A QUALITATIVE MODELING FACILITY
FOR
INTELLIGENT SUPERVISORY CONTROL SYSTEMS (ISCSs)

by

Edgardo Ivan Cifuentes
B.Sc., University of Concepción, Chile, 1980

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES
Department of Mining and Mineral Process Engineering

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
June 1995

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Edgardo Ivan Cifuentes

Department of Mining and Mineral Process Engineering

The University of British Columbia
517-6350 Stores Rd.
Vancouver, BC V6T 1Z4
Canada

June 28, 1995

ABSTRACT

Control strategies based on purely mathematical algorithms have only limited ability to cope with the type of operating conditions found in full-scale industrial applications. One approach to overcome this limitation integrates elements of automatic control theory, artificial intelligence and operations research into the design of a control system. This approach is employed in this research study to design an intelligent supervisory control system (ISCS). Elements of artificial intelligence to provide "human-like" characteristics for the ISCS are central to this research.

Pseudo-Qualitative Modeling as a central component of an ISCS is proposed in this research. This approach provides the mechanisms required by the ISCS to handle heuristic knowledge and approximate reasoning required in many supervisory control applications. A simulation study has been performed to demonstrate the validity of this approach. Simulation results have shown that this technique can handle either poorly-defined heuristic models or accurate models based on mathematical concepts. In fact, pseudo-qualitative modeling provides a framework to integrate qualitative and numerical models into a knowledge-based system.

A prototype of an ISCS was implemented using ProcessVision, a real-time SCADA (Supervisory Control And Data Acquisition) software package. This prototype system was applied to the C-line grinding circuit at Highland Valley Copper (HVC) to monitor and detect tonnage restrictions that affect circuit production. The diagnosis of tonnage restriction is currently performed manually by metallurgists on a weekly basis. This method is a heuristic procedure based on highly subjective judgment. The diagnosis results provided by the ISCS prototype during the evaluation period were well within the range of those reported by HVC metallurgists. Implementation of this ISCS prototype has demonstrated the feasibility of incorporating qualitative modeling into a commercial real-time SCADA system widely used in industrial applications.

TABLE OF CONTENTS

Page

ABSTRACT ii

TABLE OF CONTENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

ACKNOWLEDGMENTS ix

CHAPTER 1 INTRODUCTION 1

1.1 Background 1

1.2 Organization of the Work 2

1.3 Research Conducted in this Work 3

1.3.1 Objectives and Scope 3

1.3.2 Significance of the Proposed Research 4

1.3.3 Methodology 5

CHAPTER 2 ADVANCES IN SUPERVISORY CONTROL 7

2.1 Supervising Automatic Control: A Need 7

2.2 Incorporating Intelligence Into Supervision. 12

2.3 Advances in Supervisory Control 16

2.3.1 Determining Setpoints — Optimizing Control? 16

2.3.2 Fault Diagnosis 19

2.3.3 Expert Control 22

2.3.4 Human Operators in Supervision 23

CHAPTER 3 INTELLIGENT CONTROL APPROACH 26

3.1 Introduction to Intelligent Control 26

3.2 Intelligent Control Approach: Main Characteristics 27

3.2.1 Overall Structure 27

3.2.2 Representing and Reasoning Upon Symbolic Knowledge 28

3.2.3 Dealing with Uncertain and Incomplete Knowledge 29

Page

3.2.4 The Need to Learn 30
3.2.5 Making Decisions on Time 31
3.3 Elements of Artificial Intelligence (AI) in Intelligent Control (IC) 32
3.3.1 Knowledge Representation (KR) 32
3.3.2 Handling Uncertainty and Incompleteness 37
3.3.3 Learning Techniques 39
CHAPTER 4 DESIGN OF AN INTELLIGENT SUPERVISORY CONTROL SYSTEM (ISCS) 41
4.1 General Structure 41
4.2 General Functional Definition 43
4.2.1 Global Functions of the ISCS 43
4.2.2 Supervisory Control Tasks 44
4.2.3 Regulatory Control Tasks 47
4.3 Hardware and Software Requirements 47
4.3.1 General Software Specifications 47
4.3.2 General Hardware Specifications 48
4.4 Potential AI Techniques Involved in the Design of an ISCS 49
CHAPTER 5 PSEUDO-QUALITATIVE MODELING APPROACH 51
5.1 Introduction 51
5.2 Alternative Techniques to Qualitative Modeling 52
5.3 A Pseudo-Qualitative Approach to Qualitative Modeling 54
5.4 Building a QM of a Head Tank 60
5.5 Potential and Limitations of the Approach Proposed 69
5.6 Incorporation of Qualitative Modeling into an ISCS 73
CHAPTER 6 DESIGN AND IMPLEMENTATION OF AN ISCS PROTOTYPE 75
6.1 Objective and Scope 75
6.2 Background 76
6.3 General Structure 76
6.4 ProcessVision Software Package 77
6.5 Tonnage Restrictions in the C-Line Grinding Circuit at HVC 79
6.6 Design Aspects of the ISCS Prototype 82
6.7 Experimental Results 91

Page

6.8 Evaluation of Results 102

CHAPTER 7 FINAL CONCLUSIONS AND RECOMMENDATIONS 108

7.1 Final Conclusions 108

7.2 Recommendations for Future Research 111

CHAPTER 8 CLAIMS TO ORIGINAL RESEARCH 112

NOMENCLATURE 113

BIBLIOGRAPHY 115

APPENDIX A AI ELEMENTS THAT CAN BE INCORPORATED INTO AN ISCS 125

APPENDIX B PROCEDURE TO BUILDING A QM INTO THE ISCS 141

APPENDIX C KNOWLEDGE BASE FOR THE HEAD TANK 143

APPENDIX D KNOWLEDGE BASE FOR THE ISCS PROTOTYPE 171

LIST OF TABLES

Page

Table 6.1 Restrictions handled by the ISCS prototype and their possible causes. 86
Table 6.2 Diagnostic results provided by the ISCS. Period: March 26 - 31, 1995 103

LIST OF FIGURES

Page

Figure 2.1 Factors affecting the application of automatic control....8
Figure 2.2 Results from the application of automatic control, (adapted from Rogers, 1985)....9
Figure 2.3 Effect of automatic control on the excursion of the controlled variables, (Latour 1976)....10
Figure 2.4 Effect of supervisory control on automatic operation....12
Figure 2.5 Block diagram of an expert control system, (Åström, 1989)....23
Figure 3.1 Intelligent control: merging control theory, artificial intelligence and operations research, (Saridis 1987)....26
Figure 3.2 Overall structure of an intelligent control system, (Rao, 1992)....28
Figure 3.3 Example of frame-based knowledge representation....33
Figure 3.4 Example of semantic net representation....36
Figure 3.5 Fuzzy representation of "high" for the power draw....37
Figure 3.6 General structure for empirical learning techniques, (adapted from Forsyth and Rada, 1986)....40
Figure 4.1 Approach proposed for the ISCS....41
Figure 4.2 General structure for the Intelligent Supervisory Control System (ISCS)....42
Figure 5.1 General structure of a QM under the pseudo-qualitative approach proposed....54
Figure 5.2 Fuzzy Sets that define qualitative information of variables....55
Figure 5.3 Quadratic input/output behavior represented qualitatively using primitives **qm_function** and **qm_table(3)**....58
Figure 5.4 Behavior predicted by a first order dynamic model as a variable evolves from one steady state value to another....59
Figure 5.5 Head Tank analyzed in the Case Study....60
Figure 5.6 QM for the level of fluid in the tank, based on a ***qm_table(3)*** primitive....62
Figure 5.7 Actual and predicted level of fluid in the tank; ***frate_in*** takes the values 100, 50, and 75 lpm....66
Figure 5.8 Simulation results for two valve openings: $v = 50$ % and $v = 100$ %; ***frate_in*** takes the values 100, 50, and 75 lpm....66
Figure 5.9 Simulation results on two similar tanks but with different areas; ***frate_in*** takes the values 100 and 50 lpm....67

Page

Figure 5.10 Simulation results obtained from a corrected QM of the tank. ***frate_in*** takes the values 100, 50, and 75 lpm. 68

Figure 5.11 Representation of a quadratic relationship under the pseudo-qualitative approach proposed (continuous line) and under a purely qualitative one (shaded areas), using a generic definition of fuzzy concepts. 71

Figure 5.12 Representation of a quadratic relationship using a pure QM at different levels of resolution and the pseudo-qualitative approach proposed. 72

Figure 6.1 General structure of the implementation of the ISCS prototype. 77

Figure 6.2 C Line Grinding Circuit at HVC. 79

Figure 6.3 Setting a high limit for the fresh feed setpoint. 81

Figure 6.4 Sequence of major tasks performed by the ISCS. 84

Figure 6.5 Estimating tonnage losses from historic trends (schematic trends). 87

Figure 6.6 Estimating tonnage lost due to a "soft ore" restriction (schematic trends). 88

Figure 6.7 User Interface: Process Overview Display 90

Figure 6.8 User Interface: Tonnage loss per Category. 91

Figure 6.9 Soft ore restriction: April 1, 1995, day shift. 92

Figure 6.10 Restriction RR_FEED_CUT: April 1, day shift. 94

Figure 6.11 Restriction RR_FEEDER_DOWN: March 30, day shift. 96

Figure 6.12 Restriction RR_ORE_SUPPLY. March 30, night shift. 97

Figure 6.13 Restriction RR_SAG_STOPS: March 29, day shift. 99

Figure 6.14 Many restrictions due to plugged chutes. April 1, day and night shifts. 100

Figure A.1 Fuzzy concept "hard," P(OH/h) 127

Figure A.2 Concept "high" for the power draw 128

Figure A.3 Inference network for rule R2. 129

Figure A.4 Combining pieces of evidence when they are not conditionally independent. 132

Figure A.5 Domain theory to compare values of process variables. 135

Figure A.6 Example of a control loop operating OK. 136

Figure A.7 Proof tree for the example of Figure A.6. 137

Figure A.8 Generalized explanation for the control loop case. 137

Figure A.9 Unrestricted generalization for the case of the control loop. 138

Figure A.10 Restricted generalization for the case of the control loop 139

Figure A.11 User defined concept hierarchy 139

ACKNOWLEDGMENTS

This is a very special moment in my life, a moment that I share intimately with my wife and best friend, Monique. She patiently endured many lonely days and nights and the postponement of important aspects of our family life because ... "I had to work on my thesis." Her role in the thesis was not passive, though. She read my thesis drafts and discussed with me about some of the issues involved in the research; now she is an expert in the area. Even more important, her steadfast support and gentle words of love and encouragement were essential for me to overcome the stressful and dark periods of this research project. I am forever grateful to her.

