UNIVERSITY OF ALBERTA

Hybrid Intelligent Matrix Simulation System For BCTMP Process

by

Hassan Farzadeh ©

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

FALL 1996

National Library of Canada

Acquisitions and Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada

Direction des acquisitions et des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-18257-6

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR:	Hassan Farzadeh
TITLE OF THESIS:	Hybrid Intelligent Matrix Simulation System For BCTMP Process
DEGREE:	Master of Science
YEAR THIS DEGREE GRANTED:	1996

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

[illegible]

P.O. Box 1790
Slave Lake, Alberta
Canada, T9E 5M8

May 9/1996
DATE

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommended to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Hybrid Intelligent Matrix Simulation System For BCTMP Process** submitted by **Hassan Farzadeh** in partial fulfillment of the requirements for the degree of **Master of Science in Process Control**

[illegible]

Dr. M. Rao, Supervisor

[illegible]

Dr. D.G. Fisher

[illegible]

Dr. M. Meng

May 9, 1996

DATE

ABSTRACT

For the past 20 years, pulp and paper mills have invested in information technology with the firm belief that having more information would indirectly improve mill operation. Slave Lake Pulp Corporation (SLPC) in Alberta, Canada has installed a mill-wide information management system, MOPS, developed by MoDo Chemetics. MOPS collects and stores all the crucial process data points synchronously or asynchronously from the DCS, bale handling system, power monitoring system, and manual test entries. Operators use MOPS graphical displays to monitor quality, production, and chemical consumption. MOPS helps operators to check the status of the mill quickly, make decisions efficiently, and access the operating conditions for new production or repeated grades. However, the mill operation still heavily relies on operator's experience. How to apply intelligent system technology to solve the problem is the topic of this research.

This thesis describes the development of an Intelligent Matrix Simulation system (IMS), a hybrid simulation system for the operational support of a BCTMP pulp mill. IMS combines features from neural networks, case based reasoning, rule-based and frame-based intelligent systems and simulates the process behavior. The IMS knowledge base is organized using Meta-COOP frames. The BCTMP process is divided into smaller sub-processes such that each sub-process can be represented in one frame. The inferencing in IMS is divided into a number of local systems in terms of system functions and/or process decomposition. Each local system integrates case based reasoning (CBR), numerical calculation (NC), heuristic rules (HR), and neural networks (NN) to solve individual problems. IMS incorporates time-based historical process data from MOPS, operational case knowledge from the relational database of Bale Quality Information System (BQIS), mathematical simulation models, and experts knowledge from the

proce[illegible] and operations. It also serves as a knowledge base for on-line fault diagnosis and emergency handling in another module of IOMCS project.

IMS is an on-line real-time intelligent system developed in a client - server environment with a server being an Alpha Open VMS and clients being the PC workstations running Microsoft Windows 95 or Windows NT. The prototype of IMS system has been successfully installed in the mill computers. It has been proven useful in assisting the daily operation of the mill. The enhancement of the IMS system is continuing on-site since the author is employed by the mill.

ACKNOWLEDGMENTS

I would like to express my appreciation to a number of people for their valuable assistance in the completion of my thesis. First and foremost, I would like to thank my supervisor, Dr. Ming Rao, for his direc'ion on the project and help. I would also like to thank the members of my thesis committee: chairman Dr. Williams, Dr. Fisher, and Dr. Meng who graciously gave of their time and help to my thesis.

I would also like to express my appreciation to all my co-workers at Slave Lake Pulp Corporation for their input and advice regarding the direction and needs of the project. Technical support from MoDo Chemetics, Perde Enterprises, and Hyper Consulting is appreciated.

Another group of people to which I am indebted are those from Intelligence Engineering Laboratory. These people helped me for the past four years with their expertise and time. Especially Dr. Qijun Xia and Dr. Yiqiang Shu.

Finally, I would like to thank my wife for her support and encouragement during my graduate studies and research.

Financial support for this project by Slave Lake Pulp Corporation and Canada/Alberta Forest Partnership as well as Natural Sciences and Engineering Research Council (NSERC) is greatly appreciated.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABREVIATIONS

ADMT	Air Dry Metric Tonne
ANN	Artificial Neural Network
BCTMP	Bleached Thermo-Mechanical Process
BDMT	Bone Dry Metric Tonne
BQIS	Bale Quality Information System
CBR	Case Based Reasoning
CVD	Current Value Database in MOPS
DBMS	Data-Base Management System
DEC	Digital Equipment Corporation
HDB	Historical Data-Base in MOPS
HR	Heuristic Rule
IDIS	Integrated Distributed Intelligent System
IE	Inference Engine
IMS	Intelligent Matrix Simulation System
KB	Knowledge Base
KBS	Knowledge Based System
MOPS	Mill-wide OPtimization System
NC	Numerical Calculation
NN	Neural Network
OOP	Object-Oriented Programming

RDB	Relational Data-Base developed by DEC
RDBMS	Relational Data-Base Management System
SQL	Standard Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol

Chapter 1

1. INTRODUCTION

In the 1990's, the world-wide movement to improve product quality and to protect the environment has prompted pulp and paper companies to look for new technologies that can aid them in meeting the market and public demands, while remaining cost competitive. The extensive applications of computer based information management system is one of the major moves taken by pulp and paper companies. Currently, modern pulp and paper mills have installed Distributed Control Systems (DCS) and mill wide information management systems.

Slave Lake Pulp Corporation (SLPC) is utilizing the best available technology to produce a high quality market pulp product. The mill has implemented the Mill-wide OPtimization System (MOPS) [Chemetics 1995] , an integrated value-added information system, and the Bale Quality Information and tracking System (BQIS) [Prime 1994]. These information systems collect over 2300 process data points per minute from many different sources and store them in the databases. These information systems perform such functions as material tracking, statistical process control, report automation and analysis. MOPS helps operators and process engineers to check the status of the mill quickly, and to make decisions efficiently. BQIS helps the marketing department to sell and serve its customers efficiently.

Product quality and production profits are influenced by many parameters from process conditions to raw material supplies. The knowledge concerning the operation includes complex technologies from different areas. Using MOPS, however, the operation still heavily relies on operators' expertise and experience [Frith et. al. 1992].

Faced with the vast amount of data from MOPS, even an experienced operator may find it difficult to deal with some operations such as grade transition, process optimization and trouble shooting. Typically, at any given time, there are only two operators controlling the process and handling the large amount of data [Farzadeh et. al., 1996]. The industry has an urgent need for technology that can extract useful and relevant knowledge from the vast amount of available data to assist the mill operations. Process modeling, simulation and process optimization are the important means to tackle the problem. The next section will give a brief overview of existing process modeling and simulation techniques.

1.1 PROCESS SIMULATION

In pulp and paper, chemical, petrochemical, and petroleum industries, there is a wide spread in the use of process modeling and simulation technology over the past two decades at almost all levels of engineering activity, including process research, development, and design as well as plant operation. Process modeling today still requires significant skill to be carried out efficiently. The ultimate objective of any simulation is to develop a computer program that is capable of simulating the response of engineering facilities, that is the response of a real plant to control signals issued to the field devices by operators or computers. The complexity of modern control systems and the potential consequences of errors in those devices signals the deadly need for such kind of simulators [Bozenhardt, 1990]. The common applications of process modeling and simulation in process industries include:

- *Process design and development*: Process design and development are the earliest users of process simulation. Traditionally they use static mass and energy balances for designing equipment and determining operating conditions. A simulation can be used advantageously in the design of real plants to detect bottlenecks or under-used units and to test the reaction of the system to non-nominal production conditions, and thus to improve the original design. Using only the static model, however, is not sufficient. Recently, as the more sophisticated simulation tools became available, dynamic models are used for improving the design.

- *Control system design or revision*: Modern process industries are controlled with DCS and/or PLC systems. With a simulation model, control engineers have a test bench for control schemes. A control scheme can be tested against plant simulators before final installation, and the control system design and revision can be safely done. One particularly interesting and worthwhile application area is the integration of process design and control since it is of utmost importance to uncover potential control problems at an early stage of the process design.
- *High level operation support functions*: The more sophisticated operation support functions, such as operation optimization and trouble shooting, need a plant simulator as its model base. The operators also need to simulate the control action before finally issuing it.
- *Operator training*: The process simulator can serve as a training tool for the new operation personnel to familiarize them with the plant. There has been a growing interest in the use of training simulators in the process industry. There are many reasons behind it: a desire to increase overall productivity, the need to comply with changes in safety and environmental legislation and, most importantly, the wish to reduce the risk associated with human error. For example, because of obsolescence, most production companies have upgraded or are planning to upgrade the control system to a DCS. However, there is usually concern among

many operating personnel about the control system conversion. Process operators are reluctant to switch from their familiar pneumatic board system to a "television like" electronic system. It becomes obvious that a thorough training program, including hands-on experience, will be an essential element of the control system effort.

- *Maintaining and accumulating knowledge*: Most of knowledge from the original designers of the plant is separated and have never been updated. The knowledge from the individual mill engineers and operators that they acquired in real time operation is seldom kept in official documents. Process simulators are the best tool and technology to unite, maintain and accumulate the knowledge from the original designers, mill engineers, and process operators.

1.1.1 NUMERICAL MODEL BASED PROCESS SIMULATION

A traditional simulation environment consists of simulation blocks linked with each other. The simulation blocks are second generation based programs (such as FORTRAN-based subroutines) with hard-coded, pre-determined solution methods. Inside the subroutines is usually a set of differential equations. The solutions to differential equations must be developed. The drawbacks of this kind of the methods are:

1) A simulation consist of hundreds of simulation blocks. This makes it very difficult for the simulation to be extended beyond the original scope.

2) Since the influence of any part of the subroutine can not be isolated, substantial effort is required when there is any change in the process, such as feed-stock, flow configuration, equipment failure of transitions of flow, energy or mass regime.

From an engineering point of view, one of the critical requirements of a good program is that users should only need to identify the engineering parameters associated with each process component, and components relationship, i.e., only to create a database. No further programming in computer languages should be required. However, the traditional simulation environments do not have such capabilities.

To solve the problem, a modular approach has been employed for both steady state simulation and dynamic simulations. In the modular approach, the units (equations representing these units) are solved one at a time according to a pre-defined calculation sequence. The modular approach have made the simulation more flexible. This method is structured because it partitions the equations into smaller subsets which can be solved separately or simultaneously. It is a mixed mode because it allows dynamic and non-dynamic units in the same flow sheet as well as simultaneous solutions of steady-state and dynamic simulation problems. Also, the steady-state model and dynamic model are represented by the same set of equations, specified variables and state variables and the equations are solved by the same numeric method.

The sequential modular approach has been proposed and vastly adopted in simulation for process flow sheeting because of its robustness and reliability [Evans, 1981; Westerberg et al., 1979]. In this approach, the simulator consists of modules for each kind of equipment which compute the outlet streams given that the inlet streams and equipment parameters are input data. Knowledge about the equations involved in each module and its specific task permits testing of data, proper initialization, verification of results, etc. When solving a steady state flow sheet, the solution from one module is used as input to the following module in the sequence. Closed loops are solved by tearing streams, and iterating until the tear streams converge. Examples of steady state sequential modular simulators are FLOWTRAN [Rosen and Pauls, 1977] and Aspen Plus [Aspen, 1988]. In dynamic sequential modular simulators, the states and input stream values in each module are known at the beginning of each time step. The values at the following time step might be calculated explicitly from the states and input values at the present time step. Another possibility is to calculate the values at the following time step implicitly by sequential calculations. Here the output values at the present time step from the module earlier in the sequence, are used as input values to calculate both the states and the output of the actual module. For both schemes each module might contain its own integration routine, and the flow sheet calculation might be a combination of explicit and implicit calculations. Examples of dynamic sequential modular simulators are DYFLO [Frank, 1972] and CADAS [Eikaas, 1990].

However, this rigid problem formulation gives rise to some disadvantages: the nested nature of the calculations. There is an iterative procedure to converge the tear streams; each one of these interactions involves an iterative method to converge each module, and finally each one of the last iterations involves an iterative procedure for physicochemical calculations. As a result, the required CPU time is large. Besides, design and optimization problems require a new iterative loop added at the top of the former nested loops. Two alternative approaches has been proposed: equation-oriented and simultaneous-modular simulators. In the equation oriented approach, the equations describing all unit operations are gathered into one large equation system, representing the whole flow sheet. For the steady-state case, this is a set of nonlinear algebraic equations, represented by a spare occurrence matrix with asymmetric blocks along the diagonal. Usually the equation set is solved by a Newton like method. When the equation oriented approach is applied to dynamic cases, the equation system is a set of ordinary differential and algebraic equations (DAE), which is usually stiff, sparse and nonlinear. When solving stiff equations, generally two solution strategies are applied to solve the equation set. One strategy is to use a DAE solver directly on the set of equations. Another possible strategy uses a two level solution scheme, employing an ODE solver to solve the ordinary differential equations and an AE solver to solve the algebraic equation [Moe and Hertzberg, 1994]. Examples of equation based dynamic simulators are Ascend II [Locke and Westerberg, 1983] and DynSim [Sorensen, 1990], while SpeedUp [Perkins and Sargent, 1982] is a combined steady state and dynamic simulator.

A way to overcome the difficulties of the sequential modular approach while retaining its advantages is to remove the constraint requiring that streams have the same logical direction as in the physical problem. Associated with the process flow sheet is a directed graph where the arcs have the same directions as material steams. Differentiating between the material flow in the process and the logic flow in the resolution, the direction of the streams is determined to ease the resolution and to take advantage of available information. This new strategy [Montagna, 1993] has been incorporated and used in the process simulator SIMBAD [Leone et al., 1987; Motagna, 1993].

The simulation for batch plants is also an important research topic. Specialty chemicals are in most cases produced in small quantities, and therefore flexible batch plants are more economical for such production processes. In continuous flow plants, the structure of the process is constant during normal operation. Adequate simulation models therefore have a fixed structure. The model are constructed either by explicitly writing the differential and algebraic equations in some programming languages, or, in more modern systems, by graphically combining basic blocks which corresponding to individual unit operations. The batch plants are characterized by the lack of a "steady state". Firstly, the operation in the single process units are not stationary but characterized by a trajectory. Secondly, depending on the actual use of the process units, the dynamics and the couplings of the plant are different, not only in the sense of parameter variations but also in their principal structure. In most cases, the sequence of batches in the plant is also variable. Different products are produced in different quantities and under certain constraints (e.g. possible sequences in the variation of product properties due to a nonideal separation of the batches), with variable urgency or time-pressure. Thus for a complete analysis, orders as a temporal factor and the resulting scheduling of the process units must be included if the operation of a plant is to be simulated. Engell and Wollhaf [1994] proposed a structure of a simulation system for the operation of flexible batch plants which includes both continuous and discrete dynamics. In the simulation system there are two types of objects: "units of material" and "plant units" which both have internal (continuous and discrete) dynamic states. According to the recipes and the decisions on production sequences, dynamic processes are generated dynamically by the combination of these two types of objects. A number of such process - interacting or not - is simulated until a phase in one of the units is stopped. Then all related dynamic simulations are stopped, the state variables of the material and the equipment are updated, and the system is reconfigured according to the recipe for the next production phase. Usually the dynamic simulation is thus embedded into a discrete-event system and is no longer static but a dynamic element itself. The high-level controller is a purely discrete dynamic system.

1.1.2 INTELLIGENT PROCESS SIMULATION

Intelligent process modeling, simulation and design require the intuition and experience that is not provided by flow sheet simulators. Sometimes, the rigid differential equations are difficult to obtain due to the non-linearity and lack of complete understanding of process mechanism. Intelligent simulation have the potential to meet the demands of process industry.

Intelligent system technology has rapidly gained applications in process simulation since 1980. In the early 1980's, the first generation of AI tool emerged, such as OPS5, Prolog and LISP. These systems provide capabilities such as data driven computation, knowledge acquisition, and intelligent code generation. However, they were very difficult to integrate with existing systems and ran on specialized hardware. The second generation of AI tools gradually replaced the first generation from 1980 to 1987. These tools had the capability to model and prototype industrial systems. Representatives of second generation systems are KEE, ART and Nexpert. The major obstacles for the application of these systems [Ponton, 1991] are speed, performance and the development time. Since 1987, the third generation of expert system tools have been developed, such as Mercury KBE. The third generation systems are large hybrid systems which possess the knowledge of simulation experts, allowing the use of multiple representations of mathematical models, and have "built in" techniques for dynamic simulation and simultaneous solutions. These techniques are chosen, implemented and monitored automatically by the expert system. The system has a wise user interface to allow any user unfamiliar with simulation and modeling technologies to generate and use rigorous dynamic process simulations. We called the third generation of simulation tools "intelligent simulation tools". Intelligent simulation tools are a cross-linked field of artificial intelligence and process simulation.

An important development of intelligent simulation was the design of DECADE [Gallant, 1988], a prototype expert system for catalyst selection. DECADE contained limited information concerning the Fisher-Tropsh reaction. This system was implemented with Franz-Lisp, its extension SRL1.5, and OPS5, utilized a hybrid blackboard system architecture. Within the knowledge sources, the knowledge base is represented by production rules (OPS5), frames (SRL), and procedures (Franz-Lisp); and the inference engines utilize recursive, depth-first search for classifying reactions and means-ends analysis for the proposal of reactions steps on surface of the metal. DECADE proved to be capable of recommending materials that can lead to a specific product and suggesting operating conditions in terms of temperature and pressure.

Since flow sheet simulation has been successfully applied in process simulations, a broader approach for process simulation is to create an expert system environment for flow sheet simulation. There are two methods to accomplish this objective: one is using the same programming language for flow sheet simulator and expert systems; another is using a different language for each. An important issue for such intelligent simulation environments is the integration of flow sheet simulators that are typically written in a procedural language with expert systems that are commonly written in a symbolic language [Biondo, 1992], a practical approach is to develop an intelligent front end for flow sheet simulator. For example, PROCESS is a large FORTRAN program that runs in batch-mode on large or medium scale computers. Fjellheim [1985] developed an intelligent front end for the PROCESS flow sheet simulator using LISP and LOOPS. The objective was to produce a knowledge based support system for creating and modifying PROCESS flow sheets for off-shore process plants. This knowledge based support system provided assistance for PROCESS modeling that supported the engineer by incorporating some of the modeling technology in a formal knowledge base.

Stephanospoulos [1987] developed a software support environment, namely DESIGN-KIT, to aid process engineering activities, including the synthesis of preliminary process flow sheets, configuration of control loops for complete plants, planning and scheduling of plant wide operations, and operational synthesis. DESIGN-KIT was implemented on an AI workstation and used an equation-oriented approach for flow sheet simulation. The prototype is operated in a single uniform programming environment, utilizing SYMBOLICS' Common LISP Flavors to create panes, command menus, and mouse actions and to represent other objects. DESIGN-KIT is embedded in IntelliCorp's KEE frame to describe processing units, control loop components, process operations, design methodologies, and production rules.

SALT is a flow sheet simulator written in PL/I. Biondo [1989] created an expert system environment for SALT and a knowledge based front end for ASPEN/SP (written in FORTRAN) with expert system shells written in Pascal. The environment integrated a flow sheet simulator, statistical estimation models, and an expert system in a mainframe environment to provide access by laymen to a tool for estimating costs and predicting performance of advanced technologies. Programs written in REXX were used to interface SALT with external routines in the Expert System Environment, a shell written in Pascal.

To solve the problems that arise from the lack of complete understanding of process mechanism, qualitative simulation is proposed for handling incomplete knowledge about dynamic systems. De Kleer and Brown [1984] and Kuipers [1989] proposed qualitative differential equations for describing and handling incomplete knowledge about dynamic systems. Their form is derived from the ordinary differential equations of the process with special qualitative quantities and dependence functions. It is possible to perform qualitative simulation using qualitative models to obtain predictions about the behavior of a process. By replacing all qualitative terms in a qualitative differential equation by formalized expressions, one can get a quantitative model and simulation as a limit.

"Object-oriented programming" is a technology which can address the requirements for the simulation of modern process industries. In an Object-Oriented intelligent simulation system, each object simulates one real-hardware components of the plant. The objects can be elementary or sophisticated. The LISP-based product of Artificial Intelligence, Inc., named "Mercury KBE" is such kind of a tool.

In an intelligent object-oriented simulation system, each component is implemented separately. The interactions among the components are defined by using an interface which allows flexible simulations to communicate with different components without significant impact on the original system. The major advantage of the system is to integrate the variety of functions, such as :

- equations of mass transfer

- equations of heat transfer
- equations of momentum transfer
- heuristics or rules
- integration of methods to solve differential equations
- database calls/searches
- probabilistic calculation
- simultaneous equation solver
- lead/lag transfer functions
- equipment performance characteristics
- instrumentation characteristics,
- and so on.

The object-oriented approach is a modular approach to software development. There are several desirable characteristics associated with building an intelligent simulation system for industrial process. First of all, the underlying simulation methods and modeling techniques can be data driven by the declarative component of the knowledge base. That is, by changing the attributes of objects which are undergoing simulation cases, the optimal simulation procedures are automatically configured and utilized. It eliminates the need for either modification of the procedural code or running a sub-optimal simulation. Secondly, the simulation has its intimate knowledge about the domain. Each module or unit operation can be modeled independently by a domain expert and inserted into configuration without modifications to any other code. A well developed intelligent simulation environment allows the user to easily install, use and modify the system, without the requirement to know the details about computers and simulation technology. However, the intelligent simulation system have the following limitations:

1) Complex knowledge acquisition process: The knowledge acquisition has come to be seen as a bottleneck in the process of building knowledge-based systems (KBS).

2) Inability to learn: Most of the KBS's developed are fragile, and hence are

unable to recognize the changes in the environment. KBS's do not provide robust learning strategies to update their knowledge effectively.

3) Inability to handle the huge amount of data: Current KBS's do not provide efficient management for large and distributed knowledge bases. This is mainly due to the complex structure of the knowledge base entities required to facilitate the reasoning capabilities.

4) Domain dependence and validity: The knowledge bases for many KBS's are developed for the specific domains. Beyond these domains, the performance degrades dramatically.

5) Slow execution speed: KBS's incorporate large data structures, and their search strategies are computationally expensive, thus resulting in slow execution speeds.

1.1.3 NEURAL NETWORKS FOR PROCESS SIMULATION

Because of a lack of complete understanding of the process mechanism, differential equations which describe the dynamics of the processes are very difficult to obtain. Thus, the knowledge base in intelligent systems is also very difficult to develop. Because of these problems, after two decades of near eclipse, the technology of neural networks has been receiving a great deal of attention in process industry, especially in the field of process modeling and simulation [Rehbein et al., 1992]. A neural network is an information processing method that works better than traditional computing methods on certain problems. As the name implies, neural networks are somewhat related to biological systems – which can learn and adapt to changing environments. Neural network technology borrows ideas and inspiration from biology, and some terms used to describe neural networks reflect the association. Several factors motivate the use of neural networks:

- They are capable of extracting essential characteristics from vast amount of data which contain noisy and irrelevant information.
- They are able to learn from experience, whereas most other techniques rely on pre-programmed algorithms.
- They have the capability to generalize new knowledge from previous examples.
- They exhibit a greater degree of robustness and fault tolerance.
- They can respond to sensor inputs quickly.

There are several opportunities for applying neural network technology in the process industry. Process models can be developed for which first principle models are difficult to produce. If adequate process history data is available, a sufficiently accurate neural network model can be produced. Even in the case in which a first principle model is available, a neural network model can be built from the first principle model output. With the neural network model, the product quality results, which typically require a time delay to allow lab analysis, can be predicted on-line, allowing process corrections in a more timely manner. Neural networks can detect failed sensors and provide failure alarms. In addition, a reconstructed value can often be created by the neural network model for use while the instrument is out of service for repair.

