         * Detected *
                         : STAT: occ     Prior:       ETO:
Feb  7 13:53 - E9403     : Plugged EO Washer Inlet Box          * Reset *
Feb  7 13:57 - E9403     : Plugged EO Washer Inlet Box          * Detected *
                         : STAT: warn    Prior:       ETO:
Feb  7 14:47 - E9403     : Plugged EO Washer Inlet Box          * Reset *
Feb  7 15:06 - E9403     : Plugged EO Washer Inlet Box          * Detected *
                         : STAT: warn    Prior:       ETO:
Feb  7 15:21 - E9403     : Plugged EO Washer Inlet Box          * Reset *
Feb  7 15:23 - E9403     : Plugged EO Washer Inlet Box          * Detected *
                         : STAT: warn    Prior:       ETO:
```

Figure G-1.

```
                    *** IOMCS ACTIVITY LOG FILE ***
                       Created: Feb  8 14:01

Feb  8 14:03 - R930398  : Low D1 Tower Level                    * Detected *
                        : STAT: occ     Prior:      ETO:
Feb  8 14:05 - E9403    : Plugged EO Washer Inlet Box           * Detected *
                        : STAT: warn    Prior:      ETO:
Feb  8 14:08 - E9409    : CD Sample Line Plugged                * Detected *
                        : STAT: occ     Prior:      ETO:
Feb  8 14:15 - R930260  : High PH on CD Stage                   * Detected *
                        : STAT: occ     Prior:      ETO:
Feb  8 14:16 - E9409    : CD Sample Line Plugged                * Reset *
Feb  8 14:16 - E9403    : Plugged EO Washer Inlet Box           * Reset *
Feb  8 14:18 - E9403    : Plugged EO Washer Inlet Box           * Detected *
                        : STAT: warn    Prior:      ETO:
Feb  8 14:22 - R931697  : CD Sample Line Water Leaking          * Detected *
                        : STAT: occ     Prior:      ETO:
```

Figure G-2.

```
                    *** IOMCS ACTIVITY LOG FILE ***
                        Created: Feb  8 15:25

Feb  8 15:27 - E9403     : PLUGGED EO WASHER INLET BOX    * Detected *
                         : STAT: warn      Prior:        ETO:
Feb  8 15:27 - R930260   : HIGH PH ON CD STAGE            * Detected *
                         : STAT: occ       Prior:        ETO:
Feb  8 15:30 - E9403     : PLUGGED EO WASHER INLET BOX    * Reset *
Feb  8 15:32 - E9403     : PLUGGED EO WASHER INLET BOX    * Detected *
                         : STAT: warn      Prior:        ETO:
Feb  8 15:37 - E9403     : PLUGGED EO WASHER INLET BOX    * Reset *
Feb  8 15:46 - R931697   : CD SAMPLE LINE WATER LEAKING   * Detected *
                         : STAT: occ       Prior:        ETO:
Feb  8 15:48 - E9403     : PLUGGED EO WASHER INLET BOX    * Detected *
                         : STAT: warn      Prior:        ETO:
Feb  8 16:19 - R930260   : HIGH PH ON CD STAGE            * Reset *
Feb  8 16:46 - R930260   : HIGH PH ON CD STAGE            * Detected *
                         : STAT: warn      Prior:        ETO:
Feb  8 16:54 - R930260   : HIGH PH ON CD STAGE            * Reset *
Feb  8 16:56 - R930260   : HIGH PH ON CD STAGE            * Detected *
                         : STAT: warn      Prior:        ETO:
Feb  8 17:01 - E9403     : PLUGGED EO WASHER INLET BOX    * Reset *
Feb  8 17:03 - R930260   : HIGH PH ON CD STAGE            * Reset *
```

Figure G-3.