My deepest gratitude is to Dr. John Meech, who supervised the latter part of this work. It has been a pleasure to work with him; he always showed great enthusiasm and a positive attitude towards this research. His comments and suggestions, drawn from his expertise in the area, were essential for the success of this research project. I was impressed by his incredible energy and willingness to work even overnight and during weekends to read my drafts in order to give me a timely feedback. My gratitude also to Prof. Andy Mular for his financial support of this project. Also, thanks to Randy Morrison, Mile Boss, Andy Burkert and other friends that proof-read early thesis drafts.

My gratitude to Dr. Brian Flintoff from Brenda Process Technology, for his support and encouragement during the first part of this work. Thanks are extended to Noranda and MITEC for their financial support. My sincere gratitude to Comdale Technologies for facilitating ProcessVision, the SCADA software used for the development of the ISCS prototype. My appreciation to Highland Valley Copper (HVC) Mine for giving me the opportunity to evaluate the ISCS prototype in one of their grinding circuits. Special thanks to Hans Raabe who provided the expert knowledge to build the prototype. Thanks also to Colin Lindsay and John Mitchell for their comments on the prototype, and to Brent Veitch for his enthusiasm and readiness to help with the interconnection between the ISCS prototype and the HVC control system.

Finally, in retrospect, I can only conclude that none of the above would have been possible nor meaningful without the love of the eternal and only living God.

CHAPTER 1

INTRODUCTION

1.1 Background

Substantial progress has been made in automatic control over the past few decades. Many different advanced control algorithms using a variety of approaches have been proposed. However, the application of the new algorithms has been rather slow, so a gap between academic algorithms and practical versions found in industrial applications has occurred. Furthermore, it is now well-recognized that automatic control theory fails to cope with many practical control problems found in an industrial environment (Tong 1977; Wittenmark and Åström 1984).

Human operators have played an important role in supporting the automatic operation of a process. An operator often takes control of the process, overriding the automatic control system when it is unable to cope with some specific operating circumstances. Does the operator have a mathematical model of the process, or advanced numerical control algorithms? This is doubtful. A human operator performs well even in the absence of precise information. He/she possesses the ability to handle partial and incomplete information and reason at a low level of resolution. These are key characteristics that need to be incorporated into a control system.

This research focuses on the design of a control system that supports integration of both numerical and qualitative elements. Central to this research is the development of a qualitative approach that offers the mechanisms required by a control system to handle the subjective judgments and approximate reasoning found in the manual supervision of many industrial processes.

1.2 Organization of the Work

The second part of this chapter presents in detail the objectives of this research study as well as the methodology followed. Chapter 2 presents an introduction to supervisory control and a brief description of recent advances in the area. The need for incorporating human-like intelligence into a supervisory system is established in this chapter. Chapter 3 describes an approach to incorporate intelligence into a supervisory system. The chapter describes the fundamentals of this approach and the different artificial intelligence elements involved. Chapter 4 presents the overall structure proposed for a supervisory system that integrates numerical and qualitative elements. It briefly presents the hardware and software requirements for the implementation of such a system, as well as the major artificial intelligence elements that may be incorporated into the system.

Chapter 5 is a key section of this thesis as it describes in detail an approach to incorporate qualitative modeling into a control system using a modified technique which is called Pseudo-Qualitative Modeling. A simulation study is presented and discussed as the basis for validation of the method. Chapter 6 presents the development of a prototype system incorporating a pseudo-qualitative model. The results of the application of the prototype to a full-scale industrial grinding circuit at Highland Valley Copper Mine are presented and evaluated. Chapter 7 presents a synthesis of conclusions about the central aspects of this research. Finally, Chapter 8 presents a list of claims to original research as the specific contribution of this study.

1.3 Research Conducted in this Work

1.3.1 Objectives and Scope

The main objective of this research can be expressed as:

> Incorporation of a qualitative modeling facility into an Intelligent Supervisory Control System (ISCS) to assist in the real-time supervision of the automatic operation of an industrial process.

For ISCS we understand a supervisory control system designed under an intelligent control approach, i.e. its design integrates elements of artificial intelligence with conventional elements of automatic control.

Some of the specific sub-goals for this project are:

i) Gain in-depth knowledge of different approaches to incorporate human-like intelligence into a supervisory control system.

ii) Propose an approach to qualitative model that is suitable for operation in a real-time environment and able to represent heuristic knowledge about the behavior of industrial unit operations.

ii) Design and develop an ISCS prototype system that employs a qualitative approach to assist in the supervision of a full-scale industrial operation.

This study is intended to be a basic platform to demonstrate the feasibility of incorporating qualitative models into the design of a supervisory control system. Qualitative models provide a supervisory system with appropriate mechanisms to handle incomplete knowledge and approximate reasoning about the behavior of a physical process.

The work focuses on the use of a pseudo-qualitative modeling approach to build adequate process representation into a supervisory control system. An existing intelligent SCADA package (Supervisory Control and Data Acquisition) called ProcessVision is used as the reservoir for the qualitative models. The research will demonstrate how qualitative models can be applied within an existing artificial intelligence (AI) tool.

Other AI elements such as handling uncertainty and on-line learning will also be analyzed in brief as requirements for future AI developments.

An ISCS prototype that employs a qualitative approach to model and supervise the operation of complex processes will be developed in the last stage of this research. This prototype will be implemented using a real-time commercial system and applied to the C-line grinding circuit at Highland Valley Copper (HVC). The focus of this application will be to monitor and detect tonnage restrictions that affect circuit production. At this stage, given the objectives of this research study, the system will not intervene directly with the process. Appropriate corrective actions will be left to the discretion of plant operators.

1.3.2 Significance of the Proposed Research

Conventional control systems based on purely numerical elements have limited ability to cope with real world problems. One major approach taken to overcome these limitations has been the integration of AI elements and automatic control theory. This approach marked the beginning of the field of intelligent control.

The design of an ISCS as proposed in this research is an important step towards development of a new generation of supervisory systems that integrate quantitative and qualitative elements. There have been other studies in this area but none of them with the scope of this research. The results of

this research will be supported by the application of an ISCS prototype to a full-scale industrial process.

Finally, the other relevant aspect of the research is the area of application: semi-autogenous grinding. Semi-autogenous grinding is one of the most important stages within mineral processing, and offers a challenging world of opportunities for improvement to automatic control practitioners.

1.3.3 Methodology

The first step of this research is to review the aspects of supervisory control as they relate to the need for AI methodologies. The second step includes a review of different AI techniques available for knowledge representation, reasoning and learning. The focus of this review is on qualitative modeling and the unique requirements of real-time applications. A specific approach to qualitative modeling, i.e. the pseudo-qualitative modeling approach, is proposed.

Third, the pseudo-qualitative modeling approach is refined and validated via simulation studies. A qualitative model (QM) of a well-understood process, a head tank, is built and simulated under different conditions; the simulation results are compared to those obtained from a mathematical model of the same process. This simple process was purposely chosen for this simulation study as it helps to visualize clearly, and demonstrate, the principles of the approach proposed.

Fourth, an ISCS prototype is built following the general structure proposed for the design of such systems. This prototype, implemented using a real-time commercial SCADA system, incorporates the pseudo-qualitative modeling approach proposed and demonstrates the ease of its application within available AI software.

The final step of this research involves the application of the ISCS prototype built to the C-line grinding circuit at Highland Valley Copper (HVC). The ISCS provides on-line advice to assist in the supervision of the automatic operation of the process. The system monitors and detects tonnage restrictions that affect circuit production. Experimental results are employed to assess benefits and limitations of the approach proposed.

CHAPTER 2

ADVANCES IN SUPERVISORY CONTROL

2.1 Supervising Automatic Control: A Need

In the early days of the industrial era, processing plants were designed and constructed with virtually no instrumentation. Lack of control knowledge and technological limitations were mainly responsible for this. Since then, we have witnessed major technological breakthroughs which, combined with advances in knowledge, have led to automatic control as a proven technology. However, there is no guarantee that automatic control applications are always successful.

It is tempting to view the solution of an automatic control application problem merely as the installation of a commercially available state-of-the-art piece of equipment. Although we disagree with this practice, the reality is that many current industrial applications of automatic control (and the mineral process industry is not unique in this regard), seem to follow this practice without regard to fundamental understanding of control theory. Mular (1989) states in his study of process control in mineral processing plants that: "Since 1977 older computer control systems have been or are being replaced by up-to-date distributed control equipment which incorporate microprocessor controllers." However, such "up-to-date" hardware technology may not employ modern control strategies. In 1989, Mular and Burkert concluded that "To date, advanced control techniques have found minimal use in control strategies for SAG circuits." Since that time, however, we have seen a number of successful developments in the use of modern techniques for grinding mill circuit control (Eggert and Benford 1994; Perry and Hall 1994; and McDermott et al. 1994).

Installing a computer-based process control system does not guarantee improved control. Various factors affect the application of automatic control to industrial processes, the combination of which may mean the difference between failure and success for a particular application. These may be classified as inherent to the control system, to the process, to humans (personnel related to the application), and to the environment, as illustrated in Figure 2.1. The control problem is not only a piece of hardware together with control algorithms that manipulate process inputs to achieve desired output targets. Thought must be given to other issues involved in the overall control problem. Flintoff and Mular (1992), referring to the effects of process and human factors on control applications, note: "If a circuit or plant is plagued by breakdowns, or is a victim of poor design or poor operation, chances are that control will not provide the most cost effective solutions to the problems. In the extreme, these problems can render the control system prohibitively expensive and/or ineffective."