Presently, neural networks are only used for the modeling a single process unit. To apply neural networks to the modeling and simulation of plant wide system, the crucial issue is to integrate neural networks with expert system and conventional differential equation based system. The integration is discussed in several research papers in the medical field [Hudson et al., 1990], but there are very few applications reported in the process industry.

1.2 SCOPE OF THIS RESEARCH PROJECT

The above techniques all have their unique advantages and disadvantages. Most of the existing process simulators either have complex and computationally extensive heuristics to make them optimal and adaptive, or use a single scheme at all times resulting in performance degradation. The knowledge coded in the system is mostly plant specific, not well developed and highly correlated with plant status. Clearly none of the existing systems can meet the goals of a research project that is sponsored by industry and government. Therefore, have analyzed the fundamentals of human behavior and thinking patterns to develop a system which can emulate these human behaviors.

A way to improve the performance of process simulators is to incorporate the human reasoning method in system design and to utilize the best features from all of the above methods. Typical steps of human reasoning can be summarized as follows:

- to create categories
- to use specific rules
    - rules can be cascaded
        - "If A then B" . . .
        - "If B then C"
        - A--->B--->C
- to use heuristics --- "rules of thumb"
    - heuristics can be captured using rules
    - heuristics represent conventional wisdom
- to use past experience --- "cases"
    - particularly evident in precedence-based reasoning
    - store cases using key attributes
    - pump may be characterized by: size of pump, manufacturer of pump etc.
- to use "expectations"

In order to make the computer-based simulator perform reasoning similar to humans, the computer models can be organized in the following way:

- Frames
    - frame attributes called "slots"
    - each frame is a node in one or more "is a" hierarchies
- Computers use rules A--->B--->C
    - Set of rules is called knowledge base or rule base
- Computers use cases
    - Set of cases is called a case base
- Computers use pattern recognition/expectations

The integration of the different simulation methods in a harmonious environment could enhance the capability and generality of process simulators. For example, knowledge-based system (KBS) technology has been concentrating on the construction of high performance programs specialized in limited domains. ANN's can analyze the large quantities of data to establish patterns and characteristics in situations where rules are not known and can deal with incomplete and noisy data. Case-based reasoning methods (CBR) can utilize the previously available cases in the simulation program. Therefore, KBS, CBR, and ANN can be integrated to solve tasks that require different problem solving techniques. Such an integrated system can provide enhanced inferencing functionality and dynamic architectural control to develop approaches for traditional problems such as pulp process modeling and simulation.

In an effort to solve the above problems, the Intelligence Engineering Laboratory at the University of Alberta, MoDo Chemetics, Slave Lake Pulp Corporation, Perde Enterprises, and Canada - Alberta partnership in forestry initiated this project to develop an intelligent process simulator, called **Intelligent Matrix Simulation System** (IMS), to be used for answering "what-if" types of scenarios in process operation and for training inexperienced operators. IMS is based on the concept of integrated, distributed, intelligent systems [Rao, 1991]. It is a knowledge integration environment that applies Artificial Intelligence (AI) techniques to solve complex problem by combining independent specialized software packages. The key construct to this architecture is the Meta-COOP, an expert system building tool developed at the Intelligence Engineering Laboratory. Meta-COOP coordinates, integrates and communicates with the individual software packages.

Based on symbolic inference and numeric models, IMS displays the relations of process variables graphically for various operating conditions. The main functions are: (1)) Extraction and accumulation of process engineers/operators' private knowledge; (2) process optimization; (3) on-line simulation of process operations; (4) operator training. IMS can also be used as a knowledge base for other modules such as trouble shooting. Another important purpose of this project is to develop an expert system building shell in the MOPS such that users can develop their own knowledge bases for different process units. IMS is embedded in MOPS and increases the "intelligence" of MOPS.

The thesis is organized in the following manner: the first chapter is this introduction, which covers the general objectives and motivation of the project. Chapter 2 presents the background about Bleached Chemical Thermo Mechanical Pulping (BCTMP) and existing Mill Information Systems in Slave Lake Pulp Corporation (SLPC). Chapter 3 describes the system architecture of IMS. Chapter 4 presents the knowledge integration. In chapter 5 the integration of IMS with MOPS and BQIS is described. Chapter 6 presents the conclusions.

Chapter 2

2. DESCRIPTION OF BCTMP PROCESS AND MOPS

Using the BCTMP (bleached chemi-thermo-mechanical) process, Slave Lake Pulp Corporation, (SLPC) produces market bleached pulp from aspen or mixtures of aspen/softwood. The BCTMP process combines thermal energy, chemical pre-treatment, mechanical refining energy and hydrogen peroxide bleaching.

2.1 BCTMP PROCESS

The process includes three-stage pressurized refining equipment, two stages of peroxide bleaching, and low-temperature flash drying. The first bleaching stage is an interstage configuration between the primary and secondary refiners. The process flow diagram is shown in Figure 2.1. Aspen roundwood is delivered in treelength form, offloaded by portal crane and either placed on the log feed deck or into log storage. The storage area accommodates up to four months storage. Aspen roundwood is debarked using mechanical ring debarkers, and then fed to a 117-inch chipper. The manufactured aspen chips are fed to a storage pile of nominal 10 days storage capacity. The aspen component represents the majority of the wood supply and the remainder is supplied as by-product from sawmills. The softwood is brought to the mill by chip truck and stored in a separate chip pile.

The chips are fed from storage by stoker-type reclaimers. The reclaimers are fed by a mobile chip loader. A metering conveyor from each of the stoker reclaimers delivers the hardwood and softwood chips in a pre-set mix to a belt conveyor which feeds the chip screening system. Accepted chips are discharged to the main chip conveyor feeding the pulp mill's No. 1 atmospheric pre-steaming bin.

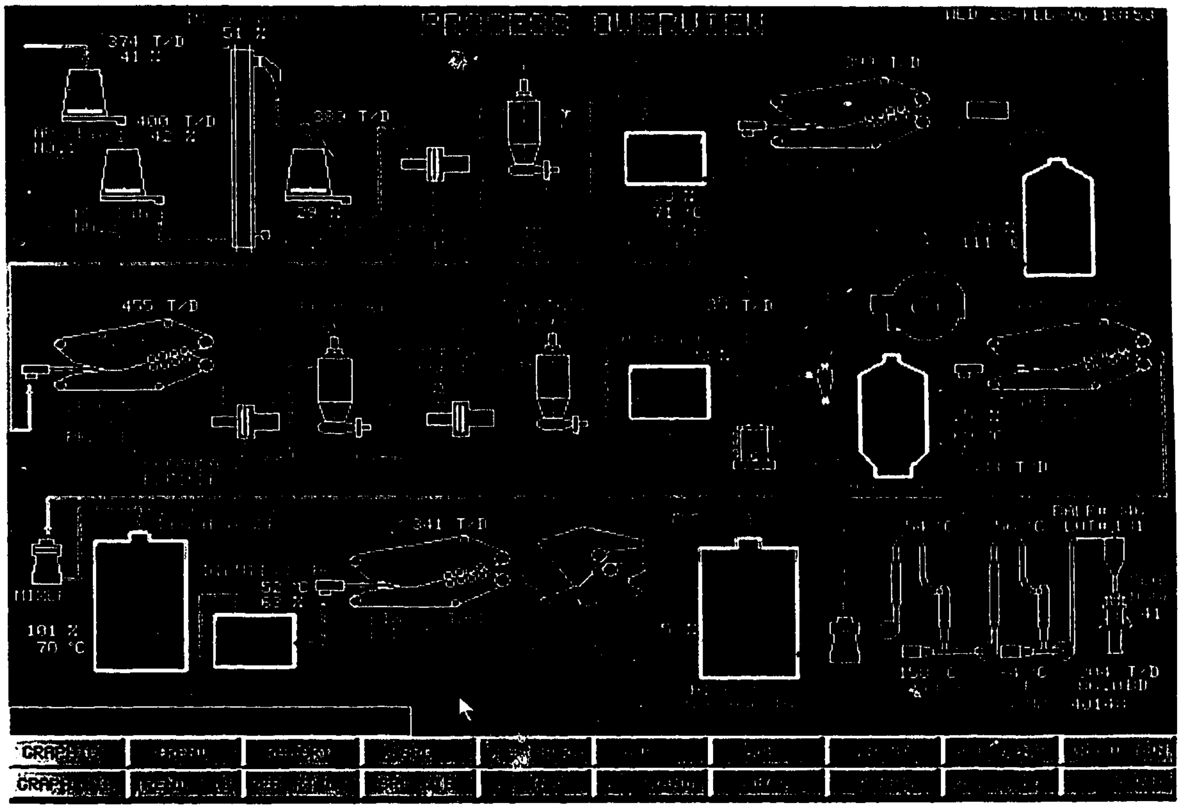


Figure 2.1. Process flow diagram at SLPC

2.1.1 Chip Washing and Conditioning

A 30-minute atmospheric pre-steaming bin (APS 1) with screw-type discharger is followed by a metering screw discharge conveyor and down-chute to a scrap separator and chip transfer tank from which the washed chips are pumped to a double-screw drainer feeding the Chemical Impregnation Plant. From the chip drainer, the chip slurry water is removed and directed to a sidehill screen to a separate chip fines and other floatable material.

A sand settling tank with driven scraper collects chip-conveying water from the chip drainer sidehill screen and the preheater screw conveyor. The clarified water from the sand separator is cleaned and re-used for dilution in the chip transfer tank and the scrap separator. The sand separator rejects are discharged to the effluent treatment plant sludge handling system, and the chip washer/scrap separator rejects go to the rejects screw thickener and bunker. From the bunker, they are conveyed by front end loader to the waste wood incinerator. The drained chips are screw-conveyed to the chemical impregnation chip surge bin (APS 2). From the chip drainer, the entire chip flow can bypass the Chemical Impregnation Plant to feed the primary refiner preheater directly. Drained chips are conveyed to the APS 2 bin having a storage capacity of 15 minutes at 85% full.

From the APS 2 bin discharger, chips are fed to a plug screw feeder, compressing the chips to approximately 50 to 80 % consistency to remove resins, extractives, and air before being released into the atmospheric impregnation vessel. The chips are vertically conveyed through an impregnation solution, then discharged after three to five minutes retention, and transferred by screw conveyor to the reaction chip bin (APS 3) with 20 minutes storage capacity. From No. 3 APS bin, chips are conveyed to the primary refiner conveyor system.

2.1.2 REFINING

Chips from the No. 3 APS bin are discharged to the primary refiner system comprising a plug screw feeder, pressurized refiner and discharge blow line to a pressurized pulp cyclone. Recompression of the chip mass in the plug screw provides further removal of resinous material and also ensures a uniform feed rate to the refiner which is designed for a maximum working pressure of approximately 450 kPa. The plug screw feeder also provides a pressure seal for the refiner which may be operated at a different pressure than adjacent process elements.

The primary refiner is equipped with a 23,000 HP synchronous motor to drive the single rotating disc unit. Pulp fibre developed in the primary refiner is discharged through blow lines to a pressurized pulp cyclone at a consistency of 48 - 50 percent dryness. Cyclone operating pressure is 170 kPa or higher, controlled by pressure regulating valves on the cyclone vent lines. An outlet device on the bottom of the pulp cyclone provides the pressure seal between the unit and its pulp discharge conveyor. Pulp from the discharge conveyor of the primary cyclone is fed to an inter-stage bleaching system comprising stock dilution chest, wash presses, and medium consistency bleach tower. From the bleach tower, pulp stock is diluted and conveyed to second stage washers which thicken the fibre to the required feed consistency for second stage refining.

Partially refined pulp from inter-stage bleaching is screw-conveyed to the single secondary refiner through a metering screw conveyor and a plug screw feeder. The second unit, also equipped with a 23,000 hp synchronous motor, also discharges pulp via blow lines to a pulp cyclone, with generated steam vented to the heat recovery system. The refined pulp discharge from the secondary cyclone is released to atmospheric pressure via an outlet device to an agitated latency chest of approximately 30 minutes storage capacity, to remove curl from the pulp.

The diluted stock from the latency chest is pumped to the suction of the primary screens fan where a pump feeds the two primary screens in series. The primary screens accepts are discharged to the suction of the primary cleaners fan pump, located at the cloudy white water chest, while the rejects from these screens are discharged to the suction of the secondary screen fan pump, also located at the cloudy white water chest.

The accepts from the secondary screen are returned to the pump suction of the primary screen fan pump, with the rejects from this screen discharged to the rejects collecting tank. From the rejects tank, reject stock is thickened on a sidehill screen, then transferred to the transfer chest or the dilution zone of the interstage bleach tower.

Screened stock is diluted with cloudy white water and pumped to the primary cleaner bank. The accepts from the primary cleaners are thickened on a disc filter and discharged through a thick stock pump to a semi-bleached high density storage tank of eight hours pulp storage capacity. The cleaner rejects are pumped to the secondary cleaners, the rejects from which will be processed through tertiary cleaners, while the secondary accepts are re-introduced to the primary cleaners feed. Rejects from the tertiary cleaners are further treated to remove clean stock in fourth and fifth stage cleaners. Rejects from the fifth stage cleaners are discharged to the effluent treatment sludge handling system.

2.1.3 BLEACHING

Pulp from the refiner plant semi-bleached high density storage tank is diluted to 5% consistency in a conventional dilution zone and centrifugally pumped to the third stage of washing. Two double wire presses wash and thicken the stock to 35% consistency before it is fed to high consistency mixers where peroxide bleach chemical is added. From the mixers, stock is discharged by gravity to the second stage peroxide bleach tower, with a nominal retention capacity of three hours. From the second stage bleach tower, the fully bleached pulp is withdrawn by a discharger and fed to a dilution chest for pumping to the Dewatering Plant.

The Bleach Plant, comprising refiner inter-stage bleaching and second stage high consistency bleaching, as described above, is capable of achieving 85 degrees brightness on aspen pulp. Efficient washing and the recycle of bleach liquor contributes to the bleaching cost economy.

From the dilution chest of the second stage bleach tower, pulp is pumped at 5% consistency to the final stage of washing and dewatering to 50% consistency. Dewatered pulp is conveyed to a one-hour capacity bleached high density tower or directly to the two pulp fluffers, located in the Flash Drying Plant. The conveyors from the dewatering presses are the reversing type, to permit pulp production to be re-routed to a repulper and broke tank, in the event there is a short-term curtailment of production in the downstream processes.

Both strong and weak pressate from the wash presses are collected in separate pressate tanks and re-introduced to the process in a counter-current flow to maximize wash efficiency.

2.1.4 FINISHING

The flash dryer is a two-stage unit with a cooling stage. The dryer is a low-temperature type, and operating temperature is limited to approximately 60°C in the first stage cyclone. The second stage drying tower is operated at approximately 50°C. Emissions from the dryer are designed to meet standards set by Alberta Environmental Protection.

Dried pulp discharged from the second stage cyclone enters the cooling stage transport fan for conveying to the cooling stage cyclone located above the slab press in the bale finishing line. Air supply to the cooling stage is controlled to approximately 35 °C using a mixture of fresh air, slab press exhaust return air and interior mill ambient air.

The dried pulp is finished in a baling line, consisting of slab press, bale press, bale moisture meter, top and bottom wrapper dispenser, tyer, folder, jet printer, and scale. The finishing line provides for maximum flexibility and security of bale finishing operations, including exit/re-entry points and storage conveyors off the finishing line. A moisture meter is installed in the baling line for continuous measurement and recording of weight and moisture content. The moisture sensing equipment is located in the bale press plate.

2.2 MILL WIDE INFORMATION MANAGEMENT SYSTEM

The key indicators used by analysts to define the global pulp and paper industry are per capita consumption of paper, total tonnage produced and cost per tonne. One of the most impressive statistics on the global industry is that of making more product with lower cost. One major reason is that the industry is blessed with suppliers which are spending money on research and development to produce new technologies that will enhance both productivity and quality. However, today's technology still requires mill operators to determine what they need and how they will use it to get the most out of their equipment. To unleash the power of new technology requires more than just installing it at a mill.

2.2.1 MOPS

The MOPS management system is divided into two basic modules: MOPS core programs and MOPS workstation. The core program is written in C and C++ and resides in a Digital Alpha server 1000 running Open VMS operating system. The workstation program, called WinMOPS, written in Microsoft C++ resides in any PC running Microsoft Windows NT or Windows 95. All of the displays have a static background against which dynamic data is displayed. These graphs can display every kind of data that MOPS can receive as well as trend curves, tables, bar graphs, alarm lists and logicals (Figure 2.2).

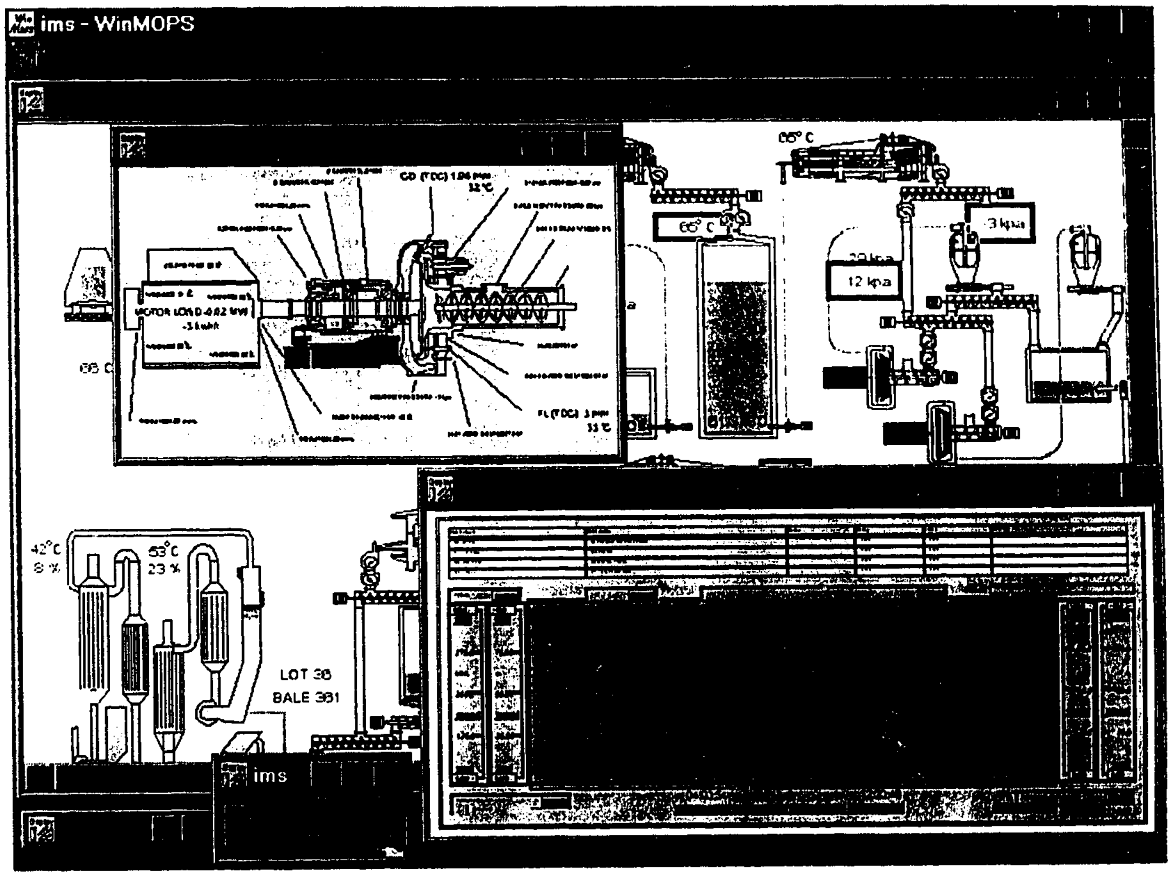


Figure 2.2. Sample MOPS screen

MOPS, Mill wide information and OPtimization System, collects data from pulp processes and stores it in database. As each value is entered into the database, the current date and time are stored with it. MOPS also provides tools for analyzing and viewing the stored data. Sources of Data include Distributed Control Systems (DCS), Programmable Logic Controllers (PLC), supervisory computers, laboratory instruments, and others. Data stored by MOPS include process measurements, targets, outputs, test results, equipment run/stop status, as well as any calculated values and logical states. Figure 2.3

shows how data flows from the process, through MOPS, to the users.

2.2.1.1 SOURCES OF DATA

MOPS collects the data in one of three ways: by capturing data from other systems, by accepting manual entry of data, and by calculating data from captured or manually-entered data.

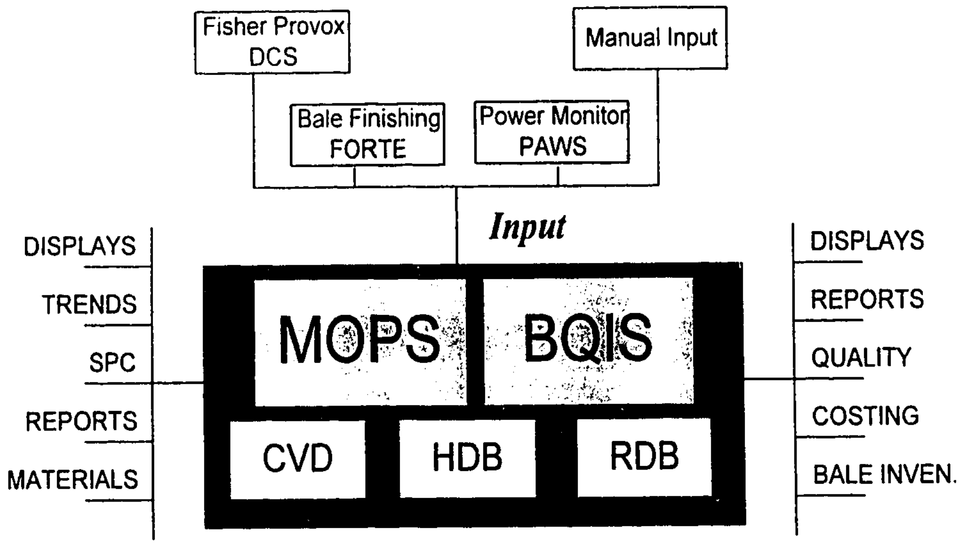


Figure 2.3. Architecture of Information Management System

1. Data is collected through links to other systems, which may be a real-time process control system such as a Distributed Control System (DCS), or an off-line system such as a laboratory analyzer with a built-in link to external computers. Usually, such data is transferred to MOPS automatically at regular intervals; however, the transfer may also be initiated manually when the data becomes available (for example, at the end of a laboratory test, or when the cost of a chemical changes). The important point is that the data already exists in the memory of another system, and MOPS gets a copy directly from there.

2. Manually entered data. Some data can't be captured directly from another information system, but must be keyed into MOPS manually. For example, the per-unit cost of a chemical may be available only from a printed catalog or an invoice. Other data may result from laboratory tests performed manually.

3. Calculated Data. Often, data that is essential to MOPS users can't be obtained anywhere in the required form, but it can be calculated from other data to which MOPS does have access. For example, the cost of production chemicals per tonne of pulp can be calculated from various flows, densities, temperatures, tank levels, composition analyses and suppliers' bulk prices. Similarly, the energy efficiency of a process can be determined from its input power.

2.2.1.2 *THE DATABASES*

Most of the mills in Alberta built recently have advanced digital control systems. These mills have already developed a standard for numbering the components of their control systems, normally called tag names. MOPS allows users to choose these tag names or External Point Names (EPN's) as an identification name to store data in the database.

A database is a collection of data organized in such a manner that the data can be retrieved easily. The MOPS database is divided into two sections: Current Value Database (CVD) and Historical Data-Base (HDB).

CVD contains only the most recent value for each point from which MOPS is collecting data. MOPS collects the data and stores it in this database. When a new value arrives, it replaces with the old value. The previous value gets stored to a predefined location in HDB using "trend loggers"

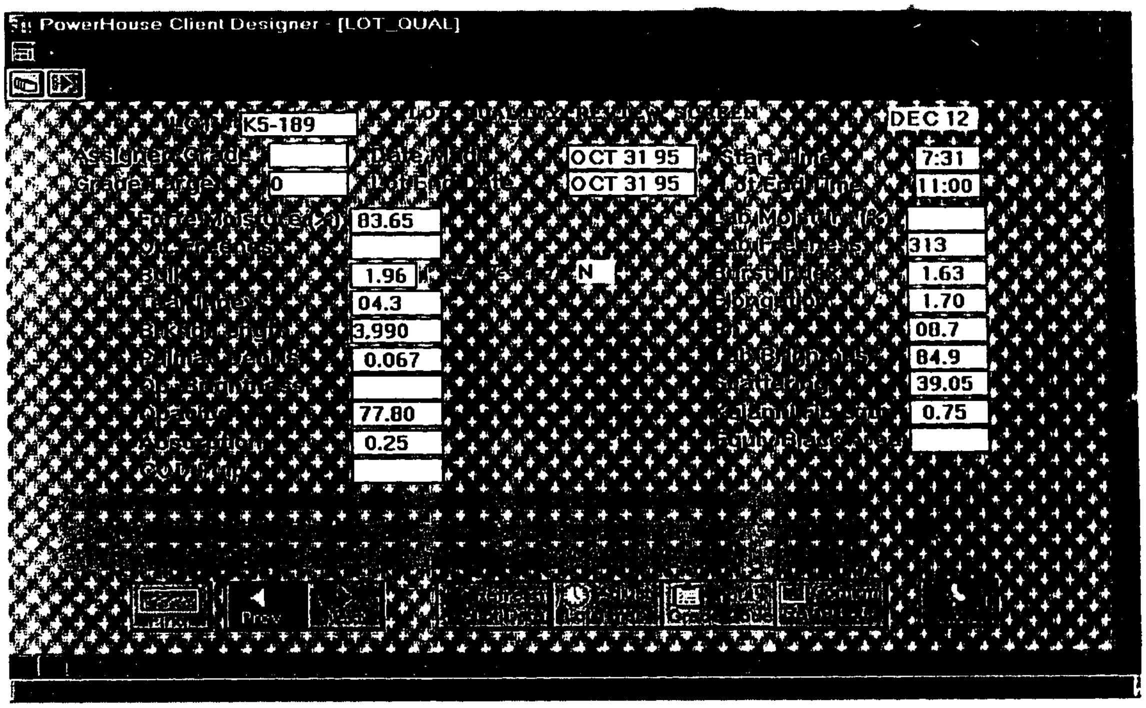
HDB contains all the previous values for every point for which MOPS is storing historical data. HDB uses a predefined logic to store various data. The HDB is divided into two major sections: synchronous and asynchronous. For points in the synchronous HDB, the trend logger stores one value whenever a predefined unit of time passes. This structure is not efficient since many points do not change for a long period of time. Therefore, to minimize the disk storage space, points are usually stored in the HDB asynchronously. MOPS uses a data compression technique to store only the values that would preserve its resolution as it ages. This compression is very simple, but effective, each point has a dead band or tolerance limit and if the value is within this limit, it stores the beginning value and its time and discards all the remaining values.

2.2.2 BQIS

Bale Quality Information System (BQIS) is being developed to provide a tool for employees at all levels of the plant to have direct access to information organized primarily on a per lot basis. It is developed using a 4GL tool from Cognos, called PowerHouse [Cognos, 1994]. BQIS consists of the following four modules (Figure 2.4).

- Production and quality management: Information on a completed bale on the finishing line is immediately available within the system with a "tentative" flag which needs to be approved and confirmed by a mill authority. This is normally done after all the lab test results becomes available after the lot is completed. This module then collects all the quality and costing data from MOPS and Forte and assigns to each lot. The system is capable of compensating for the rejected and off-grade bales.
- Inventory management: This module is used for assigning bales / lots into orders and attaching other cost such as transportation or remote warehouse cost.

- Order processing: Main function of this module is customer invoicing.
- Order management: This section includes modules for customers, order tracking, and shipping.


PowerHouse Client Designer - [LOT_QUAL]

LOT QUALITY REVIEW SCREEN DEC 12

Lot	K5-189				
Assigned Grade		Date Made	OCT 31 95	Start Time	7:31
Grade Target	0	Lot End Date	OCT 31 95	Lot End Time	11:00
[illegible] Moisture (%)	83.65			Lab Moisture (%)	
[illegible] Freeness				Lab Freeness	313
Bulk	1.96	[illegible]	N	Burst Index	1.63
Tear Index	04.3			Elongation	1.70
Breaking Length	3.990			[illegible]	08.7
[illegible]	0.067			Lab Brightness	84.9
[illegible]				Scattering	39.05
Opacity	77.80			[illegible]	0.75
Absorption	0.25			[illegible]	
[illegible]					

Figure 2.4. BQIS Lot quality screen

Chapter 3

3. SYSTEM ARCHITECTURE

This chapter presents the system design for the Intelligent Matrix Simulator (IMS). IMS is a software system integrated with several existing applications including MOPS and BQIS. IMS is an innovative system for graphical data simulation, knowledge exploration and analysis. It helps engineers and operators visualize and analyze data relationships so they can better understand and control the underlying processes. IMS is mainly implemented in C++ for the server running under the Open VMS operating system, while some of the user interface is written in Microsoft Visual Basic for client PCs running Windows 95.

IMS is a knowledge integration environment which consists of a symbolic reasoning system, numerical computation programs, database management system, computer graphics packages, and multimedia interfaces. Two supervisory units fulfill the selection, coordination, operation and communication of the independent software systems. This integrated software environment allows execution of programs written in different languages, and the application of multiple knowledge bases in the system. One of the main objectives of IMS is the integration with existing MOPS and BQIS systems [Farzadeh et. al., 1996] as shown in Figure 3.1.

The main function of MOPS is to collect data from various sources and to present them to users in a graphical form. The Bale Quality Information System (BQIS) captures data from the MOPS databases and the finishing line system (FORTE), and stores them in a relational database. It relates each bale of pulp to process operating conditions, warehouses, and customers. Using BQIS, a user can get the production and raw material cost of a given bale of pulp, and determine the warehouse location or the customer it has been shipped to. IMS simulates the process variables on-line to suggest the optimal operating conditions, considering the minimum consumption of chemicals, power and raw materials. Therefore, IMS complements the functions of MOPS and BQIS by giving predicted values to operators for achieving the on-grade bale of pulp. Table 3.1 presents the relations of the existing information systems with IMS.

Table 3.1 IMS relationship with MOPS and BQIS

	MOPS	*BQIS*	*IMS*
Purpose	Collect data and store in a time based relationship	Collect data and store in a Lot based relationship	Capture data from MOPS, BQIS, LOGBOOK and simulate the relationships between these data
Database	CVD, HDB	RDB	RDB, CVD, HDB
Knowledge	Process	MOPS, Sales and marketing, management	MOPS, BQIS, Process Engineers, Operators

From the systems designer point of view, IMS consists of three modules:

- Knowledge capture
- Data storage
- Simulation and Reporting

Knowledge acquisition and organization is the major bottleneck in intelligent system development. To capture and update the heuristic knowledge continuously, we have developed an on-line knowledge acquisition system called, Logbook. All the process operators, shift supervisors use Logbook to enter the newly encountered operating conditions of the mill into the system. A module in IMS scans these entries and then adds or updates its knowledge base.

Historical and current data of the process are stored respectively in HDB and CVD databases which were developed by MoDo Chemetics. CVD and HDB databases are optimized for the storage space and the access speed. All the BQIS data resides in a commercial database RDB, developed by Digital Equipment Corporation (DEC). RDB is based on an object-oriented model that stores and manages complex objects and their behavior. RDB provides standard based implementation of SQL (standard Query Language) for both data definition and data manipulation languages. CVD and HDB data can be extracted using application programming interfaces (API), however, these databases do not support Microsoft's open data base connectivity (ODBC) calls. Therefore, all the IMS data resides in RDB with full access to CVD and HDB databases.

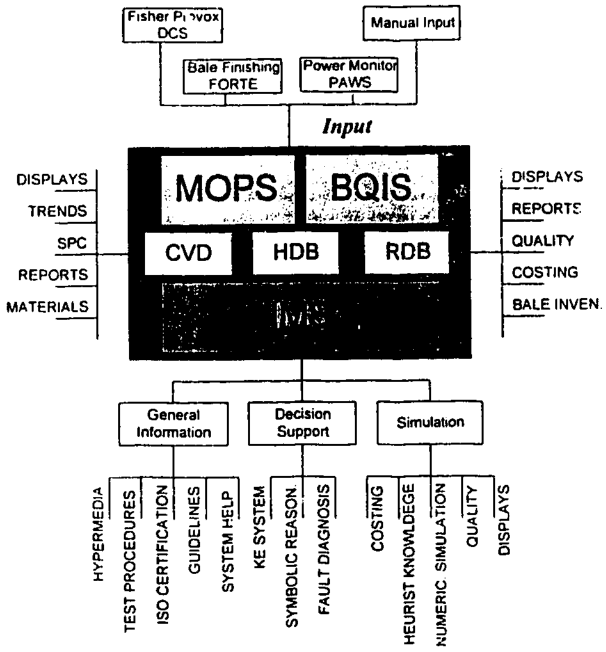


Figure 3.1. IMS integration with MOPS

The simulated values from IMS are presented to users by integrating the available commercial tools. Crystal, a Seagate software company, has developed an industry standard report writer called Crystal Reports [Seagate, 1995]. This report writer can access data from other databases, including financial systems and generate reports using ODBC drivers. These pre-formatted reports are then displayed within IMS by using dynamic link library (DLL) calls.

3.1 NETWORK ARCHITECTURE

The objective of this section is to show the data flows from process to systems and from systems to users. Figure 3.2 shows the hardware layout at SLPC. The mill

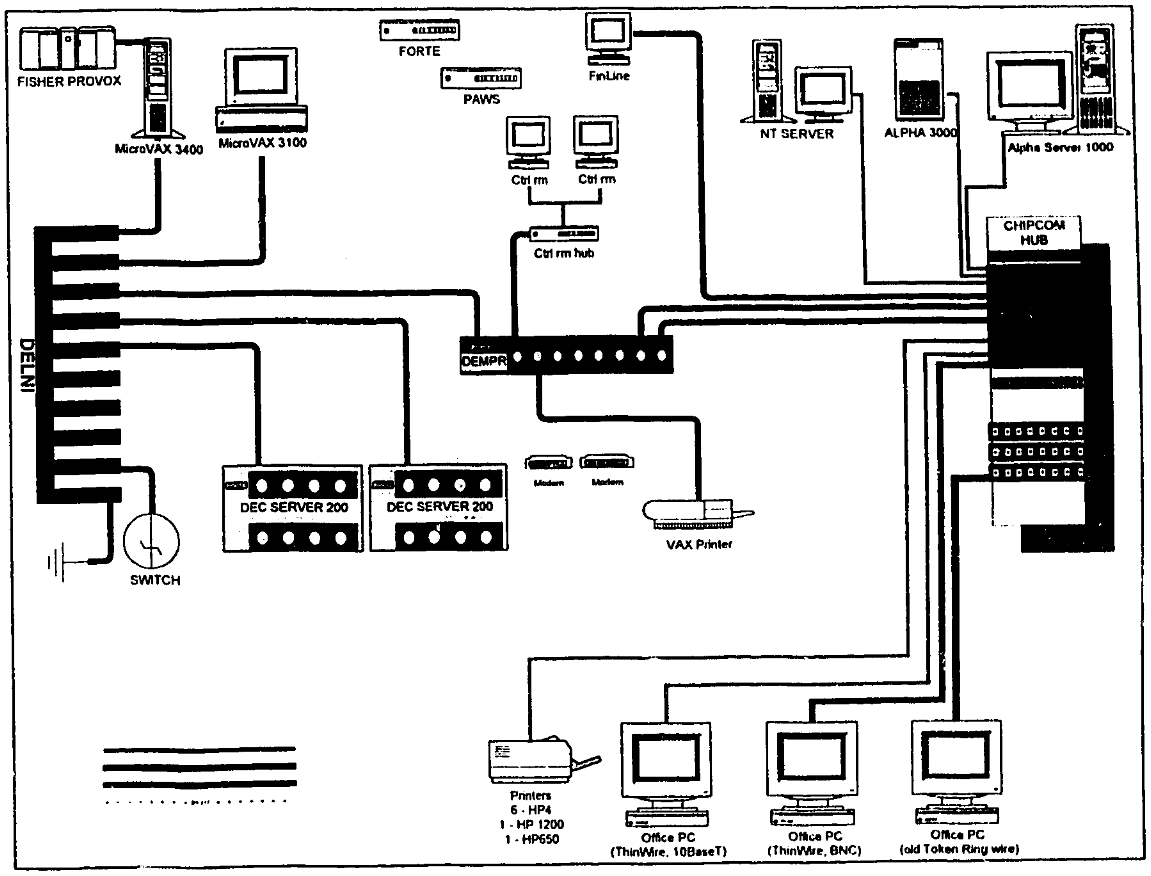


Figure 3.2. SLPC Hardware Layout

information systems collect data from four distinct resources: Fisher Provox DCS, Forte, Paws and manual input. The process instruments are controlled by Programmable Logic Controllers (PLC). Fisher Provox DCS first broadcasts the PLC data in its highway. All the computers on this highway have access to these data, including 3 operator consoles and a Micro VAX 3400. The MicroVAX 3400 is the buffer between DCS and the mill information systems. The MicroVAX 3400 multi-casts all the data to the Digital Ethernet Local Network Interconnect (DELNI) unit. DELNI is an electronic device which permits devices to communicate by using the Ethernet network transmission system. DELNI delivers these data to an Ethernet multiport repeater (DEMPR). The DEMPR in turn re-times, amplifies, and repeats all signals received on one coaxial cable segment and then passes the signal on to the bridge module of the Chipcom HUB. This bridge module is a high performance bridge that transparently interconnects off-site Ethernet networks to form a single extended LAN. The main function of this module is to isolate the local traffic through dynamic and permanent packet filtering from process control traffic. All the office computers, Alpha Workstation 3000, Alpha Server 1000 and PC file/print server are connected to Chipcom module.

The overall network is structured such that users have access to information from many different locations. The data traffic is controlled in the Chipcom HUB using a bridge between process data and office users. The majority of the process information comes from Fisher Provox Distributed Control System (process data), Forte (data from finishing line) and PAWS (Power monitoring system). Fisher Inc. has developed a VMS based hardware and software combination which makes DCS data available to external programs such as MOPS. MOPS stores these data in its CVD and HDB database. BQIS uses a data manager (developed by MoDo Chemetics) to store bale quality information in Oracle RDB™. Forte data gets stored in CVD and RDB using ForteLink routine via an RS-232 line. The core of the IMS system resides on a Digital Alpha 3000 workstation running Open VMS version 1.5. IMS uses two different routines to communicate with users; some of the user interactions are done through MOPS user program routines while most of the communication is achieved using a C++ routine, called IMS server. IMS server uses TCP/IP in an Ethernet environment to achieve interaction with its other counter part in PCs, called IMS Client. Mill Information Systems (MOPS, BQIS) and all their databases reside on a Digital Alpha Server 1000, running Open VMS 6.1. These two Alpha machines communicate with each other using Decnet. All of the hard disks are accessible to both Alpha's using VAX Cluster software. The Microsoft Windows NT Server is used as a file/print server. All the programs relating to IMS, MOPS, BQIS on the PC side reside on this server, thus, it is possible to restrict access to these files to authorized personnel.

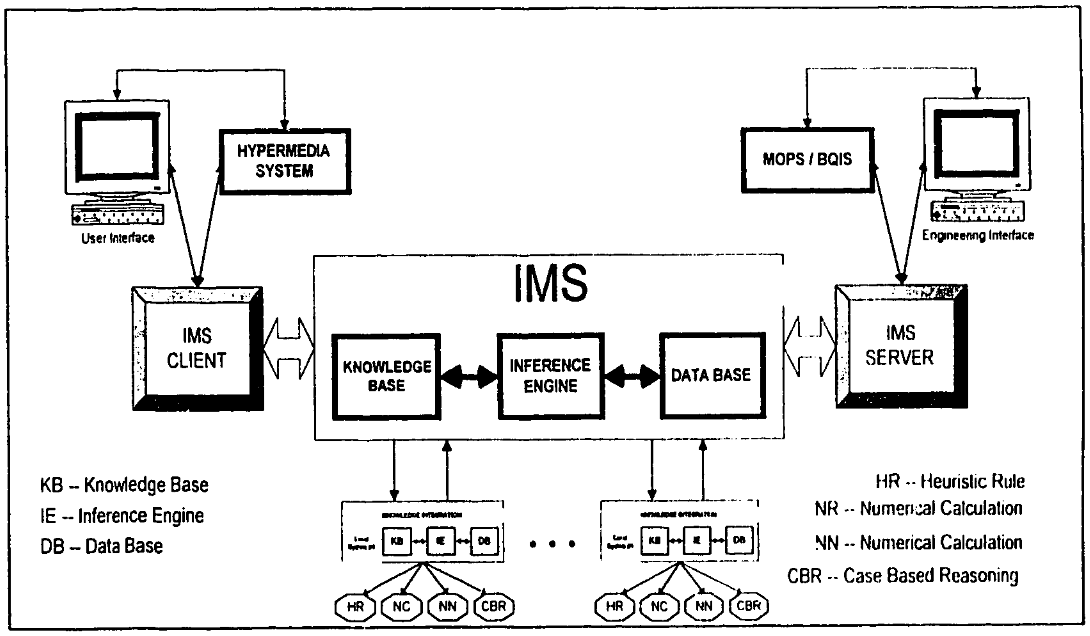


Figure 3.3. System architecture of IMS

The Forte and Paws systems do not support ethernet networks, but using a DEC Server 200, these data are converted from RS-232 cables to Ethernet and then transported to Alpha Server 1000. The FORTE (PC based hardware and software combination) bale moisture measuring system performs on-line measurements. As the bale is moved through the conveyor system, the moisture and weight are read by the system. These data are transferred to the Alpha Server 1000 and stored in RDB, CVD, HDB and Forte's propriety database. PAWS is a system to control the energy usage in the main equipment of the mill, such as refiners. Thus, operators have three combined means of managing the process (DCS, Forte, and PAWS). All the process control and instrument set points need to be controlled by these three separate computer systems. However, there are only two operators to monitor these equipment. Therefore, it is necessary to develop an on-line mill information system to enhance the decision making for the process operators.

3.2 IMS ARCHITECTURE

Process simulation for operation support is a complex task. Some knowledge is available in the form of strict mathematical models. Some is available only in heuristic production rule form. Some other is hidden in the vast amount of process data. Figure 3.3 represents the system architecture of IMS. It can be seen that the global system is divided into a number of local systems in terms of system functions and/or process decomposition. Each local system integrates case based reasoning (CBR), numerical calculation (NC), heuristic rules (HR), and neural networks (NN) to solve individual problems. The real-time relational database (DB), knowledge-base (KB) and inference engine (IE) modules are the key components in the system. The real-time database stores the most important data required for simulation. The knowledge base stores numerical models and heuristic rules. The inference engine module provides the general inference mechanism, such as backward chaining control strategy (goal-driven) and forward chaining (data-driven). The IMS server handles all the calls from external programs and redirects them to local systems. The IMS server also enables these local systems to request data from external systems such as BQIS. On the workstation side, the IMS client handles the user request and sends messages to the IMS server. For example, when a user requests a simulation. THE IMS client sends all the [illegible] ection variables to the IMS server. Then, the IMS server uses a predefined query to call an SQL routine to extract all the actual process outputs from previous operations, this operation is called case-based reasoning (CBR). The IMS server stores all the matching results from CBR in its memory. Concurrently, inference engine collects all the facts and rules for each object from the knowledge base and passes them to the IMS server. IMS evaluates all the collected cases, rules, numerical calculations, models and sends the best results to the IMS client.

3.3 CASE BASED REASONING

Case based reasoning (CBR) is used in IMS to learn how to focus on problems and narrow them down to likely hypotheses and simulations in a way that is similar to the human experts' reasoning during operation.

Case-based reasoning is a problem solving paradigm that in many respects is fundamentally different from other major AI approaches. Instead of relying solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the specific knowledge of previously experienced, specific problem situations (cases). A new problem is solved by finding a similar past case, and reusing it in the new problem situation. A second important feature is that CBR also is an approach to incremental, sustained learning, since the new experience is retained each time when a problem has been solved, making it immediately available for future problems. A very important feature of case-based reasoning is its coupling to learning. The driving force behind case-based methods has to a large extent come from the machine learning community, and case-based reasoning is also regarded a sub-field of machine learning. Thus, the notion of case-based reasoning does not only denote a particular reasoning method, irrespective of how the cases are acquired, it also denotes a machine learning paradigm that enables sustained learning by updating the case base after a problem has been solved. Learning in CBR occurs as a natural by-product of problem solving. When a problem is successfully solved, the experience is retained in order to solve similar problems in the future. When an attempt to solve a problem fails, the reason for the failure is identified and remembered in order to avoid the same mistake in the future.

Case-based reasoning favors learning from experience, since it is usually easier to learn by retaining the concrete problem solving experience than to generalize from it. Still, effective learning in CBR requires a well worked out set of methods in order to extract relevant knowledge from the experience, integrate a case into an existing knowledge structure, and index the case for later matching with similar cases.

3.3.1 CASE BASED REASONING IN IMS

The primary problem for a CBR system is determining what past situations are "similar" to the current case. The relevant past solutions need to be organized in the database so that the descriptions of input problems can be used to retrieve them. The relevance is often determined not by the obvious features of the input problem, but by abstract relationships between features. IMS uses an indexing algorithm to retrieve relevant solutions from the case base, given new action variables from the user. At the highest level of generality, an IMS based CBR cycle is described by the following five processes:

1) RETRIEVE the most similar case or cases
2) REUSE the information and knowledge in that case

3) EVALUATE the new case using other intelligent reasoning modules

4) REVISE the proposed result variables based on the evaluation

5) RETAIN the parts of this experience likely to be useful for future problem solving and store in the IMS

A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing knowledge-base (case-base). The five processes each involve a number of more specific steps, which will be described in the task model (Figure 3.4 illustrates this cycle).