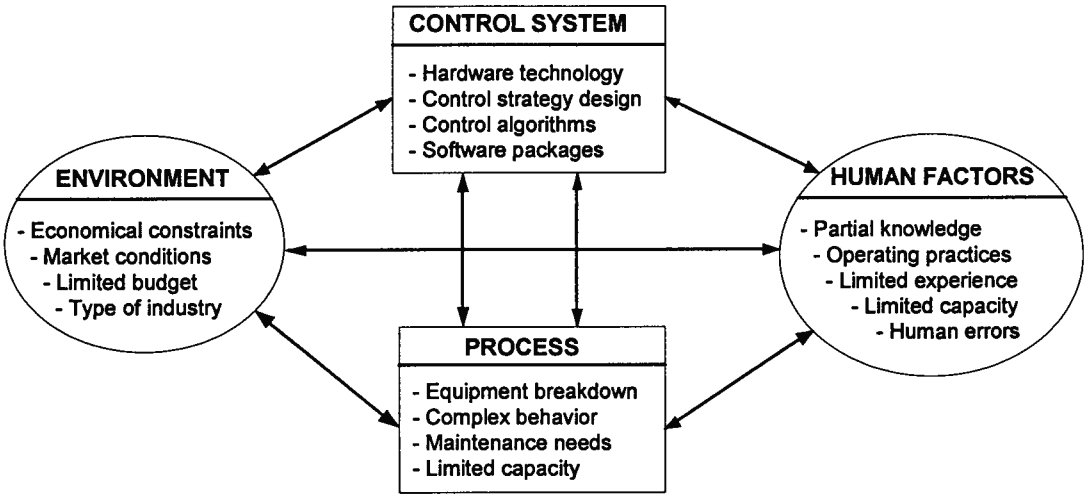


Figure 2.1 Factors affecting the application of automatic control.

The combined effect of different factors on a typical control application is represented in Figure 2.2 (adapted from Rogers, 1985). This diagram indicates that, in general, the results of automatic control application should lie somewhere between manual control and a theoretical optimum achievable.

Yet, in some extreme cases, automatic control may be outperformed by manual control. A more optimistic view of Figure 2.2 tells us that there is always room for improvement given an appropriate automatic control strategy—the theoretical optimum is always an attractive goal to aim for. Different control algorithms and different control strategies yield different results; some will be closer to the theoretical optimum while others will struggle to move away from the manual operation "ghost."

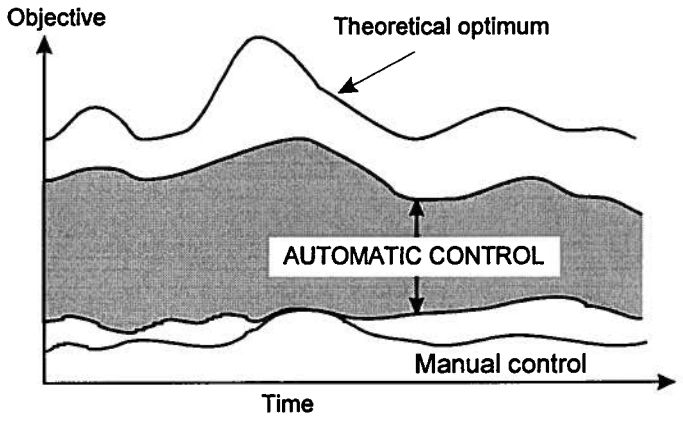


Figure 2.2 Results from the application of automatic control, (adapted from Rogers, 1985).

Advances in control theory during the past few decades are such that the theoretical optimum may be within reach. With the advent of microcomputer technology, algorithms that can deal with large delay times, time-varying parameters, multi-variable processes, noise in process variables and measurements, autonomous operation, and so on, have been developed (Fortescue et al. 1981; Fuentes and Cifuentes 1988; Clarke 1981; Clarke and Gawthrop 1981; Åström and Wittenmark 1989; Antsaklis et al. 1989 & 1990; Tuffs and Clarke 1985). However, in spite of these advances, automatic control theory on its own still fails to cope with many practical control problems found in an industrial environment. As a result, researchers recognize the importance of appropriate supervision of different design aspects of industrial controllers. A self-tuning algorithm, for instance, requires supervision of its identification procedure, parameter estimation, controller design and closed loop stability (Isermann and Lachmann 1985; Wittenmark and Åström 1984; Sanchez et

al. 1988). Such supervision is indispensable for an algorithm to succeed in the industrial world. Lack of attention to supervision has contributed to the gap between academic algorithms and practical versions that must operate in an industrial setting. The truth is an industrial controller must deal not only with complex process characteristics such as nonlinearities, delays and time-varying parameters, but also with different operating upsets such as tanks overflowing, broken belts, mills overloading, chutes plugging and so on. To address these constraints, prior knowledge or supervision is essential for any controller to survive. Even a simple PID (Proportional-Integral-Derivative) controller needs supervision. A PID algorithm is only the heart of an industrial PID controller; sampling, filtering, conditioning, alarming and anti-wind-up mechanisms are some of the logistic support needed by a PID controller.

Well-tuned controllers, in general, reduce the variance (excursions) of controlled variables, as shown in Figure 2.3 (Latour 1976; Hales et al. 1982). However, badly-tuned controllers may cause more problems than benefits; they may even cause instabilities in process behavior. Unfortunately, results such as that shown in Figure 2.3, for properly tuned controllers, are valid only during "normal" periods of operation.

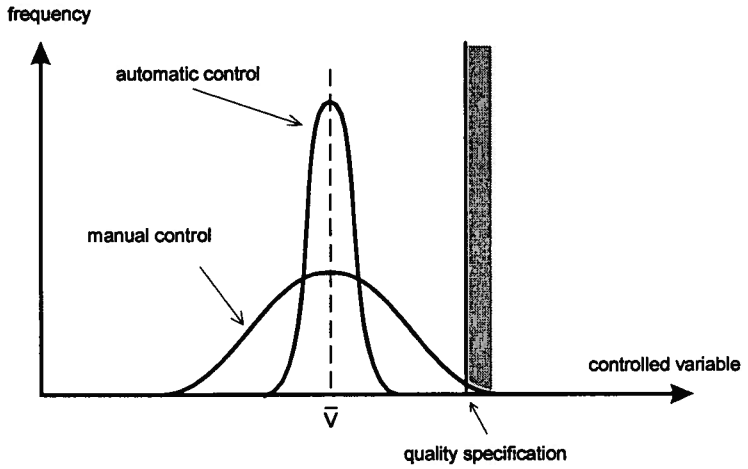


Figure 2.3 Effect of automatic control on the excursion of the controlled variables, (Latour 1976).

During normal operation, a process is not unduly disturbed, equipment operates near nominal levels (not in an overload condition or during procedures of start-up or shut-down) and process variables are within normal ranges (no alarms are reached; they are at a safe distance from unstable or highly nonlinear zones). As soon as these normal conditions are violated, a human supervisor starts to play an important role in supporting the automatic operation and expanding its limits of operation beyond the original boundaries given by "normal" operating conditions. It is a human operator with his knowledge, experience, skills and intuition who takes control of the process and is able to guide its operation through these abnormal periods of operation.

A human operator is highly resourceful but has limited capacities. His ability to supervise a process deteriorates rapidly as the complexity of the situation increases, and excessive attention is required for extensive periods of time (Drury 1976; Lees and Sayers 1976; van Delf 1985). For instance, it is dangerous for an operator to adjust controlled variables too close to limit values such as operating constraints, quality specifications, or boundaries of stable operating zones, and maintain those operating points for a prolonged time. The position of these target landmarks vary continually with time due to disturbances. Thus having narrowed down the excursions of controlled variables by incorporating regulatory control, we need a computer-based supervisory system to guide the operation of such variables. This computer-based system would lead the controlled variables toward their optimum operating points, and would deal with constraints usually found on that path. This is a task that cannot be done only once. It must be continually repeated, as the optimum point and constraints are essentially time-varying features of a given process (see Figure 2.4). Clearly, something to alleviate the stress on a human operator such as a supervisory control system is very important. The supervisory control system has to deal not only with the complexity of process behavior but also with operating disturbances such as break down of equipment, overload conditions and so on.

No matter what type of individual controllers comprises the regulatory level, a proper supervision of both individual and collective operation of the controllers is essential to bring the operation of a process closer to the theoretical optimum, as shown in Figure 2.4.

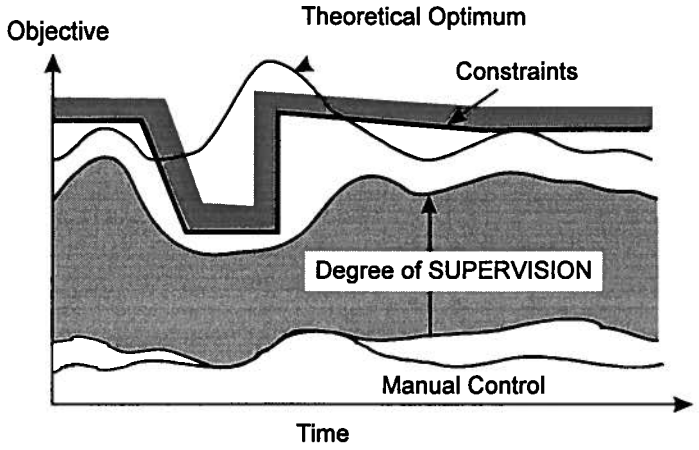


Figure 2.4 Effect of supervisory control on automatic operation.

2.2 Incorporating Intelligence Into Supervision.

Traditionally, supervision has been the exclusive domain of human beings. Despite advances in knowledge and technology, it still enjoys an aura of being an art that only a skillful operator can master. Computers, although powerful machines, are still relegated to taking more routine and well-defined tasks which require systematic execution of procedures or algorithms. Human operators have retained control of higher order functions such as long-term planning, overall monitoring, and consultation when unusual situations arise.

To design a supervisory system we need to examine the different functions and tasks involved, their nature and how humans have been handling them. Sheridan (1976) classifies supervision activities into four major task headers: planning, programming, monitoring and intervening. When planning, a human operator deals with questions and decisions concerning topics such as: the operating strategy

to use to reach production goals; the type of actions needed given some specific process behavior; the information required to assess the behavior of the process; and so on. Programming involves all the activities an operator performs when he tries to make a system operate as planned. These include: implementing a particular control strategy; setting up specific parameters for process equipment and for control system elements; and modifying the process layout as planned. Monitoring involves activities such as attending to information on displays and panels, evaluating process behavior, assessing the fulfillment of planned goals, and diagnosing causes of failures or abnormal process behavior. Intervening involves interruption by the human operator of the control loops, and so requires direct interaction between the operator and the process.

A human operator is flexible and inventive; he captures and transforms information displayed by the plant into appropriate actions according to the current goal. To perform these supervisory tasks mental representations are derived from different sources such as prior experience with plant behavior; from knowledge of internal anatomy and functioning of the plant; or from prescribed rules and instructions. Furthermore, different mechanisms are used to accomplish the goal; e.g. responding "automatically" to a situation, or identifying a problem or following a more formal step-by-step mental processing.

Fischler and Firschein (1987) state that one of the most remarkable attributes of human intelligence is the ability to convert a problem into a familiar representation so it can be solved with previously used techniques. Two people who confront the same physical reality will represent and interpret it in different ways, in accordance with a stored framework (model, metaphor or image), that is used by the person to deal with the outside world. Fischler quotes North who says, "we carry around in our head a model of the world, of society, of the local community, of the family—even of oneself, and none of us can deal with any of these entities, even superficially, without reference to the appropriate mental construct or model" (North 1976). It is the only way we have to relate to other people and to

our larger surroundings. We draw upon these models whenever we discuss affairs, whenever we vote, and whenever we plan for the future in any way.

What sort of mechanisms does an operator use to represent the information he handles? How does he represent and store his experience? Are they numbers, symbols, images or abstract concepts? What kind of mental processing does he perform? Does he use some type of logic for reasoning, or is it pure mathematical operations? Too many questions, and not very many answers. Psychologists and cognitive scientists are still at the beginning of understanding the complexities of human behavior.

Johnson-Laird (1989), notes that thinking occurs in such a dazzling variety that some cognitive scientists have despaired of understanding it. There is, at one extreme, the free flow of ideas in daydreams; which are mental processes that have no specific goal. At the other extreme, there is mental arithmetic. Here the thinking is deterministic: it has a precise goal, and it is done in a voluntary and consciously controlled way. At each point, the next step in the calculation is determined by its current state. Johnson-Laird suggests that perhaps most thinking lies between these two extremes. It has a goal, but it is not carried out like a calculation, and people do not follow a strictly determined procedure. Different people may tackle the same problem in different ways. Even the same person, given a second chance in ignorance of his first attempt, may take a different path the second time around. Nothing constrains us to just a single option at each step in a thinking process.

An important characteristic of human reasoning when supervising a process is that observations are generally expressed at a higher level of abstraction related to the normal plant state, the expected operating point, the next task, etc. Rasmussen (1976) notes that the operator, rather than "reading" instruments, appears to ask higher level questions of the system and use individual instrument readings as symbols to define system states. Rasmussen presents a detailed schematic that shows a

sequence of mental activities followed by an operator between initiation of a reasoning process and execution of an action. After detecting a specific problem, the operator arrives at different "states of knowledge" before deciding on a specific action. These states are connected to each other via appropriate mental processes. A skilled operator may not enter the sequence of states at the same starting point at all times, or may change the order of the steps, or may choose a sequence with a different number of steps. In fact, an experienced operator only "occasionally" moves through all the steps in a given sequence. A skilled operator is flexible and inventive, so his mental processing can lead him to "jump" from one state of knowledge to another later in the sequence. Rasmussen observes that, when entering a study of human performance in real-life tasks one rapidly finds oneself "rushing in where angels fear to tread."

Although we are still far from fully understanding human behavior in supervision, we are certain about one thing: the need to incorporate "human-like" characteristics into the design of a supervisory system. The system should be able to handle both quantitative and qualitative information; interact with a quantitative process world and communicate with human users in both formats. It should also be able to handle uncertain information and incomplete and approximate knowledge, and reason upon this type of knowledge. It should have the capacity to learn. It should also ... We could continue enumerating desirable (if not essential) features for the supervisory system. If we look at these features, they are simply a description of an operator's abilities in relation to plant supervision.

Researchers in the field of intelligent control (IC) are working toward this objective of incorporating human-like features into a control system. IC researchers combine elements of artificial intelligence (AI) with traditional automatic control elements to produce new alternative solutions to the control problem. Several different mechanisms have been proposed that attempt to capture the richness of humans regarding knowledge representation and reasoning with this knowledge. Some of these mechanisms are: semantic nets, decision trees, state transition graphs, frames, logic, mathematics and

production systems (Cifuentes and Mular 1992; Berenji 1990; Efstahiou 1990; Karetnyk et al. 1988; Leitch 1988; McGregor et al. 1992). Researchers have also proposed different mechanisms to add learning capacity to a computer-based system (Crosscope and Bonnell 1988; Geng and Jamshidi 1988; Kokar 1988; Lee and Kim 1988; Peters et al. 1992). Although the initial dream of a human-like computer-based system is remote and may never be reached, we believe the approach taken by IC researchers points in the right direction. As we will see in more detail in the following sections, this approach has natural advantages over a fully numerical one.

2.3 Advances in Supervisory Control

There are fewer studies conducted in supervisory control as compared to the number of publications concerning regulatory control. In many cases supervisory control is considered equivalent to a class of optimizing routines that decide the values of setpoints for regulatory control; most of the applications in the literature follow this view. Fault diagnosis has also received attention from researchers, although not always related to process supervision. The general impression is that much more work is necessary to formalize the knowledge available in this area.

2.3.1 Determining Setpoints — Optimizing Control?

A supervisory control system sits above the regulatory control level and determines the value of setpoints for the different controllers composing that level in accordance with suitable criteria (Mular 1989; Flintoff and Mular 1992). Deciding setpoints for regulatory controllers is certainly important but supervisory control should not be limited to this task alone. It is essential to incorporate mechanisms to deal with special operating conditions such as process disturbances, abnormal operation and special procedures; and so expand the automatic operation of the process beyond the initial constraints of normal operation.

The algorithms to determine setpoints can vary from optimization algorithms to *ad-hoc* empirical methods that keep processes within stable or safe operating limits. An optimization algorithm calculates setpoint values seeking to minimize (or maximize) an objective function such as an efficiency or productivity index for the process. Steady-state optimization has been more appealing, since global economic indexes are directly related to steady-state behavior of processes. Dynamic optimization of processes, which considers time as an explicit optimization variable, has been left to researchers in optimal control.

The approaches can be classified as model-based optimization and empirical optimization. Under a model-based approach, both process models and objective functions are described by algebraic equations and an optimization algorithm is used to search for the optimum point. Numerical techniques such as linear programming, gradient and steepest-ascent are often used to maximize the objective function. If models are steady state, the optimization is steady state. Empirical optimization is generally used when there is no appropriate model to describe the process. This approach may use either an explicit or implicit objective function. In this case, process measurements are used to evaluate current operating points, and heuristic algorithms are employed to decide changes on setpoint values. Part of a heuristic algorithm may look like this: "If efficiency increased with last change, then repeat that change; if it decreased, reverse last change." This empirical approach is the most common one in industrial applications not only from a lack of appropriate models, but also because of time-varying characteristics.

Mular and Burkert (1988) review control strategies for SAG circuits at a number of operations. One of the operations examined was the SAG mill at the Los Bronces Plant in Chile, also described by Jerez et al. (1985). The control strategy at this plant aimed at: maximizing the mill throughput using maximum available power, while maintaining stable power draw. An empirical approach is followed with no explicit objective function. Regulatory control is composed of two major control loops

based on variable gain PID controllers: a power loop and a bearing pressure loop. Supervisory control was implemented using a procedural language (at the time of their publication), designed to check if the mill was in an overload condition; to bring the mill out of overload; to determine the water/fresh-feed ratio; and to decide the feedrate setpoint according to the fresh-feed size distribution. If overload occurs, two alternative actions may be taken to bring the mill back to a stable zone, namely, override the power loop and use the bearing pressure loop, or decrease the power setpoint step by step during several five minute periods. A process engineer decides beforehand on one of these alternative actions. Once the mill is again in a stable region, the operator returns mill control to the power loop, which increases the setpoint in steps until the nominal value is reached. The water/ore ratio is modified by a hill climbing routine, which increases or decreases the water ratio according to the changes produced on the power draw by previous water changes.

The approach followed by McManus et al. (1975), for the same problem of maximizing the throughput is similar to that of Jerez et al. (1985). The specific heuristic strategy, however, is different than that followed by Jerez. This difference in strategy may reflect either differences in the circuits or differences between personnel involved in each application.

Most optimization strategies designed to determine setpoint values in the mineral industry follow the general empirical approach just described (Krog 1979; Ratte 1979; Bartrum et al. 1986). The approach followed by Hales and Herbst in which an expert system (ES) was used as part of the optimization strategy also falls within this category. In their case, they use a Kalman filter to estimate unmeasured variables and an ES shell to implement the heuristic strategy for optimization and safe operation of the process (Hales and Pate 1985; Hales et al. 1988; Herbst et al. 1989).

Eggert and Benford (1994) and McDermott et al. (1994) also use ES technology to supervise the automatic operation of a process. Eggert uses ES and heuristic rules drawn from experts to supervise

the automatic operation of the grinding process at Dome Mine, Ontario, Canada. McDermott's application is similar to that of Eggert but for a grinding circuit at Wabush Mines, Newfoundland, Canada. The objective of increasing tonnage throughput was met in both applications.

2.3.2 Fault Diagnosis

There is substantial work in the area of fault detection and diagnosis, although not all related to process supervision. Some of this work is oriented to digital circuits (Hamscher and Davis 1984; de Kleer 1976; de Kleer and Williams 1987; Pipitone 1986), others to medicine (Shortliffe 1976; Kuipers 1989; Weiss et al. 1979), and to industrial processes (Scarl et al. 1987; Thompson and Clancey 1986; Jones et al. 1988, 1990a & 1990b; Vagenas and Grangholm 1991; Rao and Ying 1990). Some of these diagnostic strategies also detect multiple faults as in the case of Arjunan (1988) and de Kleer and Williams (1987).

Fault diagnosis consists of a mechanism to detect the occurrence of a fault and to find the cause for that fault. Mechanisms vary from fully numerical to fully qualitative, and from using structural knowledge of the elements involved to using heuristic knowledge in the form of compiled pattern matching (Milne 1987a & 1987b).

Reiter (1987) presents a general theory for the fault diagnostic problem based on first principles (structural knowledge). He claims this theory can be applied to a variety of task domains such as medicine, digital and analog circuits, and database updates. His approach starts with a description of the system and how its components interact. Observation of the system's behavior is conducted. A diagnostic problem arises when this observation conflicts with the way the system is meant to behave. The problem is to determine those system components which, when functioning abnormally, explain the discrepancy between observed and correct system behavior. He proposes an algorithm

that determines all possible diagnoses for a given system with faults; and then uses a set of additional measurements to discriminate among competing diagnoses. Other approaches based on first principles include those of de Kleer (1976), de Kleer and Williams (1987), and Davis (1985). These methods, however, do not incorporate expert experience in diagnostic reasoning.

Another approach referred to as the "deep model" approach uses a model of the system to detect failures. In this approach the input-output behavior of a system (or of the elements of a system) is represented by a model that produces expected outputs, which are compared to observed values to determine the occurrence of a fault. Kuipers (1987) presents a model-based reasoning approach for diagnostic purposes using qualitative models and his qualitative simulator QSIM. The goal of his work is to produce an explanation that accounts for the observed differences in the behavior of the system. An adequate explanation will cover descriptions of a correctly functioning system, a description of any changes to that system that would account for the primary differences in the malfunctioning system, and finally, a demonstration or confirmation that the perturbed mechanism can produce the relevant malfunctions that have been observed.