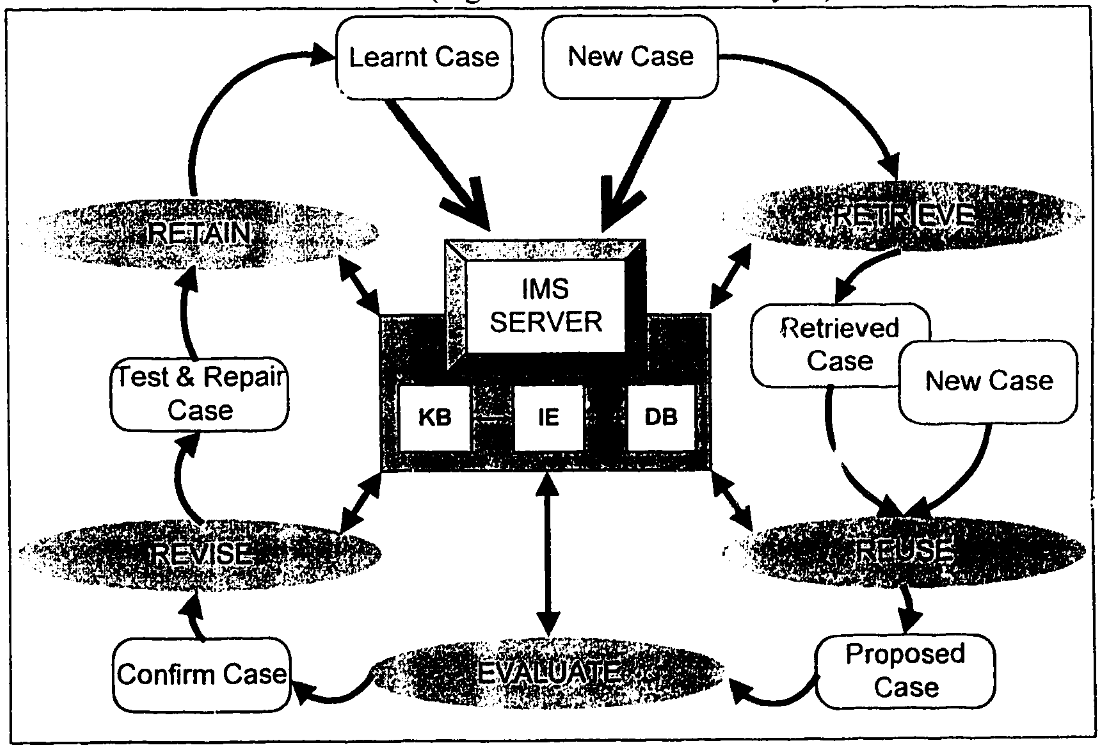


Figure 3.4. The CBR cycle

An IMS server request for simulation defines a new case. This new case is used to

RETRIEVE a case from the collection of previous cases. A generic algorithm within each object's KB calls a predefined routine for retrieving previous case(s). At this point, the algorithm has all the action variables of the IMS. Querying the database for all the constraints and then generating an SQL statement makes this algorithm very fast and efficient for finding the previous case or cases. The retrieved case is combined with the new case - through REUSE - into a proposed case. A proposed solution to the initial problem which gets EVALUATED against simulated results from neural network model, heuristic rules and numerical calculations. At this point, IMS has all the case values and each object calls the routines for simulating the object. Through the REVISE process, this solution is tested for success by applying to the process environment and evaluating against the actual result variables as well as repairing if failed. During RETAIN, useful experience is retained for future reuse, and the case base is updated by a new learned case, or by modification of some existing cases.

3.4 ARTIFICIAL NEURAL NETWORKS

An Artificial Neural Network (ANN) is used in IMS to simulate the action variables against result variables of individual objects. Neural networks can analyze large uantities of data and establish patterns and characteristics in situations where rules are ›t known. Neural networks can deal with the incomplete and noisy data. These capabilities have proven too difficult for traditional symbolic reasoning systems. ANN have been defined [Kohonen, 1988] as follows: "Artificial neural networks are massively parallel interconnected networks of simple elements and their hierarchical organizations which are intended to interact with the objects of the real world in the same way as biological nervous systems do."

The idea of building an artificial neural network was originated from an attempt to model the biophysiology of human brain. Artificial neural networks are composed of highly interconnected simple processing elements in parallel. Due to its structural and functional resemblance to biological neural networks, an ANN exhibits a number of characteristics of the human brain. ANN's, like human brain, can learn from the past experience by modifying its behavior in response to its new environment, can generalize from the previous examples to new ones as a result of its structure and abstract essential characteristics of a set of inputs containing related data. The specific characteristics of an ANN are a result of the network paradigm utilized. This paradigm is specified by the network architecture and the neurodynamics.

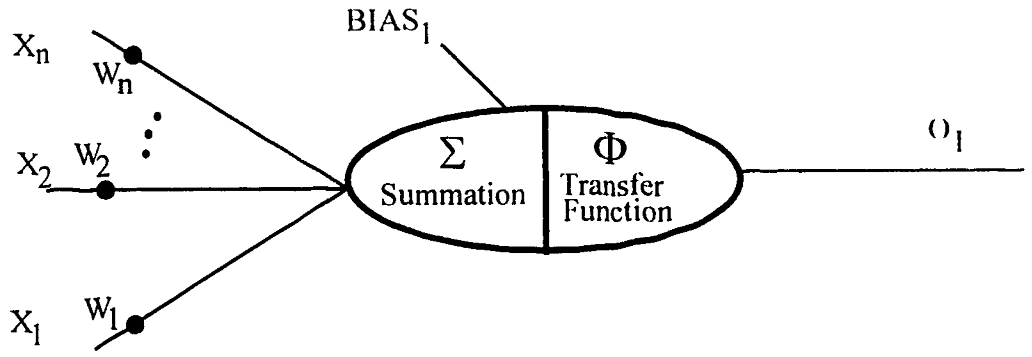


Figure 3.5. Artificial neuron with activation function

The network architecture defines the arrangement of processing elements and their interconnections. It defines what neurons or processing elements are interconnected as shown in Figure 3.5. The interconnection scheme specifies how inputs from and outputs to the processing elements are arranged as well as what the information flow direction will be. Whereas, the neurodynamics specify how the inputs to the neurons are going to be combined together and what type of function is going to be used to develop the output, as well as defining how the adaptive elements or weights are going to be modified. There are two basic types of neural networks [Caudill, 1990]: supervised and unsupervised:

> Supervised networks build models which classify patterns, make predictions, or make decisions according to other patterns of inputs and outputs they have "learned." They give the most reasonable answer based upon the variety of learned patterns. In a supervised network, you show the network how to make predictions, classifications, or decisions by giving it a large number of correct classifications or predictions from which it can learn.
>
> Unsupervised networks can classify a set of training patterns into a specified number of categories without being shown in advance how to categorize. The network achieves this by clustering patterns. It clusters them by their proximity in N dimensional space where N is the number of inputs. The user tells the network the maximum number of categories and it usually clusters the data into that number of categories. However, occasionally the network may not be able to separate the patterns into distinct categories.

Neither type of networks is guaranteed to always give an absolutely "correct" answer, especially if patterns are in some way incomplete or conflicting. In this regard, the technology is similar to biological neural functioning after it was designed, and differs significantly from all other conventional computer software. Neural networks may not work at all with some applications. Some problems are well suited for the pattern recognition capabilities of a neural network and others are best solved with other methods.

The spectrum of different paradigms is extensive. The computational features exhibited by a network are dependent on the paradigm developed.

3.4.1 THE BACKPROPOGATION PARADIGM

The network used in this research project is the popular backpropogation algorithm (supervised network). The backpropogation algorithm [Rumelhart and McClelland, 1986] learns adequate internal representations using deterministic units to provide a mapping from input to output. This procedure involves the calculation of a set of output vectors **O** using the current weights **W** (a set composed of all matrixes $\mathbf{W}_m$, where m=2 ... i, Where $\mathbf{W}_2$ would be the matrix of weights between the input and the first hidden layer and $\mathbf{W}_i$ the matrix of weights between the last hidden layer and the output layer) and the input vectors **X**. The error is estimated by comparing **O** with the target vector **T** and using an error function. This error function is defined for a specific $\mathbf{X}_p$ and $\mathbf{T}_p$ as follows:

$$E_p = \frac{1}{2}\sum_i (t_i - o_{il})^2 \qquad (3\text{-}1)$$

where the index **p** represents an input vector target output relationship that conforms the input vector set **T**, **i** represents the output nodes of the output layer in the network, and **l** is the total number of the layers. $\mathbf{t_i}$ is the targeted output for the **i**th output node and $\mathbf{o_{il}}$ is the response obtained from the **i**th output node using the corresponding $\mathbf{I_p}$. The learning procedure minimizes $\mathbf{E_p}$ by performing steepest descent and therefore obtaining an appropriate **W**.

The net input to a neuron is: $net_{im} = \sum_{l} \mathrm{w}_{ijm} o_{jm-1} + \Theta_{im}$ (3-2)

where $\mathbf{w_{ijm}}$ represents the weight between the **j**th unit of layer **m-1** and the **i**th unit of layer **m**. $\Theta_{\mathbf{im}}$ represent the bias for the *i*th unit of layer **m**. The activation function utilized is the logistic function given by: $o_{im} = \frac{1}{(1+e^{-net_{im}})}$. (3-3)

The overall objective of this algorithm is to minimize the error, $\mathbf{E_p}$, and to achieve a convenient **W**. Hence, it is necessary to make adjustments to previous **W** obtained until the error tolerance imposed by the final desired mapping accuracy is achieved. By establishing:

$$\Delta w_{ijm} \approx \frac{\partial E_p}{\partial w_{ijm}} \quad \text{and} \qquad (3\text{-}4)$$

$$o_{jm-1} = \frac{\partial net_{im}}{\partial w_{ijm}} \qquad (3\text{-}5)$$

$$o_{im}(1-o_{im}) = \frac{\partial o_{im}}{\partial net_{im}} \qquad (3\text{-}6)$$

then the partial derivative of Ep with respect to the weights could be expressed as:

$$\frac{\partial E_p}{\partial w_{ijm}} = \left(\frac{\partial E_p}{\partial net_{im}}\right)\left(\frac{\partial net_{im}}{\partial w_{ijm}}\right) \qquad (3\text{-}7)$$

and

$$\frac{\partial E_p}{\partial net_{im}} = -\Omega_{im} \qquad (3\text{-}8)$$

The variable Ω is calculated by backpropogating the error through the network starting with the output layer where the partial derivative of the error to the output is defined as:

$$\frac{\partial E_p}{\partial o_{il}} = -(t_i - o_{il}) \qquad (3\text{-}9)$$

the output layer Ω_{il} is $\Omega_{il} = (t_i - o_{il})o_{il}(1 - o_{il})$ (3-10)

the adjustments are $\Delta w_{ijl} = \lambda \Omega_{il} o_{jl-1}$ (3-11)

where λ is the learning rate or the step size of the steepest decent.

For the lower layers, Ω can be expressed as follows:

$$\Omega_{im} = \sum_j (\Omega_{jm+1} w_{jim+1}) o_{im}(1 - o_{im}) \quad (3\text{-}12)$$

the adjustments r Δw_{ijm} is: $\Delta w_{ijm} = \lambda \Omega_{im} o_{jm-1}$ (3-13)

The standar st decent methods tend to converge very slowly. A number of more efficient learning algorithms have been developed based on the steepest decent approach. In this thesis, we have employed one of the often used heuristics to speed up the convergence. This common heuristics utilizes the momentum factor, β, that weights the contribution of the past ΔW. Therefore the updating is modified as follows:

$$w_{ijm}(t) = w_{ijm}(t-1) + \beta \Delta w_{ijm}(t-1) \quad (3\text{-}14)$$

The availability of many commercial neural networks allow us to choose a neural network system which is compatible with our project objectives. After evaluating many different NN softwares, we chose Neuroshell from Ward Systems [1994]. This system is functionally very rich and offers a routine to integrate with IMS system.

3.4.2 NEUROSHELL

Neuroshell is a commercial ANN software developed by Ward Systems [Ward, 1994]. It offers several main menu options which enable users to develop on-line applications. Figure 3.6 shows the main menu of the module. The order of operations is to work from left to right. Developers may not have to use all of the icons that appear on a screen in order to create a working neural network module for IMS.

After collecting all the data, and using a filter module to validate them (Pre network module), the developer chooses options from the main menu of NeuroShell to train the network (Build and training modules). Figure 3.7 shows the network architecture with multiple hidden slabs and different activation functions in Neuroshell system. After training the network, contribution factors allow fine tuning of the input variables. A contribution factor is a relative measure of the importance of a specific action variable on predicting the network's output. The higher the number, the more the variable is contributing to the prediction.

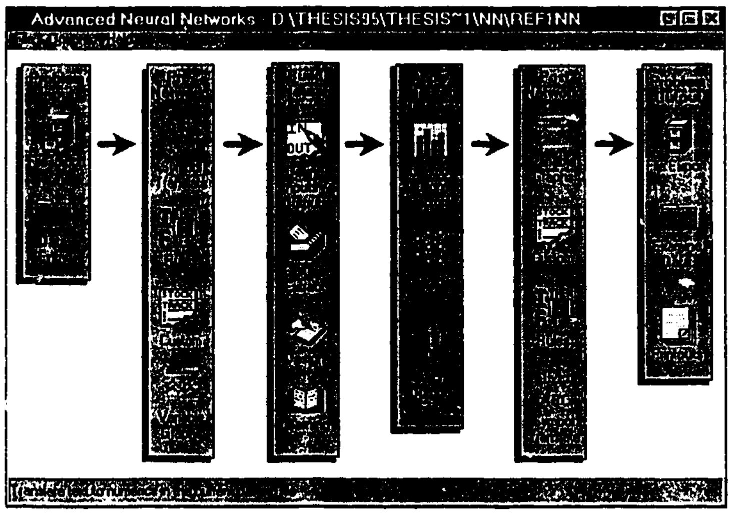


Figure 3.6. Main features of Neuroshell

A pre-processing module is used to obtain the training data set by filtering the noisy and invalid data. The developer uses the Rules module to post-process the network's predictions. When creating rules in NeuroShell to process the data, the basic rule structure is shown as the following:

IF *(possibly followed by And, Or)*

THEN *(possibly followed by And)*

ELSE *(possibly followed by And)*

Using production rules of the module, we can normalize and reorganize the data to give a better distribution. The variable graph module allow developer to graph one or more of the network's variables in many different ways. When putting data into the network, creating a scatter plot of variables insures that the training patterns are representative of the entire problem domain.

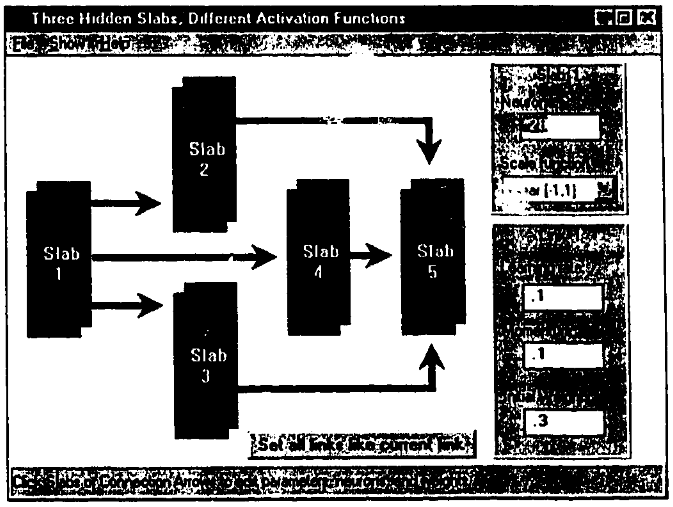

Three Hidden Slabs, Different Activation Functions
Slab 1
Slab 2
Slab 3
Slab 4
Slab 5
[-1,1]
.1
.1
.3
Set all links like current link

Figure 3.7. Sample network paradigm in Neuroshell

Application of Neuroshell in this project requires some customization to overcome the following deficiencies:

- Process variables are well tuned in the desired values and have little variations and poor distributions. Neural networks can not be trained properly based on these data only.
- There are the significant time delays between action and result variables.
- Neuroshell is a stand alone package that needs to be integrated with other IMS modules and to use the data in MOPS and BQIS.
- Evaluating the network performance with Neuroshell is limited to root mean

square (RMS) error only.

To solve these problems, an integrated environment is developed for data pre-processing, NN model training, post-processing and evaluation based on Neuroshell (as shown in Figure 3.8). This environment is integrated to IMS for data accessing and knowledge sharing with other IMS modules.

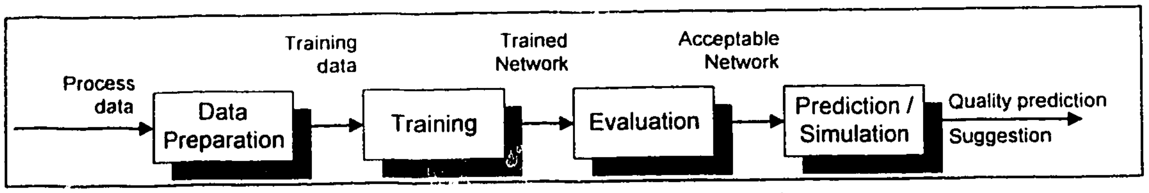


Figure 3.8. Neuroshell implementation

3.5 EXPERT SYSTEMS

An expert system is used to emulate the expert in a defined field. Expert systems are capable of symbolic processing, inferencing using heuristic rules, and explaining. Representation of the knowledge of experts lends itself well to production rules in the form "IF premise 'A', THEN conclusion 'B'". Another form of knowledge representation is the frame structure. A frame is a data structure with fields, or slots, describing the properties of the component or rule represented by the frame. Slots may contain information such as a variable name, value, or type, as well as any other pertinent information relevant to the object described in the frame. Slots may contain information about the relationship of a frame to other frames.

IMS uses a frame based intelligent system developed by the Intelligence Engineering Laboratory at University of Alberta. It is a C-based expert system, called Meta-COOP [Rao, 1991].

3.5.1 META-COOP

Meta-COOP is a development environment for knowledge systems that use object-oriented programming [Rao, Wang and Cha, 1992]. The basic object within the Meta-COOP environment is called the ***Unit*** (also called ***Frame*** or ***Object***). All items, concepts, and abstractions of the problem area are represented as units. Each unit has an arbitrary number of "slots," in which the attributes of the unit are described. Each slot represents one attribute of the unit and has several "facets," in which the attribute is specified in more detail. The Figure 3.9 illustrates the relationships between unit, slots, and facets.

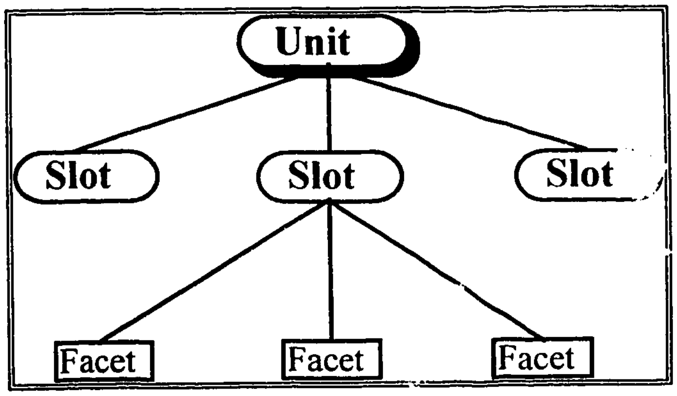


Figure 3.9. Relationships between unit, slots, and facets.

Each slot has its own attributes, which are called facets. Facets can control the specific means by which inheritance is implemented for that particular slot. In some cases, local values override inherited values, and in other cases local values are appended to inherited values. A slot can also contain other facets which can be used to specify, for example, the value range for the slot value, or whether multiple values are permitted. The number of possible values in slot is called Cardinality. The facet Cardinality.Min specifies the lower limit, Cardinality.Max specify the upper limit.

There are two types of units: class units and member units. Member units describe the individual objects. On the other hand, class units group several objects with common attributes into a single class. Therefore, member units are the instances of the class unit. They are specific examples of the general concepts expressed by class units. Class units can be arranged in a hierarchy, i.e., one class can be defined as a subclass or superclass of another.

The arrangement of classes in a hierarchy, in which the more general classes are the superclasses of more narrowly defined classes, is important for the mechanism called ***inheritance***. Inheritance means that the slots in a class unit are automatically copied to its lower class and its member units. Therefore, a knowledge base can be built very effectively when several objects in a problem area have the same attributes. On the other hand, there is no point of inheriting an attribute which is valid only for the class unit for which it was defined. The inheritance of this slot by subclasses or instances are prevented by designating it as an ***Ownslot***. In contrast, inheritable slots are called ***Memberslots.*** Therefore, a member unit contains only Ownslots, since member units, by definition, can not have subclasses or instances. The rules have the following format:

```
RULE  <number>
              FACT    <condition>
                AND  <condition>
                OR    <condition>
              THEN   <conclusion>
```

When representing the <conditions> within *facts*, "AND" and "OR" can be used together or separately or not used at all. Since rules in Meta-COOP are nothing more than member units, the rules can also be grouped by means of class formation. Thus, it is possible to segment the rule base of a large expert system, and activate only the group of rules which are needed at the moment. This enhances the speed of execution for the expert system and helps to keep knowledge bases manageable. The basic operations of an application are performed when an instance or unit receives a message. Also, using the message sending facility, Meta-COOP communicates with its knowledge base by the following syntax:

send_message("*KEYWORD*", "*INSTANCE OF A UNIT*");

Communication among different units (and between Meta-COOP and the knowledge base) is accomplished by a special slot which has a value *METHOD*. When a message is sent to an instance, the system searches to find the handler which is a unit or instance that defines the method that corresponds to the message being sent. Then, the method gets executed and any result gets returned to the object which had sent the message. A method can be defined as either a procedure or a rule. If the method represents heuristic knowledge, then usually rule gets used, however, if a method represents strategic knowledge, then a procedure should be used.

The original version of Meta-COOP was designed for off-line expert system applications in a standalone host. It is executed only in UNIX and DOS environments. To satisfy the on-line real-time requirements of this project, have modified the system. The main modification is to deal with the reasoning speed and the memory management. Also, we have added many new functions to integrate it into the IMS system. These added functions include modules to retrieve data from CVD, HDB and RDB. Meta-COOP is a server based expert system building shell that communicates with external world via IMS SERVER.

3.6 HYBRID INTELLIGENT SYSTEM ARCHITECTURE

The diversity and complementary strength of advanced computing technologies, such as case-based reasoning, neural networks, expert systems suggest numerous and fruitful integration. Researchers have attempted to combine these systems to a hybrid system in a number of ways [Myers, 1990]. A hybrid architecture allows these independent systems to communicate through messages while expert system knows most of the internal workings of the other systems.

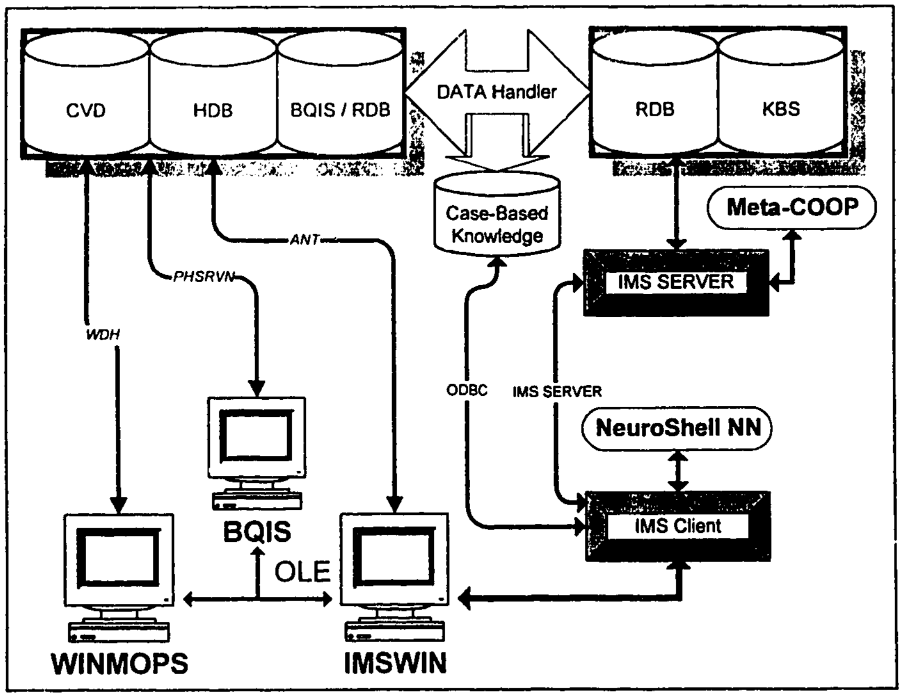


Figure 3.10. Hybrid architecture of IMS

As a hybrid intelligent system, IMS has three different knowledge bases and problem solving techniques: Case-bases reasoning (CBR) for previous operating conditions, neural network (NN) models and object based rules. These three knowledge bases have different accuracy and complement each other. NN solution is not necessarily accurate and requires a second opinion from CBR or production rules. Since CBR cases are based on previous similar operating conditions, the suggested solution from CBR is considered more accurate. However, not all the possible simulation scenarios are available in CBR. In this case, we have the production rules to validate NN results.

IMS system first obtains requested data and objects from user interface and gets corresponding data from MOPS and BQIS through a system interface (as shown in Figure 3.10). These data are passed through the CBR cycle and the closest match to the case at hand is passed to the inference engine. The inference engine retrieves all the rules related to the object (process object such as REF1) and evaluates them with the CBR cases. If there are any other successful solutions from rules, they are compared to the retrieved case from CBR. If the solutions from rules are within tolerable level of CBR, then the rule based solution is passed to the user as the answer. Otherwise, IMS requests the NN model to simulate for the given object. The NN model's solution is compared to CBR case and numerical model (if there is one). The best solution is sent to the operator and all the modules update their learning modules to update their knowledge.

3.7 Interface with Mill Information Systems

Interfacing IMS with existing systems is the fundamental requirement for the success of this project. First of all, the data from MOPS and BQIS must be available for IMS. Secondly the common user interface is achieved by integrating these systems, which increases the acceptance of IMS among its users.

3.7.1 IMS Interface with MOPS

MOPS data is supplied to the WinMOPS program via a connection to a MOPS Server. The key features of WinMOPS [Chemetics, 1995] are:

- Provide multiple windows where each window will display a picture defined to be a combination of static and basic dynamic data. Windows can be tiled, maximized or minimized.

- Static and dynamic data types provided are those currently in MOPS (line, circle, arc, rectangle, bargraph, trend, text).
- Static data attributes provided are color, scale, font and size, dashed lines and background color. All objects can be controlled from MOPS database such as x/y position, size etc.
- Users can create, save, load and modify a basic display on a network PC file server.
- Users can request or search document (picture, trend) from library by using index and name. Multiple selections can be used to search on area and category.
- Support of a user environment for each user containing such items as default library area, colors, directories.
- Support for TCP/IP communication protocol.
- OLE 2.0 object support
- External data handler (API) on Server side provides an application framework to speed development and provide 'consistency' in user program applications.

Using MOPS user program modules in Open VMS environment, have developed routines within IMS to access data from the MOPS Current Value Database (CVD) and Historical Data Base (HDB). These routines enable IMS to receive/send data from/to WinMOPS clients. The IMS server is executed as a detached process in the Alpha 3000 and gets activated when there is a call from its clients within MOPS.

By adding two function routines to Meta-COOP, have implemented IMS communication with the Alpha 3000 computer. The first function is called "getdata", which is responsible for getting requested data from the MOPS database. The second function is called "putdata", which is responsible for putting the requested data to the MOPS database. These are the steps taken to embed the IMS into the MOPS:

1. IMS Server sleeps until it is requested by the user, then the *matdsp* routine gets activated, following with a pseudo-code of behavior.
2. MOPS displays the static part of the display and returns the control to IMS by executing *usinit* function.
3. IMS gets all the required information from MOPS and BQIS database with *usgepn* and *SQL* routines.
4. IMS does its inferencing (CBR, NN, NC, HR), then uses *usput* to put the values in MOPS database and informs MOPS with *uscomp* routine.
5. MOPS sends the dynamic part of the IMS to PC.
6. IMS waits for a new user request using *usgtcm* command until user closes IMS screen. If the screen is closed then go to step 1, else go to step 2.

3.7.2 IMS INTERFACE WITH FORTE

The FORTE bale moisture measuring system performs on-line measurements. As the bale is moved through the conveyor system, the moisture and weight are read by the system. The parameters for the selected grade are used in performing the calculations for the current bale when it reaches the bale marking station. When the current lot is closed, a lot report is produced which is a listing of all the bales in chronological order followed by lot totals.

We have installed a dedicated RS232 line between the PC based finishing line system (FORTE) and the Alpha computers and developed a program to translate the received lines of text into their respective tags. A definition file is used to send the data into CVD, HDB, and RDB. This file consists of a combination of TABLE, INS, and MAILBOX statements as explained in the the following:

TABLE "name",tabletype,"id","trigger"

- "name" is the table name, up to 16 characters. For identification purposes only.
- The tabletype- integer value, not presently used.
- period- number of seconds allowed between stores of trend data. This parameter specifies to the link that even if the number of trend data arrived is not equal to the number to be buffered, the data is to be trended any ways. This is to prevent loss of data in the event of link stoppage.
- id string from FORTE to identify the table, "B" for bale report, "L" for lot report, etc.

trigger- a MOPS CVD reference. This value is retrieved from the CVD every time the table is processed. If its value has changed since the last scan, the link will go through the table looking for all INS with a trend flag of ASYNCHRONOUS (value 1) and trend them using PUTATR.

INS "epn", trendtype, trendsize, startchar, numchars, instype

- epn- any valid MOPS tag. The value of this field is stored to the EPN in CVD when table is processed.
- trendtype- 1 = asynchronous, 2 = synchronous, trended every time a value is received.
- trendsize- number of trend values to buffer before storing.
- startchar- start character of this field within the FORTE report.
- numchars- length of the INS field in FORTE record.
- instype- identifies the datatype of the INS.

MAILBOX "mbxname",size,nummessages

Specifies a mailbox to send data of the preceding INS to.

- "mbxname"- valid VMS mailbox name.
- size- number of bytes per mailbox message.
- nummessages- maximum number of messages this mailbox can store.

When this link receives the following line: *B11/16/9010:23:100121001222187*, it causes the following values to be stored in the database:

TIME = "11/16/9410:23:10
GRADE-PV = 12
LOT-PV = 1001
BALENO-PV = 222
BALEGW-PV= 187

Using MOPS material tracking system, we attach process operating conditions such as chemical / energy consumption and action variables setpoints to each bale and store them together in HDB and RDB. The RDB database contains a cost table for all the important raw materials. A unit within IMS calculates the total consumption for producing an individual bale of pulp and attaches it to the respective bale.

3.7.3 IMS INTERFACE WITH BQIS

The objective of this module is to ensure that all the necessary data gets stored into both databases. In many cases such as lab test results, the data is not immediately available and needs to be manually entered into the system. In such cases, we must implement an interface to acquire lab test results into the system and to link it to the corresponding finished product.

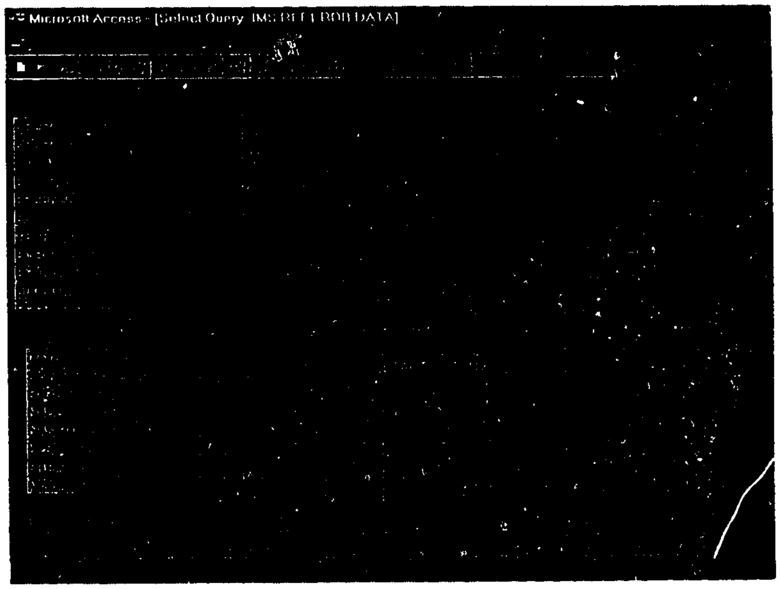


Figure 3.11. Table organization of BQIS / IMS in RDB

Meta-system frames handle all the data interaction between MOPS, BQIS and Matrix Simulator modules. A back propagation neural network from Neuroshell systems is used for generation of heuristic simulation results for prediction of the process variables with no known rules or mathematical formulas. NeuroShell network extracts the real time on-line data from MOPS historical database (HDB) or its own relational database which resides in Oracle RDB using Microsoft ODBC drivers.

Chapter 4

4. KNOWLEDGE INTEGRATION

The modular design of IMS is based on the main steps in the application of intelligent systems technology to real time pulp process operations. It was originally stimulated by the needs of integrating current mill information systems with knowledge intensive systems. The software design is divided into the unique areas of the mill based on personnel expertise. There are experts in the wood harvesting and hauling, chipping, refining, bleaching, and finishing line. They work relatively independently of each other with a common goal to produce the highest quality pulp with the lowest possible cost. One of the main sources for interdepartmental communication is MOPS and DCS. They extract the needed information from MOPS for smooth operation of the mill. However, MOPS function is to collect and store data in its databases. Process operators use MOPS trending and graphics modules to visually correlate the data. MOPS can display up to 8 tags in the same trend and has the capability of shifting each individual tag forward or backward in time. This MOPS function is used for visually extracting the relationships between process tags, but the user must know the time difference among trended tags. The time it takes for pulp to travel from one piece of equipment to another in the process varies significantly, depending on the process operating conditions. such as production rate and desired grade. Therefore, to use MOPS effectively, a user must have extensive knowledge of the process. Another alternative would be to capture the process knowledge from the different operation departments and integrate it with MOPS.

IMS has the capability to simulate system behavior under various conditions, such as grades and production ranges. It is designed as depicted in Figure 4.1.

An IMS object is described in terms of result and action variables. All process variables are classified into these two groups. The action variables are usually the operating conditions such as temperatures and chemical adding rates. The result variables are often the quality variables and those measuring the operation optimality.

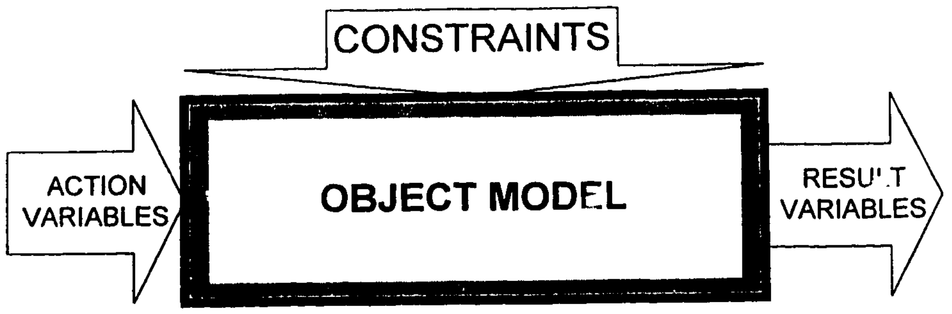


Figure 4.1. IMS Object

The IMS knowledge base is organized using Meta-COOP frames. The process is divided into smaller sub-processes such that each sub-process can be represented in one frame. Each one of these sub-processes has its action variables fed from foregoing sub-process's result variables. In order for IMS to select the correct relations for different situations, operating conditions are defined as the constraints to knowledge. The operating conditions include grades and production ranges. A grade defines the final quality of a particular product. The production range is used to refine the change of operating conditions under the same grade.

To meet these objectives with a high degree of consistency, knowledge is grouped into the following three categories and incorporated into the representation scheme of each frame.

1) knowledge about process behavior, i.e., models and data which describe the process;

2) knowledge about relationship between processes, such as the relation between

washing and bleaching stages;

3) problem solving knowledge.

4.1 KNOWLEDGE REPRESENTATION

Using the features of the Meta-COOP, have built the object oriented knowledge representation describing all the properties, states, and behavior of the SLPC process. The process have been separated into a number of modules such as refining, bleaching and drying as shown in Figure 4.2. This separation is based on the material flow and hardware layout of the process.

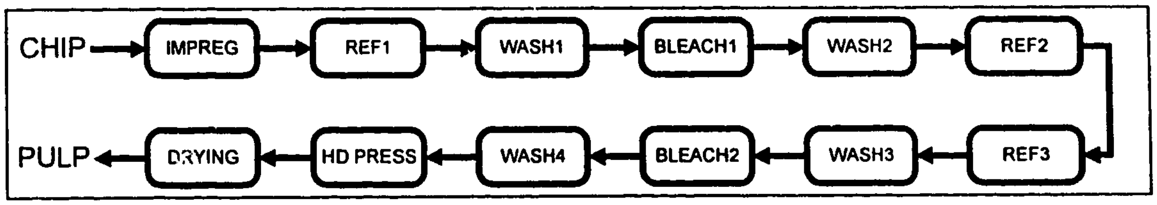


Figure 4.2: Sequential process representation

Each one of the modules are further separated into nodes to reflect the actual operation of the mill. Figure 4.3 shows only a simplified hierarchical model of the inter stage bleaching plant (BLEACH1). Since each node in the hierarchy is a physical component, the behavior model for each node is developed and connected into the whole process model. There are many pieces of equipment which are used in more than one location at the process. We have created a library of objects containing their common behaviors and use the object cloning feature of Meta-COOP. To understand the approach, consider the impregnation stage, as shown in Figure 4.3. There are 3 APS bins, 3 discharge conveyors, and 3 pumps which are functionally identical to each other. The same kinds of devices are also used in other stages of the mill, there are over 60 pumps, and 20 discharge conveyors used at different stages of the process. Therefore, developing an object model for each class of equipment has a very fast payback in the system.

The knowledge used to represent the process characteristics is stored using Meta-COOP frames. These frames are organized in a hierarchical structure according to the process decomposition shown in Figures 4.2 and 4.4. For example, a frame is designed to represent the knowledge for BLEACH1. Under this frame, there is a number of sub-frames (subclasses) to describe the various nodes in BLEACH1, such as TRANSFER CHEST. Each frame contains a number of member slots to describe the properties of these nodes. The member slots in the above representation update their values from MOPS database. In order to simulate the process dynamics on-line, we have made the extensive use of the material tracking features of MOPS.

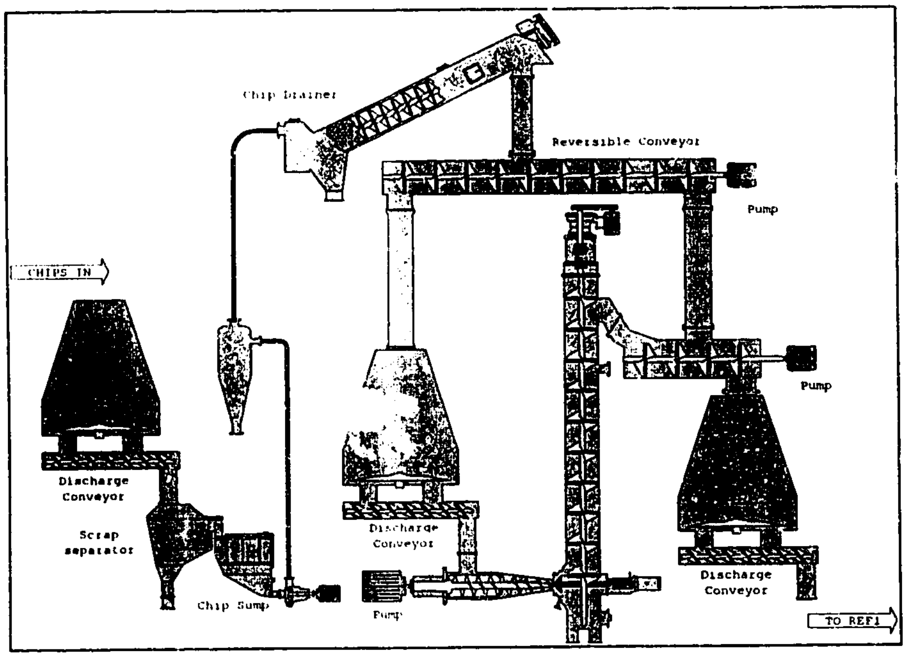


Figure 4.3: Process layout at Impregnation module

All the relevant process values in each stage get time stamped. Figure 4.5 shows the amount of time (retention time) for a given bale of pulp to pass through an equipment or travel from one location of the process to another. Retention time depends on many factors such as the production rates, tank levels, consistency and so on. In our system, we use an approximate formula to calculate the retention time based on the current production rate, nominal production rate, and equipment capacity with the assumption that the operating conditions would be close to the nominal conditions. For some equipment, the calculations have been enhanced by incorporating the pulp consistency. Here is a formula for retention time calculation of APS bin #1, where 611LC102 is the percentage level of the bin, PC01 is the BDMT production rate per day, 355 and 30 are the nominal production rate and retention time under nominal conditions respectively

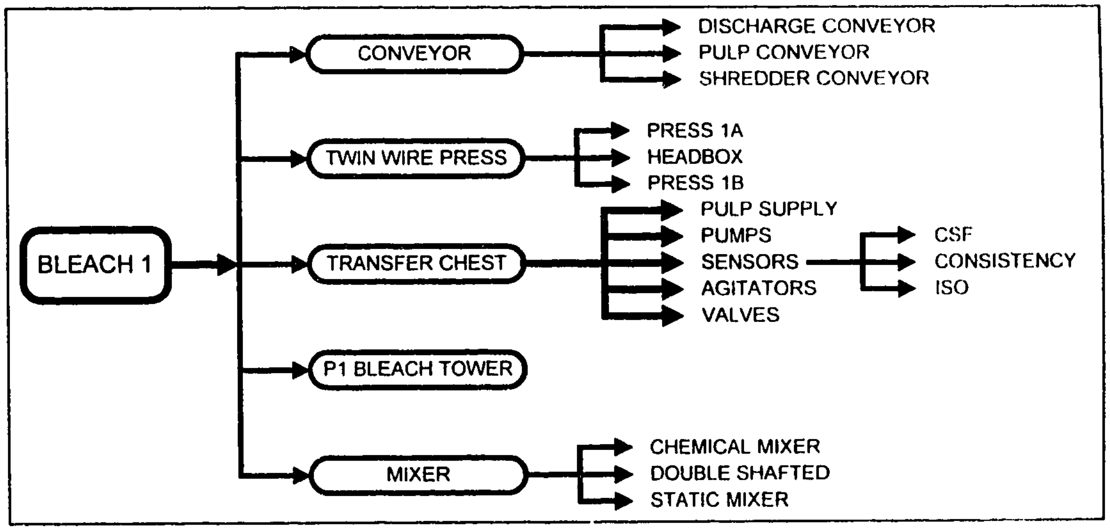
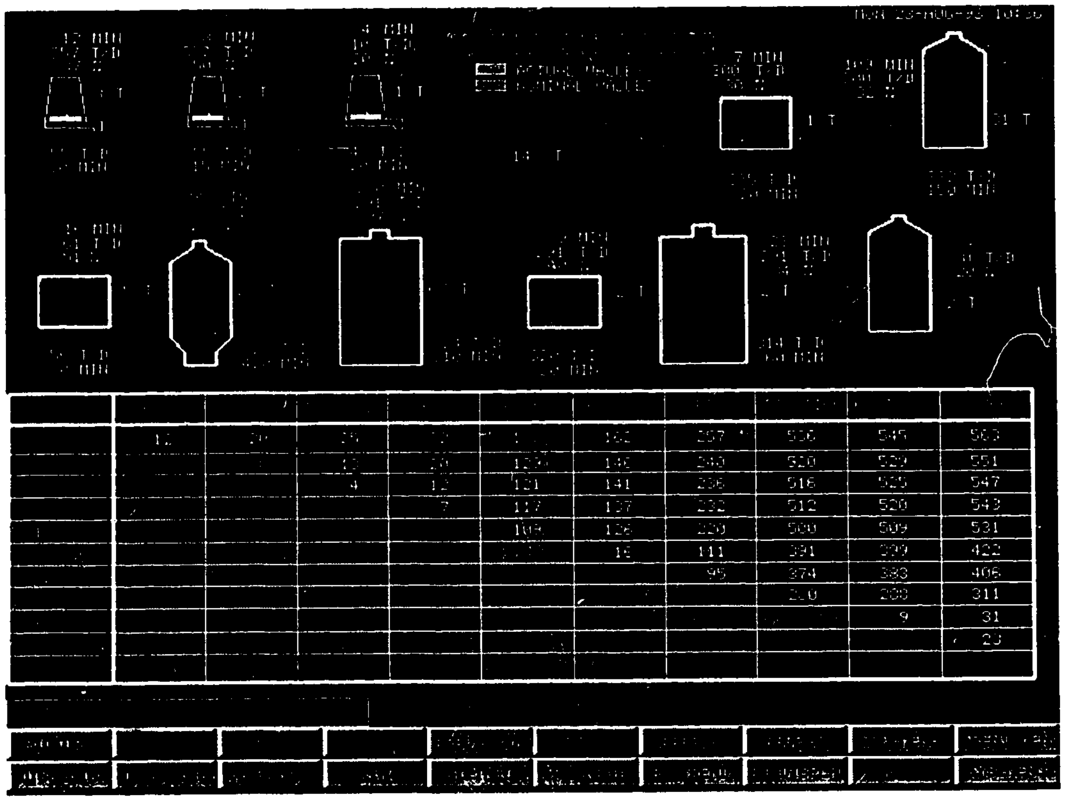


Figure 4.4: Inter-stage bleaching representation in IMS

$$APS1 \cong \frac{611LC102}{PC01} \times \frac{355}{100} \times 30 \qquad \text{minutes} \qquad (4.1)$$

Since all the important equipment have an object or unit representation within our system, we can simulate any of the "what if" type of scenarios within that equipment and pass the simulated results to the next object. For example, if we want to know the consequences of an increase in primary refiner power to the final quality of the pulp, then we get a snap shot of the operating conditions at the primary refiner at that time; simulate the effect of the specific energy increase in that stage; pass on the simulated results to next object with its retention time using messaging feature of the Meta-COOP. As an example of message passing and its simulation procedures, suppose a message tells the system that the press 1A's status has been changed. As soon as the member slot value of the press 1A is placed in the knowledge base, the system activates a method that finds all influenced objects, beginning from the lowest level of the hierarchy, and starts the simulation. The outcomes of such a message are therefore available to the reasoning process, increasing the speed of simulation.



	12	20	28	30	[illegible]	162	257	536	545	563
		3	13	20	125	146	240	520	529	551
			4	12	121	141	236	516	525	547
				7	117	137	232	512	520	543
					108	128	220	500	509	531
					[illegible]	15	111	391	399	422
							95	374	383	406
								210	288	311
									9	31
										23

Figure 4.5. Material tracking overview

This object oriented approach provides us with the capability for system extensibility. For example, if there is a change in process hardware, in the knowledge representation only the affected nodes need to be modified. This has proven very useful since SLPC have been going through a de-bottlenecking project and we have incorporated this into the knowledge structure already. Also, this hierarchical decomposition approach allows us to simulate the process models very quickly since we only need to activate the models or rules of only affected sub-processes. We do not need to waste time on searching other sub-hierarchies at the same level. It is the main reason why we can improve the simulation efficiency by using this structure model.