Scarl et al. (1987) also present a model-based approach using knowledge of structure and function of the system. Their diagnostic reasoning is as follows: pick an element for the fault detected; hypothesize some faulty state for the suspect; use the model to derive expected values for the system's sensors; test these values for consistency with actual sensors' measurements; if all sensors are consistent with their expected values then the hypothesized fault is one possible explanation for the malfunction detected. They use knowledge of the structure and function to guide the search to eliminate possible candidates.

Jones et al. (1988) compared two fault diagnosis approaches—a qualitative and a quantitative one, using a laboratory system that consists of a pump and a multiple tank assembly. The qualitative

approach referred to as "plant transgression-based diagnostics" consists of IF-THEN rules, pairing "transgressions" observed in the plant with "faults." This is built using domain knowledge acquired from an expert through first-hand experience. The second approach, referred to as "model transgression-based diagnostics," uses three types of quantitative model of the plant: a model of the normal process, a model of the observed process, and models of faulty processes. These models can be used by either recognizing discrepancies between real and model outputs or by recognizing similarities between the observed model and known faulty models of the process. The strategy also employs IF-THEN rules as in the case of the qualitative approach. Their results show that faults are easier to detect with plant transgression-based diagnostics approaches than with model transgression-based approaches; although the latter may be extremely effective at detecting faults which manifest themselves as changes in internal plant dynamics.

Thompson and Clancey (1986) used Heracles, a qualitative modeling environment derived from Mycin (Shortliffe 1976) in the diagnosis of a sand casting process. Burrows et al. (1989) used the ES shell Nexpert to improve reliability of assay information in a flotation plant. Their goal is to diagnose data provided by an X-ray fluorescence analysis system, detect erroneous data points and replace them by realistic ones. The principle used by Burrows' work is similar to that of diagnosing the operation of a process; in one case diagnosis is applied to data and in the others to physical elements. Vagenas and Grangholm (1991) present a study of the potential application of expert systems to fault diagnosis of mining equipment in Swedish mines. They consider the possibilities of incorporating different types of knowledge into an ES including descriptive, behavioral, operational, heuristic, diagnostic and maintenance knowledge; and also two diagnostic strategies: decision tree or heuristic and model-based. One of their conclusions is that diagnosis in mining still relies mostly on empirical approaches and that much work needs to be done to formalize such experiential knowledge.

2.3.3 Expert Control

The approach followed mainly by Åström referred to as "expert control" is conceptually quite close to a supervisory system for control functions (Åström et al. 1986; Åström 1989, Åström and Wittenmark 1989; Verbruggen and Åström 1990). The idea of expert control is to have a collection of algorithms of control, identification and estimation orchestrated by an ES. Instead of having one control algorithm and one estimation algorithm, the system has several algorithms including those for excitation and for diagnosis, as well as tables to store data. An ES overseeing the operation decides when a particular algorithm should be used or what sort of action should be taken at a given time (see Figure 2.5).

Åström recommends a bank of control algorithms and an associated database with encoded information for each of them in order that the ES can evaluate their behavior. A set of direct supervisory routines are designed to evaluate different aspects of the operation of the algorithms. Among these routines, we should mention an "excitation routine" to ensure proper operation of the estimator; a "jump detector" for sudden changes in parameters; a "perturbation routine" to generate an excitation signal; and a "stability supervisor" to monitor closed loop stability. The knowledge base contains process data, results from the direct supervisory routines, operating indices, and monitoring tables with a history of different aspects of the behavior of the system. The ES uses heuristic knowledge about particular algorithms and those conditions in which they can be used, and information on how to supervise the overall operation of the control system. The ES decides when to use a particular control algorithm or when there is a need for additional process excitation for better parameter estimation.

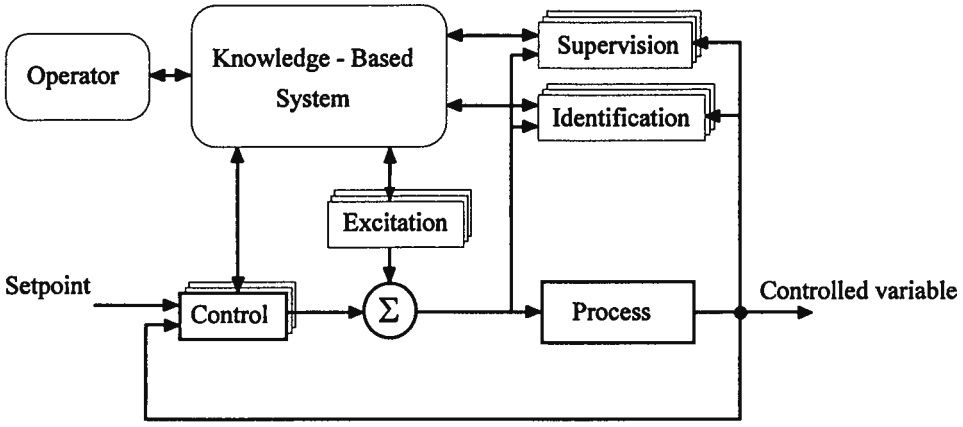


Figure 2.5 Block diagram of an expert control system, (Åström, 1989).

2.3.4 Human Operators in Supervision

Another aspect that has received attention from cognitive science researchers is the role of human operators in supervision when interacting with an automated system. This area has been ignored by researchers in process control, although it may have a definite impact on the effectiveness and acceptance of supervisory systems.

Singleton (1976) says that whenever a computer is assisting operators to control a complex process, the computer normally performs the more routine tasks whereas human operators retain the longer-term planning function, an overall monitoring function, and a consultation function when unusual situations arise. Rasmussen (1986) also agrees with this saying that, "automation does not remove humans from the system; it basically moves them from the immediate control of system operation to higher-level supervisory tasks and to longer-term maintenance and planning tasks." Thus human operators will continue playing a vital role in the overall effectiveness of a supervisory system. Good design of the man-machine interface may still account for the difference between failure and success of a system. This design, according to Rasmussen, should be based on a generic model of the

information processes involved in the decisions to be taken by a human operator; on how he perceives and represents the world, and on how he deals with a specific situation he faces.

But how does an operator perceive the physical system he is dealing with? What sort of mental representation or abstraction of that system does he have? Researchers argue that operators perceive a physical plant they are supervising as a functional abstract hierarchy. At the bottom of this hierarchy, we find physical elements with location, interconnection and form; whereas at the top, we find the overall functional purpose of such plant (Rasmussen 1986; Pikaar 1985). This perception of the "world" is quite flexible; it very much depends upon the goals and intentions of the person. In dealing with a specific situation he may only have a certain part of the whole system within his span of attention. To him, this portion is "the system" to be represented (his entire abstract hierarchy at that time) and the rest of the system becomes part of the "environment." As his focus of attention changes, in accordance with the situation, his "system "also changes eventually corresponding to the entire global system.

Another interesting classification of human behavior presented by Rasmussen (1986), distinguishes skill-based, rule-based and knowledge-based behavior. Skill-based behavior represents sensorimotor performance during acts or activities that, after a statement of an intention, take place without conscious control as smooth, automated, and highly integrated patterns of behavior. Rule-based behavior is goal-oriented and the particular activities chosen are structured and consciously controlled by stored rules or procedures that may have been derived empirically during previous occasions, communicated from other persons' know-how as an instruction or cookbook recipe, or it may be prepared on each occasion by conscious problem solving and planning. At the highest level of knowledge-based behavior, performance is goal-controlled and knowledge-based. Here, the goal is explicitly formulated and a useful plan is developed. Different plans are considered and their effect tested against the goal. This test is either physically by trial-and-error or, conceptually by

understanding the functional properties of the environment and predicting the effects of the plan considered.

Understanding human behavior and performance in supervision contributes to better design of man-machine systems. Sutton (1990) argues that to meet the challenges of designing man-machine systems, it is essential to develop models of human operators and that, as in any other field, these models have to be formalized and validated. He presents a detailed study on models of human operators closer to the lower level of behavior, referred to as "skill-based" behavior by Rasmussen. His study covers his view on man-machine interaction, proposition and simulation of different linear and non-linear models and incorporation of fuzzy set theory to man-machine interaction. Millot et al. (1985) present some approaches to man-machine cooperation in automated processes. Pikaar (1985) presents a hierarchy of activities at different levels of decision in a supervisory system that may be assigned either to the computer or to the operator depending on the level of automation. Sheridan (1976) also sees the operator of an automated system being removed from direct involvement at the loop level towards an overseeing role. He classifies supervisory functions as: planning, teaching, monitoring and intervening.

Jovic and van Delf study different aspects of human performance in supervision such as short and long-term memory, effect of motivation, data visualization, sampling rate and increasing complexity of the tasks (Jovic 1992; van Delf 1985). They describe an operators' role and interaction with a computerized system; the main tasks dependent on the purpose of the system and level of supervision responsibilities. Bösser (1985) has examined how training may affect operators' performance. Drury (1976) goes further, examining how performance is influenced by other aspects such as availability and accuracy of the information, access to intermediate and historical information, process complexity and practice on a given task.

CHAPTER 3

INTELLIGENT CONTROL APPROACH

3.1 Introduction to Intelligent Control

The term Intelligent Control (IC), traditionally attributed to Fu (1971), was more extensively discussed and officially born as a research field at the First IEEE Workshop on IC, New York, 1985. Saridis later defines it as control "which would replace the human mind in making decision, planning control strategies, and learning new functions whenever the environment does not allow and does not justify the presence of a human operator" (Saridis 1987). IC is still developing so new definitions and formalizations continue to be an important discussion topic at annual IC workshops. Saridis reinforces the idea of IC as the intersection of artificial intelligence (AI), operations research (OR), and automatic control (AC), (see Figure 3.1).

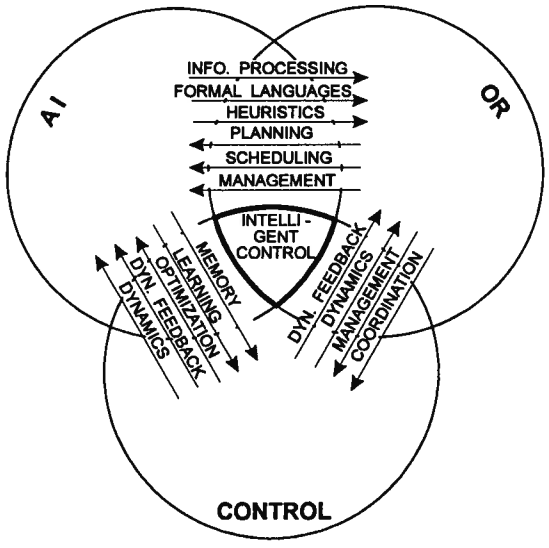


Figure 3.1 Intelligent control: merging control theory, artificial intelligence and operations research, (Saridis 1987).