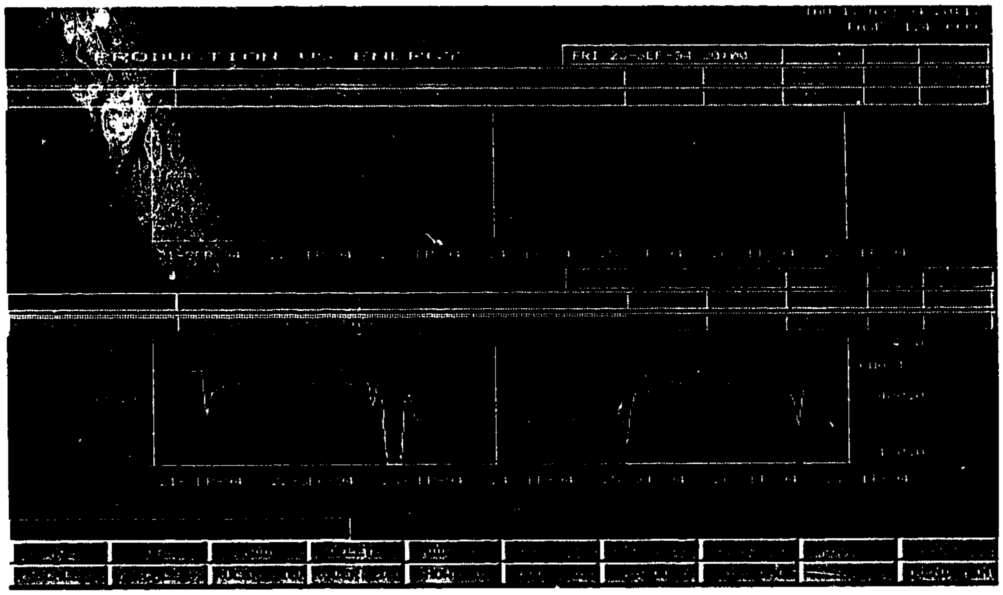
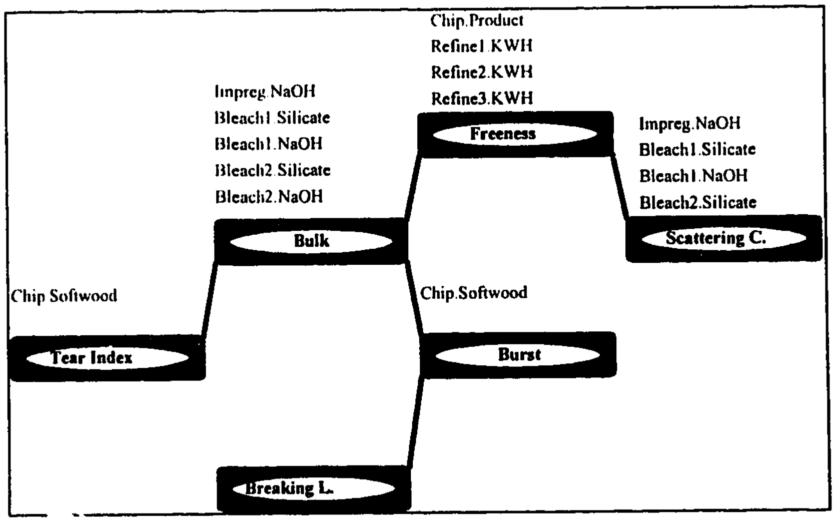


Figure 4.6: Relationship between production, grade and energy in refiners

Conventional rule based programs try to find correct solutions with the help of a system's observed behavior, whereas our model-based simulation system evaluates events based on the system's observed and predicted behavior. A problem solving model is a specific translation from component's physical description to a set of rules.

4.2 PROBLEM SOLVING MODELS

SLPC currently produces about 25 unique grades of pulp, which range from very high to low brightness pulp. Typically, there is a grade change every two days. Operation of the process is very much dependent on the grades and production rates. As shown in Figure 4.6, the operation of the secondary refiner and the tertiary refiner depends on the grades, and operating conditions of primary refiner such as the age of the plate. If the produced grade requires a lower freeness, then all three refiners must operate in order to achieve the desired quality. Otherwise, one or even both of the secondary and tertiary refiners can be bypassed. Figure 4.6 shows a grade change values where the secondary refiner is bypassed. The retention time, rejection rate, and chemical charge also play very important roles. The remainder of this chapter shows how knowledge is incorporated in IMS.



[illegible]gure 4.7: Relationship between result and action variables

Each fra[illegible] is associated with a table in the RDB which contains all the process data for that fram[illegible] These tables are used by CBR for retrieving the previous cases, NN for training and predicting, and production rules for getting the current values and constraints of the process in that frame. The primary refiner object (REF1) is the simplest frame in terms of process dependencies, hence has fewer fields in its table. These process values are stored along with equipment retention times, MOPS material tracked values from previous frames, and production time. Storing material tracked values allows us to consider the dynamic behaviour of the process.

As indicated, process variables are divided into two types: action variables and result variables. Figure 4.7 is a simplified representation of the model structure in terms of action variables, result variables and their corresponding relationship between the frames for re~ult variables. For example, the scattering coefficient is directly affected by the chemical addition rates at the bleaching and impregnation stages. It is also dependent on result variables and, freeness at the three refiners. The freeness at each refiner is affected by quality such as tree age, outside temperature and percentage of chip mixture (softwood to hardwood ratio). In this way, we connect all the relevant frames together for simulation.

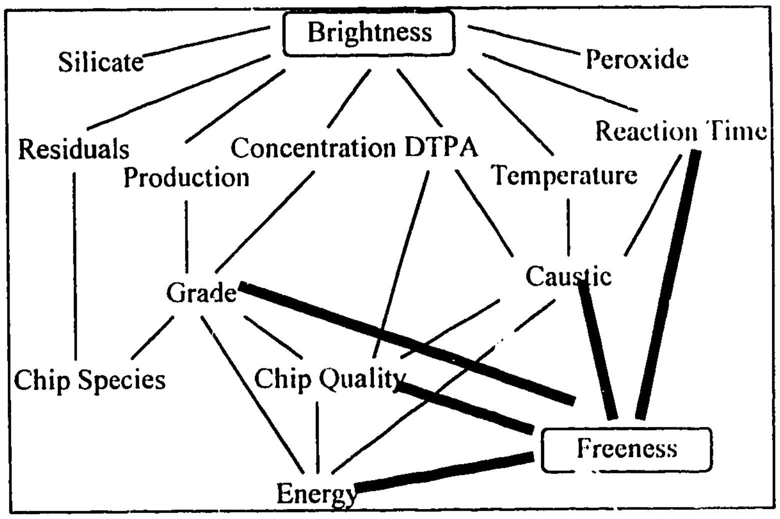


Figure 4.8: Brightness and freeness dependency relations

In a BCTMP mill, brightness and freeness are the most influential result variables in terms of mill efficiency, production, and cost. Many of the other result variables such as bulk, tear index are based on these two variables. Figure 4.8 shows a relationship between brightness and freeness and their corresponding action variables. Freeness at each frame is interdependent on brightness and vice versa by having the common action variables.

We have obtained the mathematical models for the brightness and freeness calculation. The following sections will introduce these models and show how they are implemented in IMS.

4.2.1 BRIGHTNESS CALCULATION

In a BCTMP mill, the bleaching process is the most important stage for achieving on-grade pulp. Most of the chromophores are eliminated in the two stages of bleaching. Many different action and result variables in the bleaching process have a direct impact on chromophore content. The brightness is dependent on fiber surface condition and color content. Mathematically it can be represented by the following equation:

$$Brightness = 1 + \frac{k}{s} - \sqrt{\left(\frac{k^2}{s^2} + 2\frac{k}{s}\right)} \qquad (4.2)$$

Where s is specific light scattering coefficient describing the fiber surface. It is a variable describing the ability of the fiber to reflect incoming light. k is specific light absorption coefficient describing the fiber chromophore content. It is a variable describing the color content in the fiber. During the peroxide bleaching the chromophores are eliminated and peroxide is consumed. The empirical equation describing the k calculation is :

$$k = EXP\left[LN(k_h - k_0) - A\left(\frac{CPX}{B} + 1\right)\right] + k_0 \qquad (4.3)$$

Where k_b is the amount of chromophore in the raw material coming into process;

k_0 is the amount of chromophore that can not be removed during bleaching; CPX is the consumed peroxide in the process; A and B are the correction factors. Normally, the more peroxide that is added, the higher the brightness would be. However, if too much peroxide is added to the process, it has a negative effect on brightness by reacting to other chemicals such as DTPA, this consumption is modeled by *B*. Factors determining the amount of peroxide consumed in each tower are:

- alkali amount charged: more alkali makes chemicals react faster
- reaction time: longer time increases consumption
- reaction temperature: an increase in temperature speeds up reaction
- pulp consistency: higher pulp consistency means less dilution and more concentrated chemicals

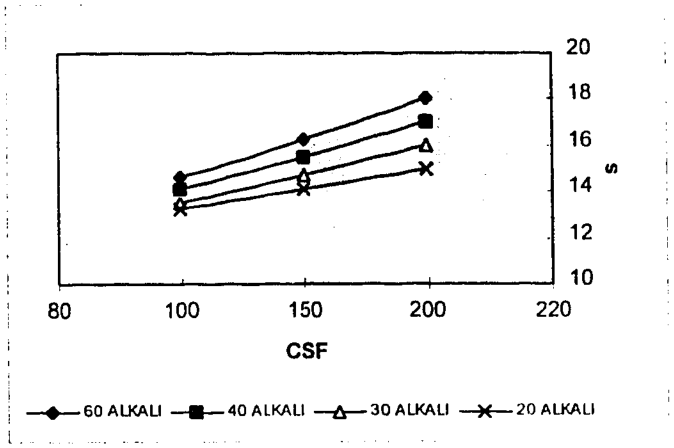


Figure 4.9: Relationship between s and the freeness

The specific light scattering coefficient, *s* is a number that describes the ability of the fiber (sheet) surface to scatter (reflect) incoming light. This coefficient is mainly related to the raw material such as wood specie, freeness (fines material) and alkali amount added in the process. The relationship between s and the freeness and alkali is depicted in Figure 4.9. We have developed a number of rules to represent the knowledge contained in Figure 4.9 and have implemented them in the IMS *s* calculation.

4.2.2 Freeness Calculation

The following relationship has been obtained for freeness calculation, which is valid at all three stages of the refining process:

$$csf = EXP(K_1 - K_2 * KWH) - K_4 * (CAUSTIC - K_5) + K_3$$

Where KWH is the specific energy consumption at the refiner; CAUSTIC is the amount of caustic added to the pulp before entering the refiner; K_3 is dependent on the incoming freeness to the refiner; K_1, K_2, K_4, K_5 are dependent on production rate, grade, chip quality, concentration, plate age, and plate gap. Figure 4.10 shows the dependency of freeness on KWH and caustic under normal operating conditions.

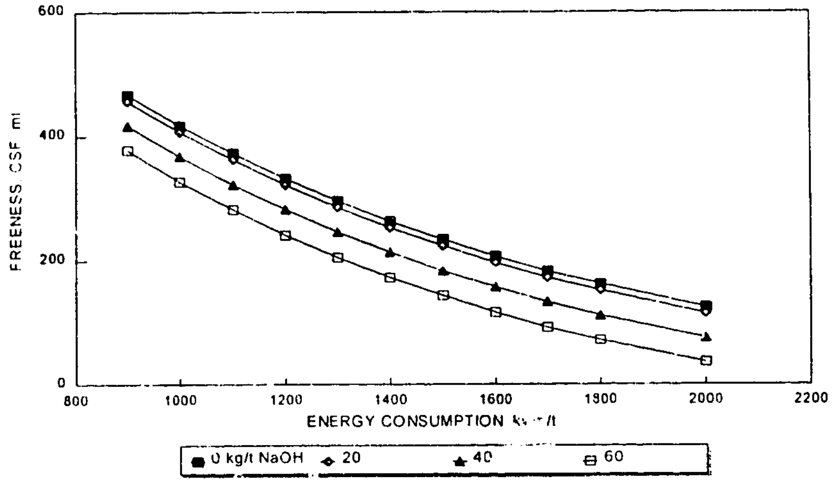


Figure 4.10: Specific Energy relationship to freeness

The calculation of the coefficients K_1, K_2, K_4 and K_5 under various operating conditions has been formulated into heuristic rules and implemented into IMS.

4.3 Neural Network Model

Because of the complexity of the pulp process, not all the result variables have mathematical formulations available. For those available mathematical formulations, there exist a few unknown constant coefficients that depend on many factors such as operating conditions. These factors are usually calculated using the experimental data obtained under normal operating conditions. When the operating conditions change from the nominal situation, the mathematical formulation accuracy may degrade. Considering this problem, we have also developed neural network models for a few important quality variables.

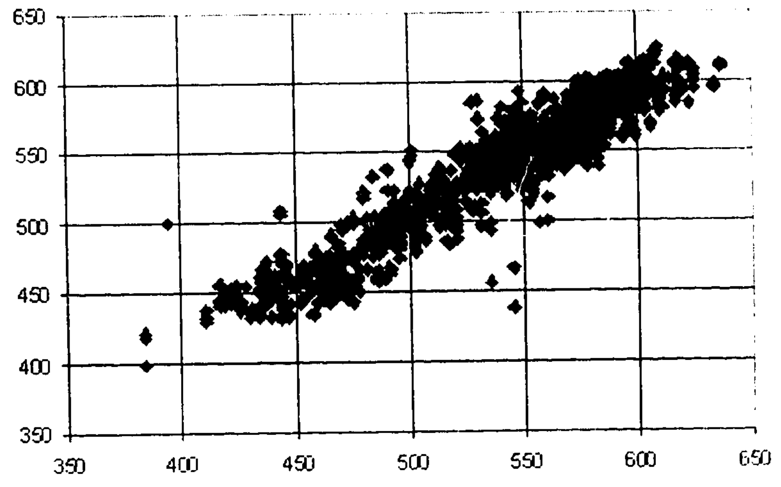


Figure 4.11: Scattering plot of actual versus predicted freeness

Figure 4.11 Shows the prediction accuracy of the freeness at REF1 using the neural network models. For this neural network model, we have used 28 action variables that influence the freeness. We selected 3200 data for the above 28 tags (ignoring all the other quality requirements) and trained the system It can be seen that the neural network model gives very accurate predictions. Figure 4.12 presents the accuracy of the brightness model at inter-stage bleaching. The accuracy is also satisfactory.

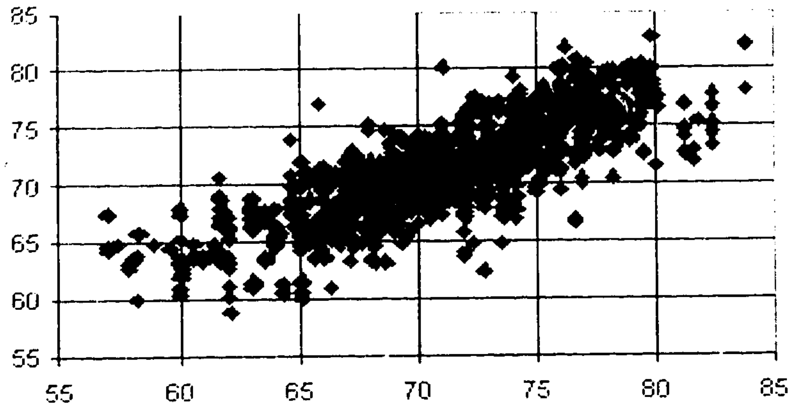


Figure 4.12: Scattering plot of actual versus predicted brightness

The use of neural network models for process simulation is illustrated by Figure 4.13. First of all, IMS collects all the relevant data from the MOPS CVD. IMS has heuristic knowledge to find out the list of the action variables for the frame. According to the MOPS material tracking, then, IMS reorganizes the data set to incorporate the retention time. Constraints, such as the pulp grade and the production range are used to select a specific neural network model from the available models. The reorganized data set is then applied to the selected neural network model and the simulation for the current frame is executed. A message, along with the predicted result variable values, is sent to the frame in the downstream (for example from REF1 to WASH1). As this continues, the message is finally sent to the last frame required for the simulation and this gives the final simulation results.

As presented above, IMS applies knowledge with various techniques including heuristic rules, mathematical models, past production cases and the neural network models. The heuristic rules are mostly used for models selection and evaluation. For some frames, there is only one model available. Whereas for other frames, there may be two or more models available for the same result variables. This raises a potential problem of selecting the right (most accurate) model for simulation.

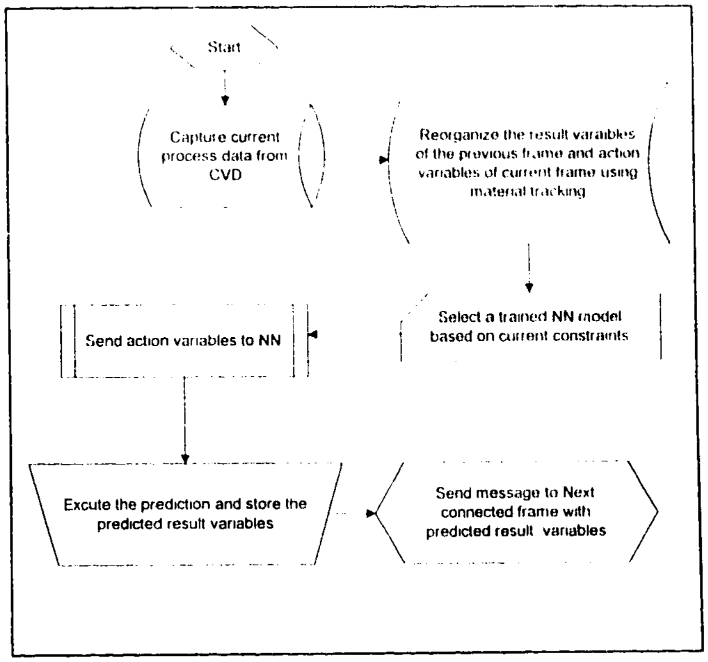

Start
Capture current process data from CVD
Reorganize the result varaibles of the previous frame and action variables of current frame using material tracking
Select a trained NN model based on current constraints
Send action variables to NN
Excute the prediction and store the predicted result variables
Send message to Next connected frame with predicted result variables

Figure 4.13: Flowchart of NN simulation

In IMS, different forms of models are implemented in different ways. For example, mathematical models are installed in the Meta-COOP as shown in Table 4.1. Meta-COOP is capable of implementing [illegible] mathematical calculations in the CONCLUSION portion of the production rules and in the METHOD. It can automatically call the action variables from other frames. This implementation is simple and efficient.

Neural network models are far more complex and are thus not easy to implement in Meta-COOP. We thus treat each neural network model as an external DLL call. Whenever an NN model is required for simulation, Meta-COOP sends a message to activate this method in its corresponding frame. Each frame contains a member slot with a valueclass of METHOD which has all the information needed to activate its corresponding NN model.

The past production cases are stored in a RDB table. Each table represents the process knowledge in terms of action variables and result variables. If the values of the action variables of a new situation matches a past case in the RDB table, the values of the result variable is assumed to be the simulation result. This knowledge is readily available in the mill production databases.

Table 4.1: Rule-based model

```
Memberslot: burst_tear_rule from FINAL
  Valueclass: RULES
  Values: { rule 403
    if WASH_4.CSF > 85  and  FINAL.BULK > 1.0
    then FINAL.BURST:=3.0/FINAL.BULK + 0.05*CHIP.SOFTWOOD/100;
        and  FINAL.TEAR:=2.2*FINAL.BULK + CS/100 };
```

The reasoning process for solving a simulation problem can be summarized as follows: as soon as the simulation problem is defined, IMS identifies the first affected frame and checks the RDB table of that frame for a result. If a past case in the table matches the current situation, the result variables will be retrieved as the simulation results for the frame; then a method in the current frame uses the heuristic rules to find the next frame for simulation and sends a message along with the obtained results. If no past cases are matched, IMS checks the availability of other simulation models, such as mathematical models or NN models. If only one model is available, IMS executes the model and gets the simulation results as well as sends a message to the next frame. If two models are available, both of them are triggered and the results from both models are compared and evaluated using a method in Meta-COOP. The better one will be taken as the simulation result and passed on to the next frame. As indicated above, this simulation-message sending process is continued until the final frame is reached and the results from the final frame is the simulation result we require.

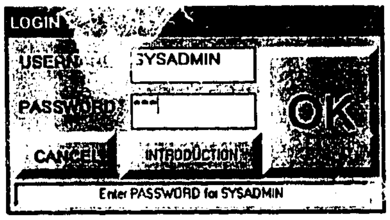
Our practice shows that it is easy and natural to apply the features and the capability of Meta-COOP for integrating hybrid knowledge for simulation.

Chapter 5

5. IMS INTERFACE AND DEMONSTRATION

The IMS user/developer interface is menu driven and customized based on the mill applications and integrated with MOPS and BQIS. The customization involves addition of new functions to MOPS user interface, creation of Knowledge acquisition system, integration with third party commercial neural network and expert systems. Some of these customizations are done on the server side, such as addition of all database communication with HDB and RDB, but most of the integration was done on the client PCs. Meta-COOP source code has been modified to be aware of all the added features; these features are also available to other modules within IOMCS project and MOPS.

In addition to Windows NT network security check, we have added a LOGIN window. The user name is used to check the access privileges of the user to various modules and databases.



SYSADMIN

Enter PASSWORD for SYSADMIN

Figure 5.1 IMS login window

In order to make the overall mill information system easily accessible, we were required to develop a general organizer to incorporate all the frequently used applications in one window as shown in Figure 5.2. All of the applications shown in Figure 5.2 have been integrated with each other using Microsoft's OLE concept. The remainder of this chapter will concentrate on IMS module only with some external function calls and communication.

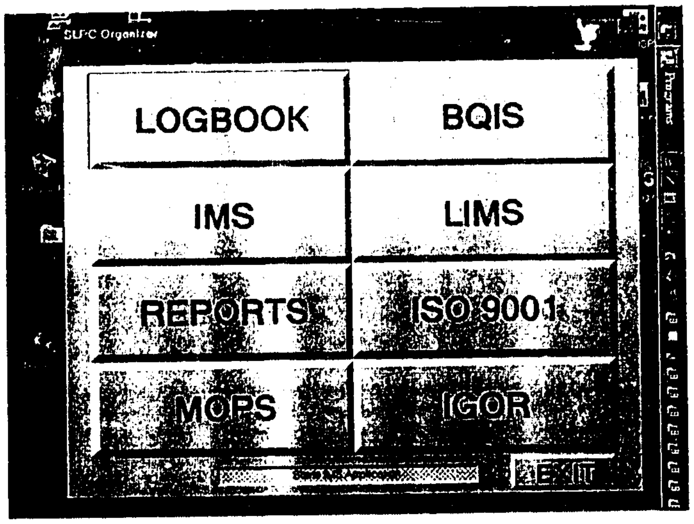
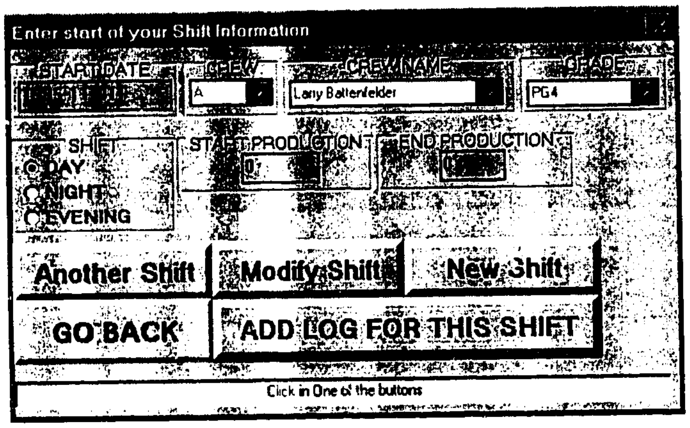


Figure 5.2 SLPC organizer screen



CREW: A
CREWNAME: Larry Baltenfelder
GRADE: PG4

Click in One of the buttons

Figure 5.3. Shift entry display

5.1 KNOWLEDGE ACQUISITION SYSTEM

Knowledge acquisition and organization is the major bottleneck in intelligent system development. The knowledge engineer is usually ignorant of the domain knowledge [Luger and Stubblefield, 1989]. To capture the domain knowledge continuously, we have developed an on-line system called, LOGBOOK. Operators, shift supervisors use this module to enter the operating conditions of the mill into the system. At the beginning of the shift, an operator selects his crew and shift supervisor for that crew, system gets grade and production from MOPS, user confirms the entries (Figure 5.3).