As argued in a panel discussion during the 1987 IEEE Int. Symposium on IC (Panel Discussion 1987), IC is not AI because an IC approach uses a more rigid (more logical, more mathematical) description of the goal and sub-goals of the system; a more rigid description of the system with gradually increasing knowledge and adequacy of the models; and a more rigid method for its design. IC is not "conventional" automatic control either, since the goal of the system is described symbolically, the system is known imprecisely, our knowledge increases gradually during operation, and uncertainty plays an important role in the design of controllers.

3.2 Intelligent Control Approach: Main Characteristics

3.2.1 Overall Structure

The approach of IC follows a general principle of "decreasing precision with increasing intelligence," which establishes a hierarchy in the distribution of intelligence within the structure of an intelligent control system. Following this approach, Saridis proposes a hierarchical system with three levels of intelligence: An "organizational" level that represents the master-mind of the system with functions dominated by AI; a "coordination" level that interfaces the high and low levels and with functions dominated by AI and operations research; and an "execution" level with high requirements in precision and functions dominated by systems theory (Saridis 1985).

Rao emphasizes that the design of an intelligent control system should "integrate" symbolic tools with existing technology in automatic control and not just substitute this existing technology (Rao et al. 1988; Rao 1992). Rao proposes an architecture to control and manage large-scale intelligent systems which integrates: knowledge on different disciplinary domains; empirical expertise and analytical knowledge; different symbolic processing and numeric computational systems; and information in different formats such as symbolic, numeric and graphic information. On top of these

different modules of knowledge integration he visualizes a "meta-system" which: coordinates and manages all symbolic reasoning systems and numerical routines; distributes knowledge into separate expert systems (ESs) and numerical routines; acquires new knowledge; finds a near optimal solution for the conflicting solutions and facts among the different ESs; provides the possibility of parallel processing; communicates with the measuring devices and the final control elements of the control system; and transforms various input/output signals into the standard communication signals (see Figure 3.2).

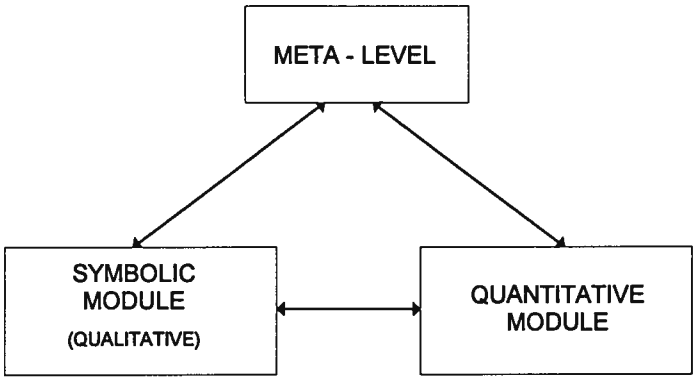


Figure 3.2 Overall structure of an intelligent control system, (Rao, 1992).

3.2.2 Representing and Reasoning Upon Symbolic Knowledge

There are many tasks associated with the operation of a control system which are essentially mathematical calculations for which proven mathematical techniques exist, e.g. filtering, parameter estimation, control algorithms and optimization techniques. The control problem, however, requires more than a collection of mathematical algorithms. As we mentioned before, it involves a substantial amount of heuristic knowledge, especially at the supervisory level of control.

An operator supervising a particular process does not check the mathematical calculations involved; rather he focuses his attention on the overall performance observed. After a change in setpoint, he

observes the evolution of the control variable, and may say: "this controller is well tuned," or "the response is too slow." Or after reading the level indication in a tank, he may say that "the level is low." He has his own representation for the concepts he handles—"low," "too slow," "well tuned" and the like; and his own mental reasoning process to handle such information.

An Intelligent Control system needs to have the same ability to handle abstract symbolic information. None of the existing AI technique can handle symbolic information with the richness of a human operator. As Rao (1992) suggests, a promising approach is integration of the different techniques available.

3.2.3 Dealing with Uncertain and Incomplete Knowledge

AI provides tools such as fuzzy sets, fuzzy logic, subjective probabilities and qualitative modeling to design an intelligent system with the capacity to handle approximate knowledge. But why bother with uncertainties and imprecise knowledge in engineering, when we all know that "engineering procedures require exact techniques and that there is no room for imprecision or intuition"? An operator would readily agree, indicating that "the reading of an instrument is a specific and precise value," and "the aperture of a valve to be closed to a certain percent opening will also be a concrete value."

It is true that physical components have "precise" status—a motor is either ON or OFF, for instance. Variables also have "concrete" values—if a power draw indicator says 3000 hp, the operator cannot say that the power draw may also be 2800 hp. However, the above examples could be somewhat misleading concerning the presence of uncertainty and approximation in an industrial environment. The fact is that even in the case when the operator has read the exact value of a variable such as the power draw being 3000 hp, his mental representation of this information adds immediately an

inherent degree of vagueness and imprecision to this information. For instance, after the reading, he may say that "the power draw is too high" and use this new representation in conjunction with his own set of heuristic rules during his decision-making process. This reasoning strategy may lead him to the conclusion that he has to "reduce fresh feed to the mill," which is also highly imprecise.

Does an operator have a complete mathematical model of the process he is dealing with? Certainly not. What he has is the ability to use a partial and approximate representation of the reality of his concern to draw conclusions. How does a grinding mill work? What are the exact phenomena taking place in the size reduction of the ore? Not even an expert may have a complete answer to these questions. However, an operator may know that "if he cuts the fresh feed to a mill, the power draw will likely decrease," enough knowledge for him to take an appropriate action when the power draw is "too high," for instance.

3.2.4 The Need to Learn

If a computational system can handle symbolic knowledge, we immediately refer to it as an "intelligent system." Current intelligent systems perform well as long as the problems belong to the class for which they were designed. However, they may fail miserably for other kinds of problems or when they deal with a situation not foreseen in the design stage.

Learning is one of the basic features that gives a system the ability to go beyond a mere execution of what was programmed at the design stage. A system without this capability should be referred to as a "non-intelligent" (or perhaps, mentally deficient) system. A definition of learning favored by AI researchers is: "Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more effectively the next time" (Partridge and Paap, 1988).

When an adaptive controller modifies its parameters to respond to changes in the process or environment, we may say that it has learned, in a limited sense. Within the context of intelligent control, learning means creating knowledge structures, not just parameter adjustments. Learning here is equated with building, modifying, or improving descriptions. These descriptions can be in the form of declarative statements, procedures, control algorithms, simulation models, or theories.

3.2.5 Making Decisions on Time

One important aspect to bear in mind, is that we are dealing with a time-varying process that does not wait for us to think of a better action—no action is appropriate if it is not applied "on time." This has led us to develop the concept of "real-time" operation for a computer system, i.e. a computer that performs tasks at specific time instants, or has access to (real) time values. For instance, to calculate the rate of change of a variable the computer needs to know the exact time elapsed between measurements.

A distortion of the concept of real-time has led us to think that if a system is to operate in real-time, it needs to respond in milli-seconds—or, hopefully, instantaneously. We agree that a human operator works on "real-time," but does he respond in a fraction of a second? Well, not always. Does he always need to respond in a fraction of a second? Not really. In some situations he may respond quickly whereas in others he may take longer (due to the process dynamics involved) to evaluate and to determine the need for an action. The important point is that his actions should be taken "on-time." How long does it take, for instance, for an operator to determine if the increase of the percentage of solids in a grinding mill is beneficial or not? Does he need to react instantly?

With this in mind, a better requirement for a supervisory system is that the decision of taking an action be made "on time." The system needs, however, to handle the concept of time and its (real)

value in order to deal with temporal reasoning such as in the calculation of the rate of change of variables.

3.3 Elements of Artificial Intelligence (AI) in Intelligent Control (IC)

3.3.1 Knowledge Representation (KR)

Besides the well known quantitative mechanisms for the representation of information, used by conventional control approaches, IC offers a rich variety of techniques taken from the field of AI to represent symbolic knowledge about processes and operators' heuristics. These techniques partially emulate human mechanisms for mental representation of knowledge.

i) KR Using Rules (Production Systems)

The most common knowledge representation technique employed by IC researchers is IF-THEN rules. It is argued, and to some extent accepted, that the knowledge handled by humans can be structured as a set of facts or assertions about the real world, and a set of heuristics of the form of IF-THEN rules to reason upon that knowledge. Facts about the process or environment may be represented symbolically by:

Property (object, value)

These facts could come from a translation of quantitative process measurements or could be reported directly by another operator. After reading a flowmeter, the operator may interpret or represent the information read by saying: "the water flowrate is too low," which can be symbolically represented by: **flowrate** (*water*, **too-low**). IF-THEN rules are intended to capture heuristic or experiential knowledge an operator has, and may take one of the general forms:

IF *antecedents* THEN *consequences*

IF *premises* THEN *conclusions*

IF situation THEN action

A rule related to the example of the water flowrate may be something like this: "IF the water flowrate is too low THEN we may have problems pumping the slurry."

ii) KR Using Frames

A frame is a way to represent knowledge about objects and events common to a particular situation. The elements of a given situation are stored as entries in the slots of a frame. This representation gives an object-oriented approach to the knowledge represented. Thus the knowledge can be easily structured and different techniques such as property inheritance can be used among these objects. Figure 3.3 shows an example of a frame describing a specific process pipe that transports slurry.

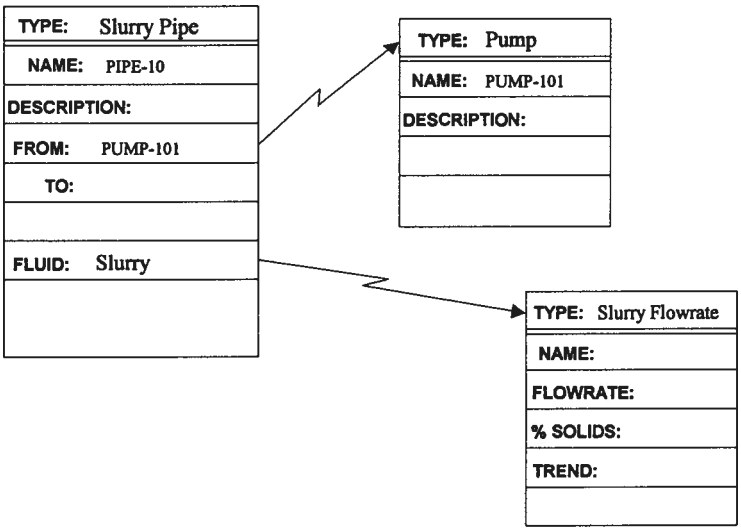


Figure 3.3 Example of frame-based knowledge representation.

It is interesting to notice that the frame for PIPE-10, shown in Figure 3.3, can be used for any pipe transporting slurry. Another important aspect is that a slot may also be treated as an object and be described by a frame, as in the case of slurry flowrate shown in Figure 3.3.

In some development tools, the organization of knowledge or facts into "classes" of facts, is a modified type of a "frame."

iii) KR Using Qualitative Models

Qualitative modeling, treated extensively in chapter 5, has been used by IC researchers to deal with incomplete knowledge and with approximate and low resolution reasoning (Abdulmajid and Wynne 1991; Cifuentes and Mular 1992; Corea et al. 1992; Leitch 1988). This approach is useful in situations where one needs only a rapid and rough estimate of what behavior is possible, rather than a very precise prediction involving quantitative mathematical models.

Qualitative modeling provides elements to represent continuous properties of the real world by means of a discrete and finite system of symbols. Only the key factors influencing the behavior of the reality are captured, so the models obtained have the desired high degree of abstraction. In this formalism both variables and their relationships have to be described qualitatively.

One of the formalisms to represent qualitatively a variable is sign ontology. Under this formalism a variable is represented by the **sign** of its **amount** and the **sign** of its **derivative**. Thus only three values are distinguished on a variable: **positive, zero** and **negative**; and we can only say that a variable is **increasing, constant** or **decreasing**. Nothing more specific can be said about it. The relationship between variables is expressed by "qualitative relationships."

As an example, let us consider a qualitative model (QM) for the relationship between the power draw and the fresh feedrate of solids to the mill. This relationship could be represented by the qualitative proportionality:

$$P\_draw \;\; \alpha \;\; FF\_solids$$

read as: "the power draw on the mill is qualitatively proportional to the fresh feedrate of solids to the mill." In algebraic notation this would be written as $\boldsymbol{P\_draw = f\,(\ldots, FF\_solids, \ldots)}$, with f an unspecified monotonic function.

If we are told that the fresh feedrate of solids, *FF_solids*, is increasing, we could immediately deduce that the power draw, *P_draw*, will also increase, provided we have not exceeded the maximum in the power versus feedrate curve. There is no numerical value involved, and there is no need for more specific information to determine the type of change that the power draw will have. This is one of the important features of qualitative models.

iv) KR Using Semantic Nets

A tree or graphic structure is typically used to describe relationships between objects, often for the general purpose of questioning and answering. A partial description of a grinding mill may be represented as shown in Figure 3.4. From this representation, we could derive for instance that: "the power drawn by the mill is **pd-value**, and that this power draw is proportional to the fresh feedrate of solids to the mill."

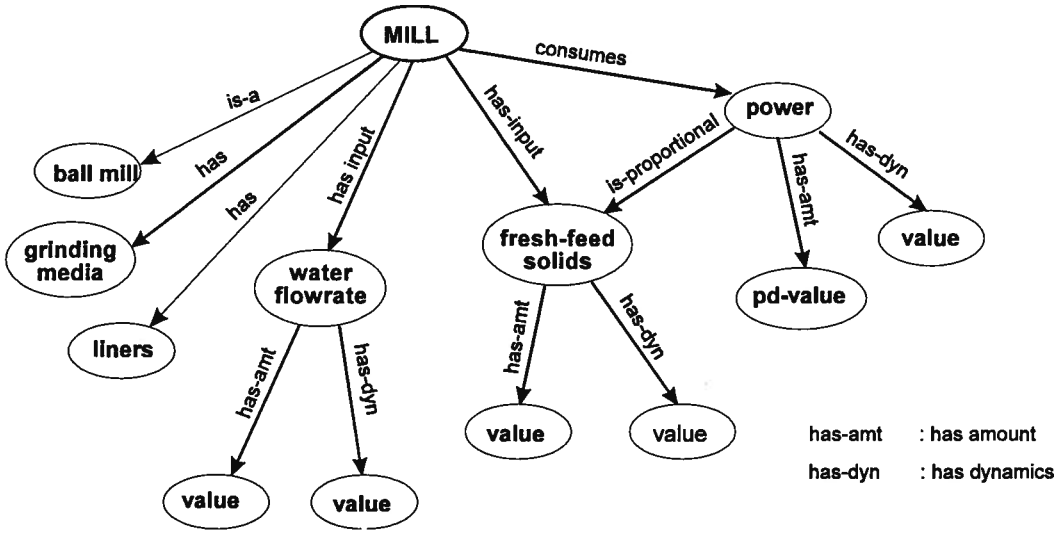


Figure 3.4 Example of semantic net representation.

v) KR Using Fuzzy Sets

Fuzzy set theory gives us the elements to deal with vagueness and uncertainty in knowledge. Human operators' language is full of fuzzy concepts such as "high," "very-low," "hard," "coarse" and so on. For instance, after reading the value of the power draw of a mill, the operator may say that, "the power draw is *high.*" He may have a range of values for which he says that the power draw is "high," with some degree of certainty. As the power draw increases his certainty in using this concept also increases to a point at which he may say that "the power draw is *definitely high.*" To represent the vagueness of the concept "high," within the context of the example, a possibility distribution of the form

$$\textbf{Poss}\{\, high = j\} = \mu_{high}(j) \qquad j \in J, \text{ the set of power draw values within "high"}$$

associated with the value of the power draw, **j**, being "high" is chosen. This possibility distribution is considered to be numerically equal to a membership function $\mu_{high}(j)$, with **J** being the domain of the values of "high" (see Figure 3.5). For values greater than J_1, the power draw is considered to be

"definitely high," whereas for values less than J_0, it is said to be "definitely not-high." In other words, a degree of truth (belief) between 0 (completely false) and 1 (definitely true) has been assigned to the proposition "the power draw is high."

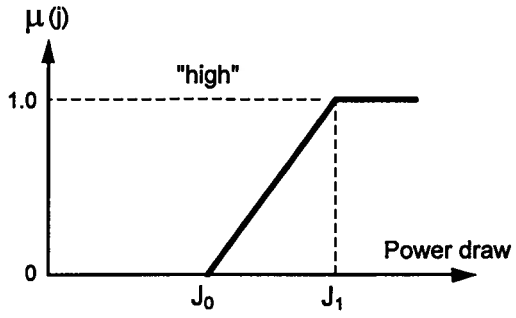


Figure 3.5 Fuzzy representation of "high" for the power draw.

3.3.2 Handling Uncertainty and Incompleteness

Problem solving and decision making by people are often done in environments where the information concerning the problem is partial and approximate. These deficiencies in knowledge may be due to factors such as: partial or unreliable information, inherently imprecise representation language and conflicting information from multiple sources.

AI researchers have proposed several approaches to represent and deal with uncertainty. However, the only one that seems to have reached the IC world is fuzzy logic with its possibility theory (Harris and Meech 1987; Karr 1991; Sugeno 1985; Peters et al. 1992; Berenji 1990; Dubois and Prade 1988). Qualitative modeling is another important AI technique considered to handle approximate knowledge (Forbus 1984 & 1986a; D'Ambrosio 1989a & 1989b).

Fuzzy logic, as proposed by Zadeh (1988), provides the means for dealing with imprecise predicates such as "high," "low," "hard," etc. A heuristic rule involving, for instance, the proposition "the power draw is high" from the previous example shown in Figure 3.5 may be given by:

IF *power draw is high* **THEN** *the ore is hard* (3.1)

Given this IF-THEN rule, we obtain a conditional possibility distribution for the hardness of the ore given the evidence that "the power draw is high," written as $\mu_{h \times j}$ and given by (see Tong 1977; Andersen and Nielsen 1985):

$$\mu_{h \times j}(j, h) = \min [\, \mu_j (j) \wedge \mu_h(h)] \qquad j \in J \,\&\, h \in H \tag{3.2}$$

where μ_j (j) and μ_h(h) are the membership functions for "power draw high" and "the ore is hard" respectively; and **J** and **H** are the sets of possible values for the power draw and ore hardness.

The output of rule (3.1), i.e. the conclusion "the ore is hard," will also be a possibility distribution similar to that of Figure 3.5 for the power draw, given by:

$$\mu_{h/j}(h) = \max_{j \in J} \{ \min [\, \mu_j (j) \wedge \mu_{h \times j}(j, h) \,] \} \tag{3.3}$$

Different authors use different approaches to deal with fuzzy sets and fuzzy logic; they use different "fuzzy" operations to obtain the results from a rule (Zadeh 1988; Dubois and Prade 1988; Tong 1977; Harris and Meech 1987; Andersen and Nielsen 1985; Isaka 1990; Karr 1991).

Another interpretation of the certainty factors (degrees of belief) associated with propositions and rules is that they account for incompleteness in our knowledge. If we assign a degree of belief 0.7 to the rule given in (3.1), for instance, we may interpret it by saying that we are not absolutely sure that the only reason for the high value of power draw is due to a hard ore. Or, from another perspective, we may say that this rule is the simplest way to estimate the hardness of the ore—and it is only a rough approximation.

In general, a heuristic rule is a partial representation of a situation. We may know that there are other factors contributing to that piece of knowledge, but due to unknown effects of those factors, or because we know that their contribution is not significant and we want to work with simple rules, or because their effect has been implicitly considered, we choose to assign a degree of belief to the rule representing the incompleteness of that piece of knowledge.

3.3.3 Learning Techniques

Most of the learning techniques used in IC fall within the category of "empirical learning." Within this category we can distinguish symbolic learning, evolutionary (genetic) learning, and connectionist learning. The main characteristic of these techniques is that they are based on knowledge-poor inductive inference. "Analytical learning" techniques are still within the AI domain. These techniques are knowledge-intensive and require background knowledge; the basic inference type they use is deduction. The knowledge learned is basically a new representation for the input information; and not necessarily new knowledge.