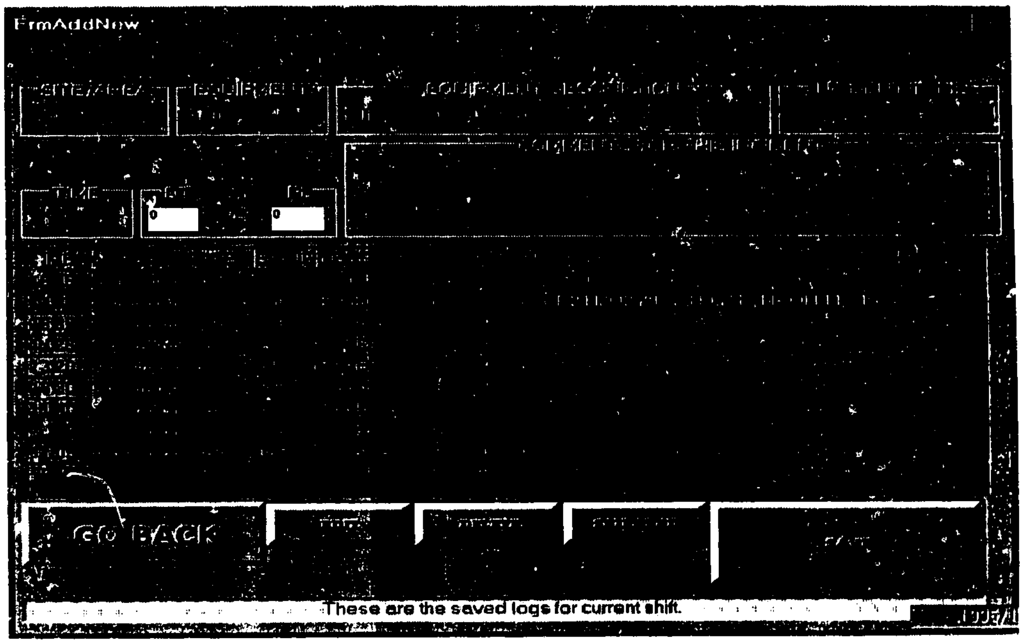


Figure 5.4: Logbook incident display

After selecting the initial information for the shift, the user enters all the incidents of process operation which are deviations from production recipes Technical department makes a recipe for operating the process and grade transitions, but sometimes operations need to deviate from a given recipe to achieve on-grade pulp. These deviations are stored in Logbook and used as a knowledge base (Figure 5.4). Logbook is attached to the RDB database for quick and easy accessing. This system also has features such as reporting the total production loss for a given period of time due to a specified equipment malfunction.

5.2 DEVELOPER INTERFACE

When developing the IMS interface, it was decided that the system would allow the developer to train and test knowledge bases off-line. However, in order to use the same system on-line, we had to connect the system to the process then capture history from RDB, CVD, and HDB and train it off-line. Therefore, all the developer options of the system are available from the end user interface. It can run within WINMOPS or as a standalone program as shown in Figures 5.5 and 5.11.

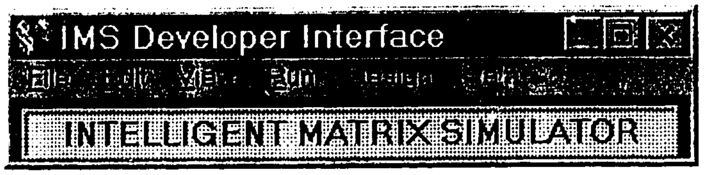
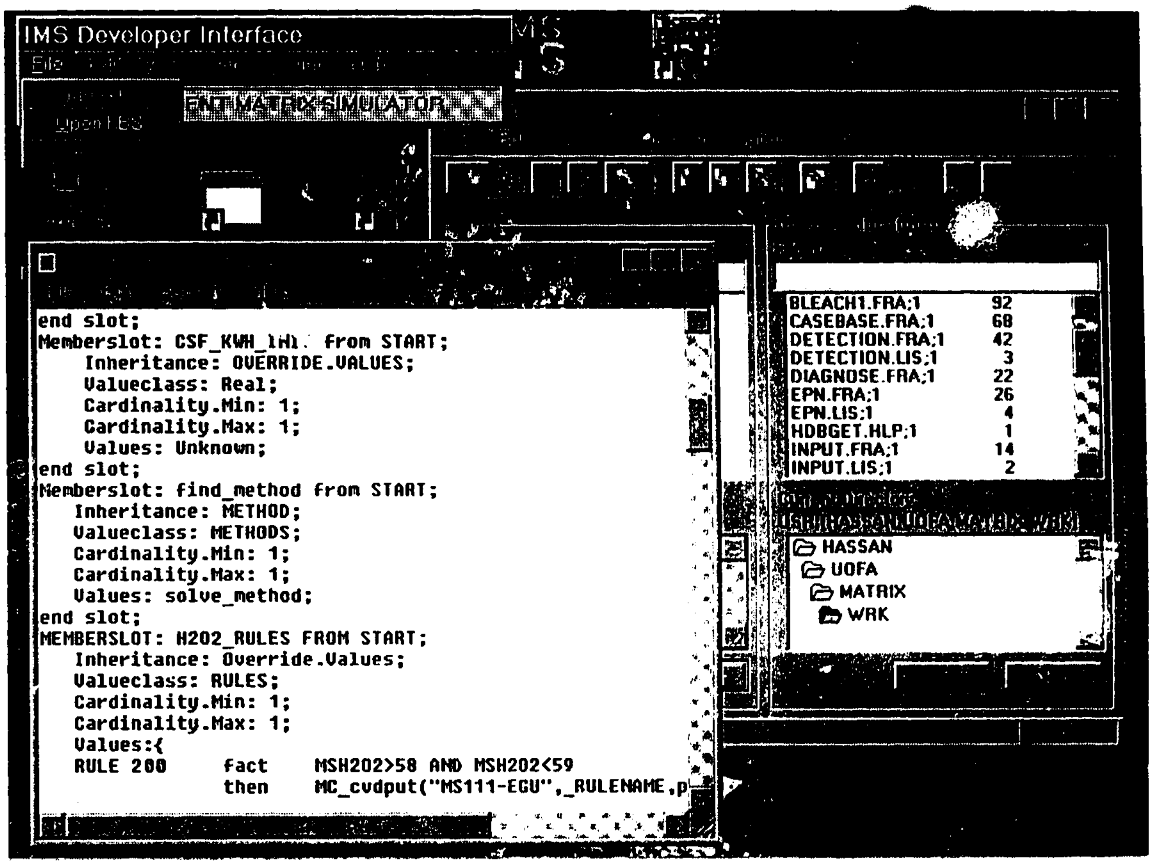


Figure 5.5: IMS Developer Interface

The FILE option is used for opening previously saved knowledge bases and neural network models. Engineers use this menu for modifying the knowledge base as shown in Figure 5.6. When the "Open Knowledge base" option is selected, the system activates an external file transfer protocol [Walker Richer & Quinn, 1994]. This protocol automatically logs on to the server and allows the user to select the knowledge base to be modified. After the modification is done, a trigger in RDB table of the selected knowledge causes the compiling module of the IMS server to compile the knowledge base. If there are any errors during compilation, users have a chance to make corrections. This knowledge base goes on-line and is used for simulation. The compiled module monitors the process operation using the data from MOPS databases and stores the operating conditions in its relational database (RDB). These stored data are used for building more cases in the logbook to be called by CBR. The graphics interface allow user to display graphically the actual outcomes of the result variables against their predicted values from the knowledge base. This can be used for graphing and further fine tuning of the knowledge. Within the knowledge base, we have developed functions which gets the data from databases. One of the functions is called MC_Cvdput as shown in Figure 5.6. MC_Cvdput is used for putting the simulated results into MOPS databases. Other functions include reading from CVD and HDB and writing to CVD and HDB databases.



```
end slot;
Memberslot: CSF_KWH_lHl. from START;
    Inheritance: OVERRIDE.VALUES;
    Valueclass: Real;
    Cardinality.Min: 1;
    Cardinality.Max: 1;
    Values: Unknown;
end slot;
Memberslot: find_method from START;
   Inheritance: METHOD;
   Valueclass: METHODS;
   Cardinality.Min: 1;
   Cardinality.Max: 1;
   Values: solve_method;
end slot;
MEMBERSLOT: H202_RULES FROM START;
   Inheritance: Override.Values;
   Valueclass: RULES;
   Cardinality.Min: 1;
   Cardinality.Max: 1;
   Values:{
   RULE 200      fact    MSH202>58 AND MSH202<59
                 then    MC_cvdput("MS111-EGU",_RULENAME,p
```

```
BLEACH1.FRA;1        92
CASEBASE.FRA;1       68
DETECTION.FRA;1      42
DETECTION.LIS;1       3
DIAGNOSE.FRA;1       22
EPN.FRA;1            26
EPN.LIS;1             4
HDBGET.HLP;1          1
INPUT.FRA;1          14
INPUT.LIS;1           2
```

Figure 5.6: Knowledge base modification facility

When "Open NN" from FILE menu option is selected, the client program activates a customized Neuroshell development environment. We have developed a data retrieval module which feeds the requested process data into Neuroshell. It uses the Dynamic Link Library (DLL) to extract data from CVD, HDB, and RDB [Maxis Inc. 1995]. Sample DLL calls from MOPS databases are shown in Table 5.1.

Table 5.1: Data retrival routine from MOPS

```
Declare Function Mops_Init Lib "ANTMOPS.DLL" (ch As Long, ByVal inifilename As String) As Long
Declare Function CVD_Read Lib "ANTMOPS.DLL" (ByVal ch As Long, ByVal tag As String, ByVal buffer As String, cvdsts As _
                    Integer, mcecode As Long) As Long
Declare Function CVD_Write Lib "ANTMOPS.DLL" (ByVal ch As Long, ByVal tag As String, ByVal buffer As String, cvdsts As _
                    Integer, mcecode As Long) As Long
Declare Function HDB_Read Lib "ANTMOPS.DLL" (ByVal ch As Long, ByVal ptdesc As String, mcests As Long, numvalues As_
                    Long, hdbsts As Any, values As Any, timestamps As Any, epndesc As Any) As Long
 HDB_ReadPoint(ch, ptdesc, mcecode, numvals, hdbsts(1), realvals1(1, 1), timestamps(1), hdbdesc)
Declare Function Mops_Uninit Lib "ANTMOPS.DLL" (ch As Long) As Long
```

The Mops_Init function establishes a connection to databases, CVD_Read and HDB_Read functions read the values from current value database and historical databases respectively. CVD_Write puts back the results into current value database and Mops_Uninit detaches the module from databases. CVD_Read can get values from CVD for up to 1000 points in one call. Data is returned in the buffers argument and the CVD status word, if any, is returned for each point in the cvdsts array argument. mcests contains the MOPS status code corresponding to each tag requested. epndescs is an array of structures containing information pertinent to each tags data, e.g. datatype, number of elements, and point handle. Both data and CVD status words are automatically translated into the local nodes representation from that on the remote MOPS system. After calling this routine, a 4 byte signed ANT status longword antsts indicating the result of the call is returned. Possible values of antsts are:

1) ANT_SUCCESS: Client application successfully connected to ANTCVD server and the read request was performed. This alone does not guarantee that valid data was obtained. Caller should also check the individual mcests return value of each point for MCE_SUCCESS before interpreting the buffer, cvdsts, and epndesc arguments.
2) ANT_INVPARAM: Handle is invalid. Make sure Mops_Init was called.
3) ANT_INITFAIL: Client application was not properly initialized.
4) ANT_ CONNFAIL: Failed to establish connection to ANTCVD server.

5) ANT_OUTOFBND: The value of the numtags argument is out of bounds.

6) ANT_READERR: Read error encountered while communicating with the MOPS CVD server.

7) ANT_WRITEERR: Write error encountered while communicating with the MOPS CVD server.

Table 5.2: Sample routine to retrieve CVD value from IMS Client

```
Sub CVDRead()
  Dim antsts, tempsts, numtags mcecode, realptr  As Long
  Dim tagname As String
  Dim temptag As Tag
  Dim index, cvdsts, i As Integer
  Dim epndesc As cvdepndesc
  Dim cvd_reals(1 To C_MAXCVDVALUES) As Single
  Dim cvd_int2s(1 To C_MAXCVDVALUES) As Integer
  Dim cvd_int4s(1 To C_MAXCVDVALUES) As Long
  Dim strvalue As String * 100

  tagname = TagBox.text
  index = VBStringToAsciz(temptag, tagname)
  realptr = CreatePointer(cvd_reals(1))
  numtags = 1
  CVD_Read(g_ch, numtags, temptag, realptr, cvdsts, mcecode, epndesc)
  ' generic buffer was used for CVDRead so now copy memory to appropriate datatype array
  Select Case epndesc.datatype
    Case TINT2
       tempsts = CopyMemory(cvd_int2s(1), cvd_reals(1), 2 * epndesc.numelements)
       DataBox.text = cvd_int2s(1)
    Case TINT4
      DataBox.text = cvd_int4s(1)
    Case TREAL4, TSCALED
       DataBox.text = cvd_reals(1)
     Case TASCII
       DataBox.text = strvalue
  End Select
End Sub
```

Table 5.2 is visual basic code of a sample client routine which reads a tag name and returns the corresponding CVD value to NN application. The following are some of the important function arguments of CVD_Read

1) ch: Handle to the ANTCVD servers returned to the caller by the Mops_Init function.

2) numtags: Number of tags to be retrieved. Valid range is 1 to 1000.

3) tags: Array of null-terminated ASCII strings of the tags to be retrieved from the CVD. These can be any valid MOPS tag such as tag[0] = 631ENER1-CV-CAL.

4) buffers: Array of pointers to buffer variables to receive the return data.

5) cvdsts: Array of MOPS CVD status words.

6) mcests: Array of MOPS status codes.

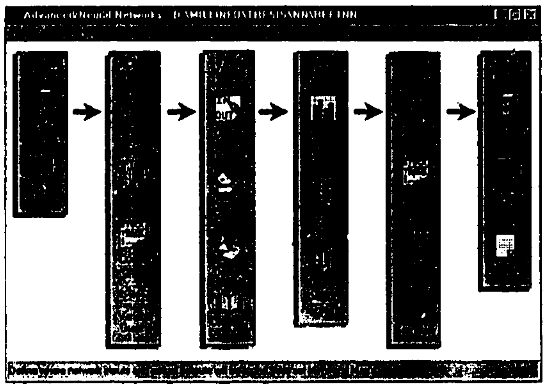
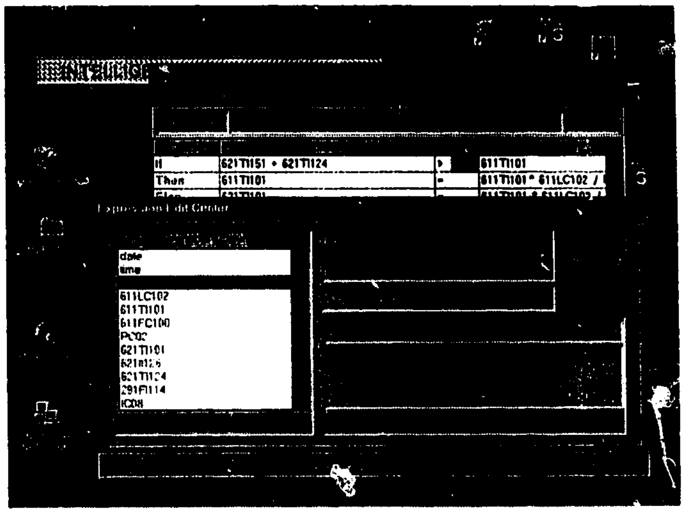


Figure 5.7: Main display of Neuroshell

7) epndescs: Array of EPN descriptor structures. If the corresponding mcests equals MCE_SUCCESS, this descriptor contains fields describing the datatype, number of elements, datalength of each element, and the point handle for the tag.

The Neuroshell based NN module is a software program that mimics the human brain's ability to make predictions or decisions based upon past experience. Just like the brain, however, neural networks are not guaranteed to always give an absolutely "correct" answer, especially if patterns are incomplete or conflicting. IMS has a module developed using Meta-COOP to evaluate the prediction results from various modules, such as Neuroshell and model-based modules. The prediction results with the highest confidence factor are then presented to users. Figure 5.7 is the development tools icons of Neuroshell.

This module offers several main menu options which enable the developer to work with real time process data within Neuroshell. The order of operations is to work from left to right. If icons appear in the same column, work from top to bottom. The developer may not have to use all of the icons that appear on a screen in order to create a working neural network module for IMS.



If	621TI151 + 621TI124	>	611TI101
Then	611TI101	=	611TI101 * 611LC102 /

date
time
611LC102
611TI101
611FC100
PC02
621TI101
621TI126
621TI124
291FI114
IC08

Figure 5.8: Production rules facility in Neuroshell

After collecting all the data from databases, the next step is data validation. This task h [illegible] ito two distinct functions: one is data selection and the other is data fil [illegible] ection task is fulfilled on the server using the production rules [illegible] The recipe table in RDB contains all the valid ranges of the [illegible] is table, in conjunction with grade table in RDB, is used by [illegible] ecting the data. The logic in this data selection is simple: [illegible] grade and the ranges of the operating conditions, we discard all process data that do not fit in the predefined range. After collecting all the data from databases the next main step is the filtering. We need to have a way of filtering out the invalid data before training the neural network. This bottleneck can be overcome by using the production rules facilities of Neuroshell as shown in Figure 5.8. After having the data and using a filter module to validate them, the developer chooses options from main menu of Neuroshell to train the network. Figure 5.9 shows the Network architecture.. available within Neuroshell system.

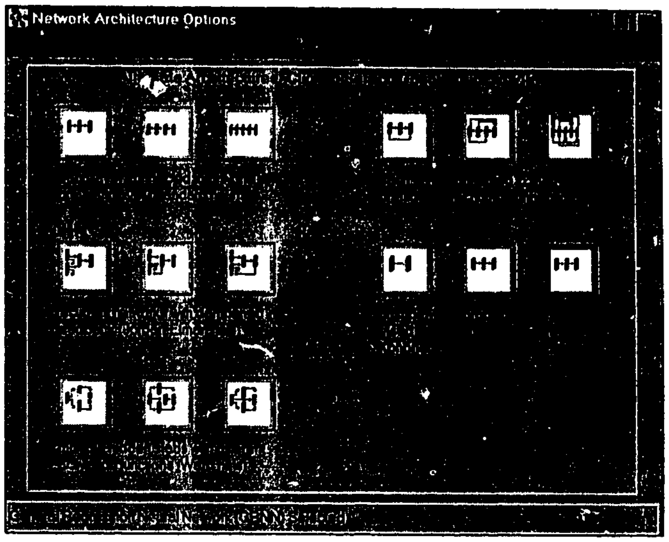


Figure 5.9: Network architectures in Neuroshell

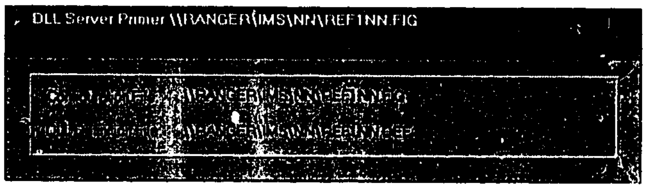

DLL Server Primer \\RANGER\IMS\NN\REF1NN.FIG

Figure 5.10: DLL server facility in Neuroshell

The DLL Server of Neuroshell is used for integrating the system with other modules. This capability allows the user to execute a trained Neuroshell network from Visual Basic, C, Excel, or Meta-COOP. The DLL Server gives the means of saving the network in a corresponding object within IMS so it can later be accessed via a Dynamic Link Library (DLL). Before any trained network can be executed by the DLL, a .DEF file for that network must be created by the Neuroshell DLL Primer which is pa.t of the Runtime facilities, as shown in Figure 5. .

Execution of a trained network is just the process of feeding an array of inputs to the network and receiving back the appropriate array of outputs (the predicted results). Once a DEF file is created, three DLL functions are needed to execute the neural network.

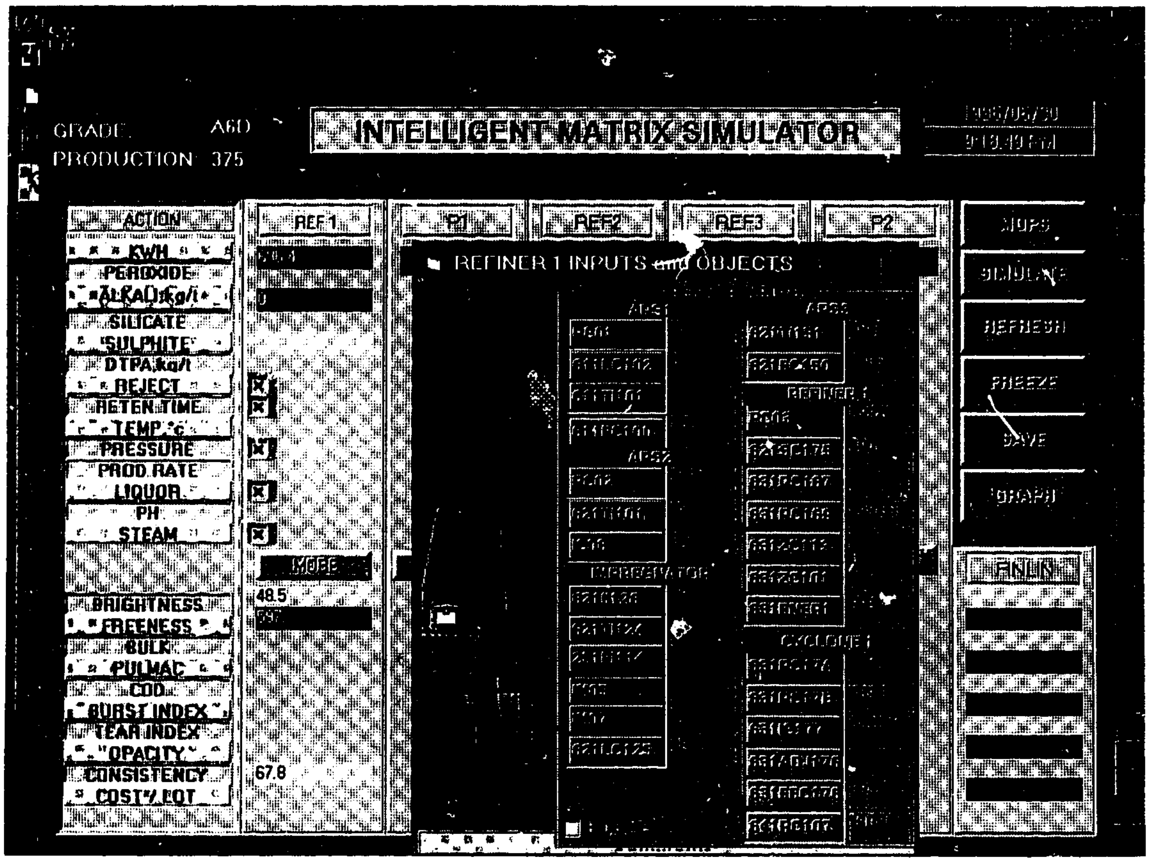


Figure 5.11: IMS user interface

1. *OpenNet* - This function reads the .DEF file and sets up the network. It gives a network number which is used for future reference to the network. It also gives the number of inputs the network expects and the number of outputs with which it will

respond. Normally, the develo already know the number of inputs and outputs anyway.

2. *FireNet* - Once the network is opened with OpenNet, use FireNet to pass inputs to the network and receive back outputs.
3. *CloseNet* - Execute this when program no longer need to execute the network. It will release all space taken by the network. The next time program needs to execute the network it will have to start from OpenNet again.

The above three functions allows Neuroshell to integrate with other modules in IMS.

5.3 USER INTERFACE

The main IMS user screen (Figure 5.11) consists of seven buttons and six menu items. The menus are mainly used for development and buttons are for end users. Using Windows NT server's user access rights for validation of user login, all the restrictions to menus get determined automatically.

The screen is divided into six matrices (columns), which represents the main stages of the process. The influential action variables are displayed on the top rows and the bottom rows show the result variables for each of the matrices. For example, the ***REF1*** object only influences three result variables: brightness, freeness and consistency. Clicking on the **MORE** button displays the action variables and their current process values. Using **FREEZE** button, user can stop the system from updating the data then change any of the action variables. In ***REF1*** object, there are 28 action variables which determines the outcome of the 3 result variables. Clicking on the **SIMULATE** button sends a message to the IMS SERVER in alpha which in turn activates its communication interface. This interface triggers the CBR function, which searches the knowledge base for known facts or rules, and activates neural network module, then simulates the known models based on the user entered data for solution. Meta-COOP collects all the results from different models and compares them to the closest fit data from previously run process cases. Meta-COOP then, sends a message along with simulated values to the user.

The **GRAPH** function is used for displaying the relevant plots from either previously collected process data, or neural network models. Figure 5.12 shows the outputs of actual and simulated values for consistency at the end of REF1 object using 600 actual lab tested data versus IMS simulated values.

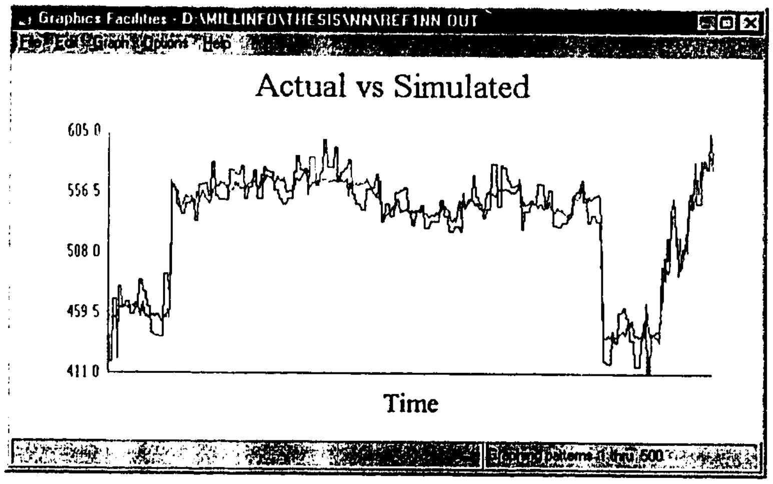


Figure 5.12: Simulated versus Lab test graph for Freeness

Selecting **SHOW CASES** from the run menu connects the user to the RDB database where all the previous cases are stored. The Open DataBase Connectivity (ODBC) tool developed by Oracle is used to connect to the RDB database. All the reports have been created using Crystal Reports [Crystal, 1995]. This allows users to view all the previous cases as shown in Figure 5.13. IMS user interface can be activated from MOPS as shown in Figure 5.14.

The help menu in main IMS display gives all the test procedures, mill operations, and grade guidelines. All the manuals are written in a hypermedia system developed by

Forefront technologies Inc. [ForeHelp User's Manual, 1995]. As shown in Figure 5.15, all the process knowledge help is available through this on-line manual as well.

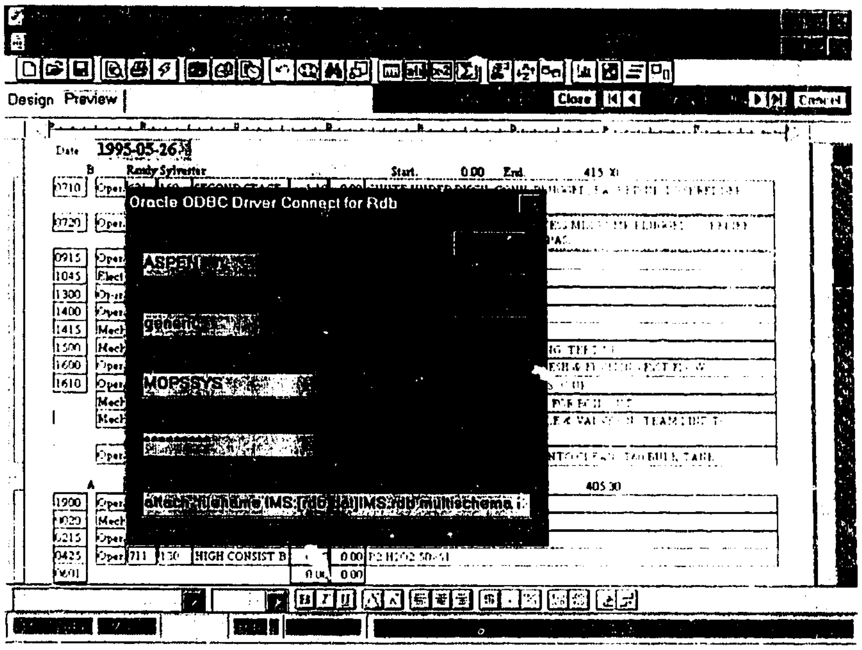


Figure 5.13: Report of previous cases in RDB database

The integrated developer interface makes it easy for mill engineers to modify and update process knowledge in IMS. The graphics user interface improves the acceptability of IMS to operators.

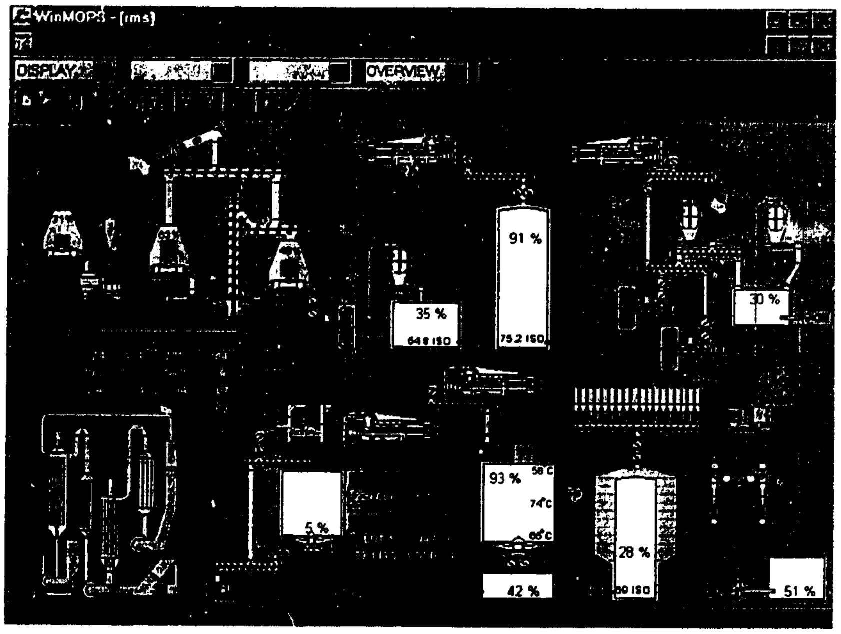


Figure 5.14: Integrated MOPS and IMS display

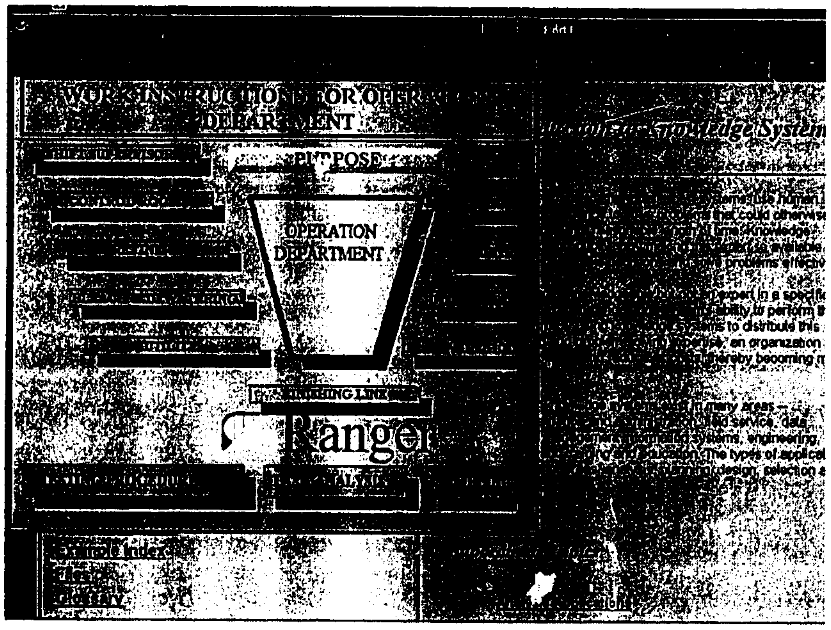


Figure5.15: IMS help facility

Chapter 6

6. CONCLUSIONS

The key factors used by analysts to evaluate the global pulp and paper industry are per capita consumption of paper, total tonnage produced and cost per tonne. An important trend in pulp and paper industry is making more and better product at lowest cost. One way of doing this is by empowering the decision makers during process operation by giving them the tools they need to make decisions on-line quickly and efficiently. Intelligent Matrix Simulation System (IMS) is such a tool developed for Slave Lake Pulp corporation. It is a user friendly graphical tool that models and displays the relationships between the *action* variables and the *result* variables for a given operating condition.

IMS is a knowledge integration environment which consists of a symbolic reasoning system, numerical computation programs, database management system, computer graphics packages, and multimedia interfaces. Two supervisory units fulfill the selection, coordination, operation and communication of the independent software systems. This integrated software environment allows execution of programs written in different languages, and application of multiple knowledge bases in the system. IMS is so integrated that it combines several individual software packages into a coordinated knowledge environment. The integration involves symbolic reasoning, neural networks, and case based reasoning; heterogeneous hardware platforms (Alpha server and Personal computer); and heterogeneous operating systems (Open VMS and Windows 95).

6.1 ACADEMIC CONTRIBUTIONS

Process models obtained from traditional methods such as system identification are usually linear approximations about chosen steady state operating condition. However, the characteristics of BCTMP equipment are highly nonlinear and very complex. Because of a lack of complete understanding of the process mechanism, differential equations which describe the dynamics of the processes are very difficult to obtain. On the other hand, most of the nowadays pulp and paper companies have applied distributed control systems and information management systems. They have vast amount of historical process data charting the system. In IMS, the historical data combined with the domain knowledge is used to develop a model that describe the input-output behaviour of the process. After modeling the process in IMS knowledge base, we have developed algorithms to simulate the process behaviour on-line.

The BCTMP process is a new process with very few experts available. Although SLPC has been in operation since December 1990, there are still many operating conditions which the mill have not encountered yet. We have used advance AI techniques to develop on-line learning algorithms. Using the on-line CBR, IMS can learn new cases and reuse the learnt cases as needed; also, ANN is used for extracting new relationships for given operating conditions and making them available on-line for prediction.

From system designer's point of view, we have developed an integrated intelligent system which is embedded into the existing systems such as MOPS and BQIS. Also, rather than using conventional sequential programming approach, IMS uses matrix simulation methodology to organize its knowledge base.

The prototype of IMS system has been developed and approved by the company. The implementation of IMS in several key process areas has been completed. Operating results have shown that IMS is very useful in reducing operator's working burden and improve the efficiency of operation decision making.

6.2 FUTURE RESEARCH

Though IMS has performed effectively in process operation, further research is required to further improve the performance and enhance the functionality of IMS. Areas of future research include:

1) Generic Model: The hierarchical IMS models are process specific. In order to improve the portability of the IMS, it is necessary to use a generic model structure. The model of each process component is treated as a module. By connecting the modules or changing the specification and definition of the modules, users can easily build models for a new application.

2) Process Optimization: The models built into IMS are the basis for process optimization. A new system will be developed to fulfill this task. When the process operation drifts from its desired conditions, the process optimization tasks will call the models in IMS. In return, IMS will perform simulations and use optimization to find a new operating condition that will result in desired quality, less raw material and chemical consumption and high production rate.

3) Engineering Interface Improvement: Currently the engineering interface of the system is not uniform among different modules of IMS. A universal engineering interface will be developed. This interface will enable mill engineers, who may not have much knowledge about intelligent system, to easily enter new models and build new applications.

4) System Extension: IMS is now implemented in several key process areas. More models will be built and more knowledge will be acquired in the following years. Our ultimate objective is to extend IMS to a mill-wide application. As one important part of the IOMCS system, IMS is being continuously improved to satisfy the requirements from other modules of IOMCS, such as fault-diagnosis.

REFERENCES

Alty J. L. and Coombs M. J. (1984) Expert Systems - Concepts and Examples, NCC Publications, Manchaster, England.

Aspen Technology Inc. (1988) Aspen Plus User Guide, Cambridge, MA 02139.

Aspen Technology Inc. (1991) Personal communication, Aspen Technology UK, Cambridge, England.

Behnam B. (1988) Introduction to neural networks for intelligent control, ITT Control System Magazine, April, 3-7.

Bhat N. and McAvoy [illegible] 90) Use of neural nets for dynamic modeling and control of chemical process [illegible] ms, Computers Chem. Engng., 14, 573-583.

Biondo S.J. (1989) Access through expert system shells to chemical process flowsheet simulators, ASPEN and SALT, Twentieth Annual Pittsburgh Conference on Modeling and Simulation, Research Triangle Park: Instrument Society of America.

Bozenhardt H. (1990) Intelligent simulation (ISIM): the new generation of expert systems, Advances in Instrumentation and Control, Proc. of the ISA 90 International Conference and Exhibit, pp. 9-16, New Orleans, Louisiana, Oct. 14-18.

Bretelle D. and Macchietto S. (1992) Dynamic simulation of a PVC batch reactor, Computers Chem. Engng., Vol. 17, Suppl., pp. S317-S322.

Bretelle D., Chua E.S. and Macchietto S. (1994) Simulation and on-line scheduling of a PVC process for operator training, Computers Chem. Engng., Vol. 18, Suppl., pp. S547-S551.

Bush M.J., and Silveston P.L. (1978) Computer simulation of wastewater treatment plant, Computers Chem. Engng., Vol. 2, pp. 143-151.

Caudill M. (1990) The Kohonen Model, Neural Network Primer. AI Expert, 25-31.

Chemetics International (1993) MOPS Engineering Manual, Vancouver, B.C.

Chemetics International (1995) MOPS User Guide, Vancouver, B.C.

Cognos Incorporated (1994) PowerHouse for OpenVMS, Ottawa, Ontario.

de Kleer J. and Brown J.S. (1984) A qualitative physics based on confluences, Artificial Intelligence, Vol. 24, pp. 571-585.

Dyne B., and M. Harvey (1992) Decision Support System for Pulp Blending Strategy, Pulp and Paper Expert Systems Workshop, Nov., Pointe Claire, Quebec.

Eikaas T.I. (1990) CADAS - A system for computer aided design, analysis and synthesis for industrial processes, Proc. of the Noric CACE Symposium, Nov. 15-16, Lyngby, Denmark.

Engell S. and Wollhaf K. (1994) Dynamic simulation of batch plants, Computers Chem. Engng., Vol. 18, Suppl., pp. S439-S444.

Evans L.B. (1981) Advances in process flowsheeting systems, Foundations of computer aided chemical process design, Vol. 1, R.S.H. Mah and W.D. Seider eds., Engineering Foundation, New York, pp. 425-469.

Evans L.B., Steward D.G., and Sprague C.R. (1968) Computer aided chemical process design, Chem Engng Prog, Vol. 64, pp. 39-46.

Fazadeh H., Rippon P., and Olofsson, J. (1996), Mill-wide Information System at Slave Lake Pulp Corporation, Proc. Canadian Pulp and Paper Association (CPPA) 82nd Annual Meeting and EXFOR'96, Jan. 30 - Feb3, Palais des Congres, Montreal, Canada, B275-B278.

Fazadeh H., Xia Q., Ying Y., Rao M., Danielson, K., Henriksson C., and Olofsson, J.

(1995) Knowledge Integration System for Slave Lake Pulp Corporation and Daishowa Peace River Mill, Journal of Record, Pulp and Paper Canada, October 1995. 25-28.

Fjellheim R.A. (1985) A knowledge based interface to process simulation, AI Applied to Simulation, ed. E.J.H. Kerckhoffs et al., San Diego: Society for Computer Simulation.

Flower J.R., and Whitehead B.D. (1973) Computer aided design: a survey of flowsheeting programs, The Chem Engr, Part I: No 272, Part II: No 273.

Forefront Incorporated (1995) ForeHelp User Manual, Forefront Inc Bouldor CO, USA.

Frank R.G.E. (1972) Modeling and simulation in chemical engineering, Wiley, New York.

Frith M.D., Henriksson C. (1992) Integration of Distributed Control and Mill Wide Information Systems at the Slave Lake Pulp Corporation Plant, CPPA, 1992 Spring Conference, Jasper, Alberta, Canada.

Gallant S.I. (1988) Connectionist Expert Systems, Communications of the ACM, 31, 152-69.

Gani R., Perregaard J., and Johansen H. (1990) Simulation strategies for design and analysis of complex chemical processes, Trans IChemE, Vol. 68, Part A, September, pp. 407-417.

Gokhale A.V., and Rice V.L. (1982) Real-Time Simulator for FCCU Operator Training, ISA Transactions, Vol. 31, pp. 39-46.

Henriksson C., Smith W., Danielson K., and Olofsson J. (1992) Value-added Information--A High Payback Investiment for the Mill, Proc. Tappi Process Control Conference, p. 5-19, Atlanta, Georgia, USA.

Hoskins J.C. and Himmelblau D.M. (1988) Artificial neural network models of knowledge representation in chemical engineering, Computes Chem. Engng., **12**, 881-90.

Hudson D.L., Cohen M.E., Anderson M.F. (1990) Use of neural network techniques in a medical expert system, International Journal of Intelligent System, 6(2), 213-23.

Hutton, L. (1992) Using Statistics to Assess the Performance of Neural Network Classifiers, Johns Hopkins Applied Physics Lab Technical Digest, Vol 13, No. 1.

Kitzmiller C.T., and J.S. Kowalik (1986) Symbolic and Numerical Computing in Knowledge-Based Systems, from Coupling Symbolic and Numerical Computing in Expert Systems (edited by J.S. Kowalik), Amsterdam, New York, Oxford, Tokyo.

Ko G., Osias M., Tremblay D., Barrera M., and Chen C. (1992) Process simulation in polymer manufacturing, Proc. European Symposium on Computer Aided Process Engineering, Computers Chem. Engng., 1992, S481-S490.

Kohonen T. (1988) An introduction to Neural Computing, Neural Networks, Vol. 1, No. 1, pp.3 - 16.

Kowalski, A. (1991) Diagnostic Expert System For Solving Pitch Problems, Pulp and Paper Expert Systems Workshop, Nov., Pointe Claire, Quebec.

Kuipers B. (1989) Qualitative reasoning: modeling and simulation with incomplete knowledge, Automatica, Vol. 25, pp. 571-585.

Larsen A.H. (1982) FLOWTRAN for process simulation, computer-aided process plant design. M.E. Leesly ed., Gulf Publishing Company, Houston, TX, (1982)

Leone H.P., Melli T.R., Montagna J.M., Vecchietti A.R., and Cerro R.L. (1987) A process simulator linked to a DBMS-3: the physicochemical properties package, Computers Chem. Engng., Vol. 11, pp. 567-579 .

Locke M.H., and Westerberg A.W. (1983) The Ascend-II system. A flowsheeting application of a successive quadratic programming methodology, Computers Chem. Engng., Vol. 7, pp. 615-630.

Luger G. F. and W. A. Stubblefield (1993) Artificial Inelligence and the Design of Expert Systems, The Benjamin/Commings Publishing, Inc. Redwood City, California.

Matson W. (1989) The Outlook for the Canadian Pulp and Paper Industry, Pulp and Paper Canada, 90:9, 297-302.

Maxus International Systems Limited (1995) ANT: MOPS API Developers guide, Vancouver, BC.

Moe H.I., and Hertzberg T. (1994) Advanced Computer architectures applied in dynamic process simulation: a review, Computers Chem. Engng., Vol. 18. Suppl., pp. S375-384.

Montagna J.M. (1993) Problem-Oriented Modules for Process Simulation. Resolution strategies for simulation and optimization, Canadian Journal of Chemical Engineering, Vol. 77, pp. 634-641

Myers W. (1990) Expert Interview with Elain Rich 'Expert systems and neural networks can work together', IEEE Expert, October 1990.

Neville J.M., and Seider W.D. (1980) Coal pretreatment extensions of FLOWTRAN to model solid-handling equipment, Computers Chem. Engng., Vol. 4, pp. 49-61.

Nishikawa Y., Kita H., and Kawamura A. (1990) A Neural Network Which Divides and Learns Environments, Proceedings of the International Joint Conference on Neural Networks. Vol 1, I-684 to I-687.

Oren T.I., and Zeigler B.P. (1988) AI in modeling and simulation: direction to explore. Simulation, Vol. 50, pp. 131-134.

Pantelides C.C. (1988) SPEEDUP-Recent advances in process simulation, Computers Chem. Engng., Vol. 12, pp. 745-755.

Papafotiou K., Assimacopoulos D., and Marinos-Kouris D. (1992) Synthesis of a reverse-osmosis desalination plant. an object-oriented approach, Trans IChemE, Vol. 70.

Perkins J.D. and Sargent R.W.H. (1982) SPEEDUP: a computer program for steady-state and dynamic simulation and design of chemical processes, Selected topic on Computer-Aided Process Design and Analysis. AIChE Symp. Ser., Vol. 78, pp. 1-11.

Ponton J.M. (1991) AI in process engineering, 1969 to 1991 and the future: some pertinenet questions, Proc. COPE'91, Barcelona, October, 15-18, pp. 3-8.

Prime Consultants Inc. (1995) BQIS Reference Manual, Vancouver, B.C.

Rao M. (1991) Integrated System for Intelligent Control, Berlin: Springer-Verlag.

Rao M., Wang Q., Cha J. (1992) Integrated Distributed Intelligent Systems in Manufacturing, Chapman & Hall, London.

Rao, M. (1992) Challenges and Frontiers of Intelligent Process Control, Engineering Applications of Artificial Intelligence, 5:5.

Rehbein D.A., Maze S.M., and Havener J.P. (1992) The application of neural networks in the process industry, ISA Transactions, Vol. 31, No. 4, pp. 7-13.

Rosen E.M. and Pauls A.C. (1977) Computer aided chemical process design: the FLOWTRAN system, Computers Chem. Engng, Vol. 1, pp.-21

Rumelhart D., and McClelland J. (1986) Parallel Distributed Processing, Cambridge, Mass.: The MIT Press.

Rumelhart D., McClelland J. and PDP Research Group (1986) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. Cambridge, MA: MIT Press/Bradford Books.

Schildt A. (1987) Artificial Intelligence using C, Osborne McGraw-Hill, Berkley, CA, USA.

Sorensen E.L., Johansen H., Gani R., and Fredenslund A. (1990) A dynamic simulator for design and analysis of chemical processes, Process Technology Proceedings, (L Puigjaner and A. Espuna Ed.) pp. 129-134. Elsivier, Amsterdam.

Spinos M. and Marinos-Kouris D. (1992) Integrated computer aided process design of waste water treatment plants on a PC system, Wat. Sci. Tech., Vol. 25, pp. 107-112. Papafotisou et al.

Stephanospoulos G., Johnston J., Kriticos T., Lakshmanan R., Mavrovouniotis M. and Sillet C. (1987) Design-Kit: An object oriented environment for process engineering, Computers in Chemical Engineering, Vol. 11, No. 6, 1987.

Walker Richer & Quinn Inc. (1994) Reflection suite Users Guide, Seattle WA 98109, USA.

Westerberg A., Hutchison H., Motard R. and Winter P. (1979) Process Flowsheeting, Cambridge University Press, Cambridge.

Xia Q., Fazadeh H., Rao M., Henriksson C., Danielson, K., and Olofsson, J. (1993) Intelligent Operation Support System for Slave Lake Pulp Corporation, Proc. 1993 IEEE Conference on Control Application, 591-596.

Xia Q., Rao M., Fazadeh H., Henriksson C. (1995) Case-Based Reasoning for Intelligent Fault Diagnosis and Decision Making in Pulp Processes, Proc. Canadian Pulp and

Paper Association (CPPA) 80th Annual Meeting and EXFOR'95, Jan. 30 - Feb3, Palais des Congres, Montreal, Canada.