Empirical learning can be described as a framework where the real and ideal behaviors of the system are compared and the system is then modified according to the difference between these behaviors (see Figure 3.6). Some of these techniques can be considered search or optimization techniques using specific algorithms such as evolution theory, genetics or neuronal operation (connectionist). There is a goal or ideal behavior and the system is continually modified to reach the goal (Forsyth and Rada 1986). Karr (1991) uses a genetic algorithm to modify definitions of fuzzy sets and thus modify the performance of a fuzzy controller. Connectionist approaches are used for learning control applications to robot operation (Miller and Hewes 1988; Isik and Ciliz 1988). An *ad-hoc* search technique is used by Crosscope and Bonnell (1988) to modify a "learning network" which can be used to model a plant for control purposes.

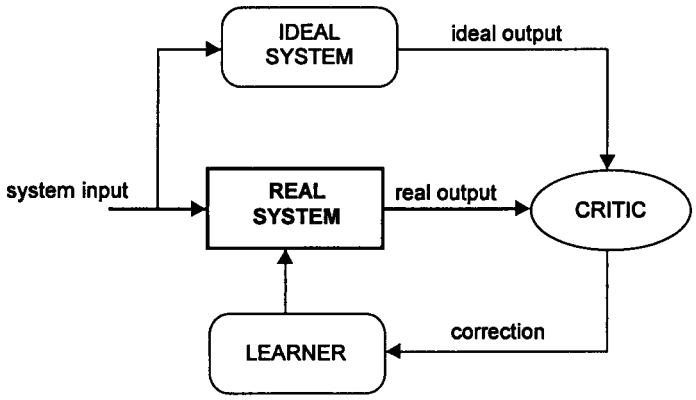


Figure 3.6 General structure for empirical learning techniques, (adapted from Forsyth and Rada, 1986).

IC researchers are cautious about using learning techniques that go beyond parameter learning. The main reason is that learning techniques are still in their infancy. Hence, it is risky at this stage to develop an application in which the system autonomously learn new rules on how to control a plant. It is much safer when the user defines the structure of the knowledge and the learning technique determines parameters only—which is fairly close to adaptive control.

CHAPTER 4

DESIGN OF AN INTELLIGENT SUPERVISORY CONTROL SYSTEM (ISCS)

4.1 General Structure

The approach proposed for the design of the intelligent supervisory control system (ISCS), is that of a supervisory system sitting on top of a regulatory control system (RCS) as shown in Figure 4.1. The ISCS deals with functions such as: monitoring the operation of the RCS; deciding setpoints; choosing operating modes for controllers; analyzing the operation of main components of the circuit as well as global operation; and supporting special tasks such as start-up and shut-down.

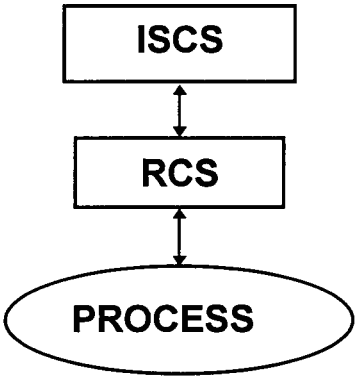


Figure 4.1 Approach proposed for the ISCS.

The design of the ISCS provides the flexibility to deal with different types of RCS. The ISCS provides the tools and mechanisms to implement the supervisory functions as required by a specific RCS. In general, the ISCS has access to the database of the RCS to read the information needed and to implement actions resulting from its decision-making process. These actions may be new values of setpoints for controllers, or direct command to equipment or final control elements.

The internal structure of the ISCS is conceived as composed of two distinctive levels of operation: a higher meta-level in a coordinating role and a lower level with two modules, one to handle symbolic knowledge about the process being supervised and the other to handle quantitative processing such as numerical algorithms and mathematical calculations (see Figure 4.2).

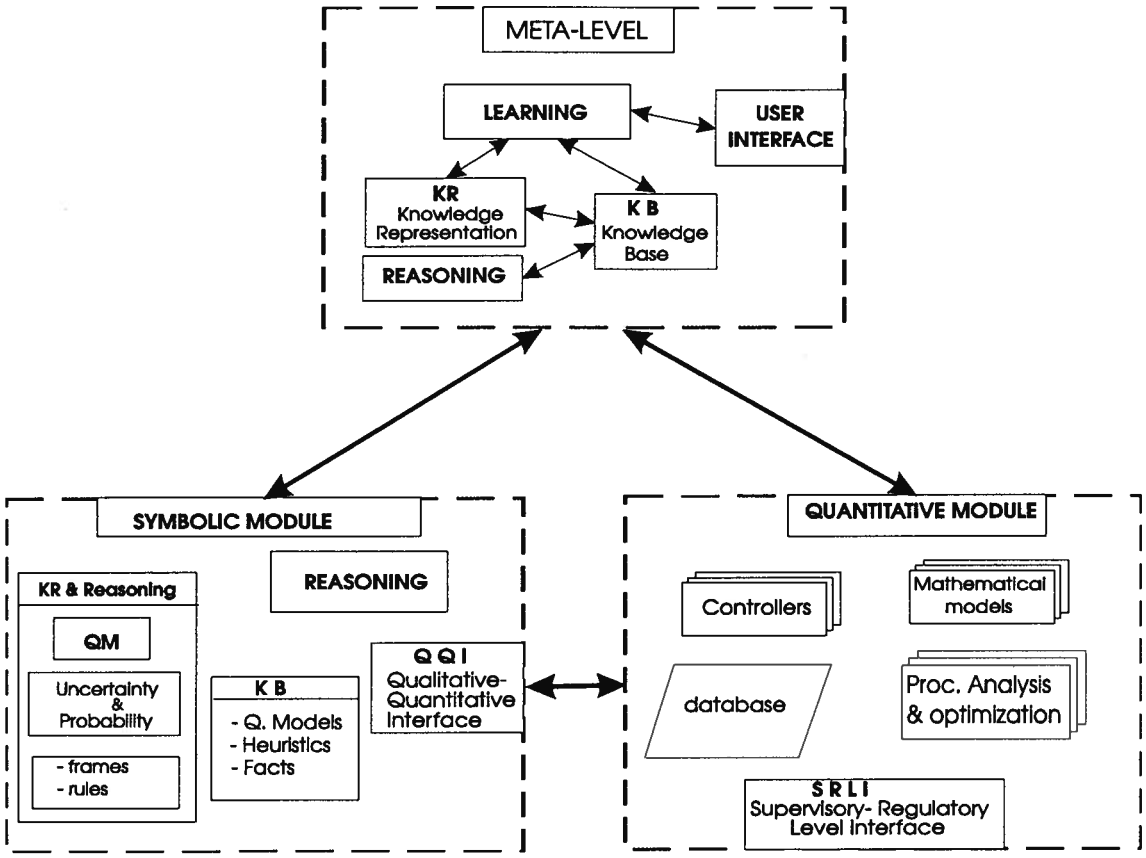


Figure 4.2 General structure for the Intelligent Supervisory Control System (ISCS).

The meta-level, rich in symbolic knowledge and abstract concept representation, coordinates the operation of the modules at the lower level. It has knowledge regarding overall strategies to supervise a process and meet goals; knowledge of conditions under which a particular control algorithm can operate; and knowledge regarding the integration of qualitative and quantitative mechanisms. It also provides the mechanisms to interact with the user. The symbolic module at the lower level deals with specific supervisory tasks such as monitoring of control loops, fault diagnosis,

special operating procedures, and translation of information between qualitative and quantitative formats. The quantitative module is responsible for any mathematical operation required by the ISCS. It may have, for instance, mathematical models, optimization routines, statistical packages and various algorithms of control. The proper integration of these two symbolic and quantitative modules is an important challenge for the design of the system and a subject for further research.

4.2 General Functional Definition

4.2.1 Global Functions of the ISCS

The main supervisory functions considered in the design of an ISCS are:

i) **Monitoring** the overall operation of the regulatory control system (RCS). The main purpose of this function is to lead the global operation of the process towards specific technical-economical operating goals such as quality specifications, maximum capacity, and maximum efficiency. The actions derived from this function are the determination of setpoint values for the different controllers of the RCS.

ii) **Monitoring** each individual controller of the RCS. The ISCS assesses the overall operation of each controller and decide if there is a need for parameter adjustment. The ISCS possesses specific knowledge to tune individual controllers. The ISCS decides on what type of controller to use according to the prevailing operating conditions. It could also decide to override automatic operation and operate in manual mode.

iii) **Assisting** the operation during special procedures such as start-up and shut-down. The ISCS possesses the knowledge to deal with these special operations, including emergency situations due to disturbances or equipment failure. During these special periods of operation the ISCS

may decide to operate with particular controllers, or use different sets of parameters for the controllers, or even override the RCS.

iv) **Fault detection and diagnosis** of the operation of the RCS. The ISCS has mechanisms to detect failure of process equipment and elements of the RCS. Depending on the results of the diagnostic process, the ISCS may take different actions by itself or advise the operator for specific actions.

v) **Interacting** with the operator. The ISCS has the elements to interact with the operator in a human-like fashion. It has the ability to explain some of its conclusions or actions to the user, for instance. The ISCS also provides the mechanisms for the user to modify knowledge and procedures currently employed.

vi) **Intervening.** The ISCS has the mechanisms to modify parameters, relevant points in the database, and configuration of elements in the RCS.

Communication with the RCS is obtained through direct access to the database of the RCS. The database includes working data used by the RCS such as analog and digital input/output, as well as parameters for controllers or other algorithms. Calculated variables such as trends, needed by the ISCS, may also be directly read from the database. However, more specialized processing such as material balances, may need more interaction with the ISCS so they may be processed at the lower quantitative level within the ISCS.

4.2.2 Supervisory Control Tasks

The major supervisory control tasks associated with the operation of the ISCS are as follows (refer to Figure 4.2):

i) Meta-Level

- **Coordination.** Coordinate the operation of the symbolic and quantitative modules at the lower level. Decide on an appropriate strategy to solve a problem. Assign tasks to the different modules involved in the solution of a problem.

- **Quantitative-Qualitative Integration.** Provide smooth integration of the two modules with mechanisms for transferring and translation of information.

- **User interface.** Interact with the user in a human-like manner providing access to specific information, explanations for actions taken, and mechanisms to access and modify the knowledge base as needed.

- **Concept learning.** Handle abstract concepts and build a hierarchy of more complex concepts as the system interacts with the user.

ii) Symbolic Module

- **Overall RCS supervision.** Monitor and guide the operation of the RCS as a whole to satisfy global operating tasks such as quality specifications, maximize production, or minimize discarded material.

- **Individual controller supervision.** Monitor the behavior of each controller providing the mechanisms for assessing their behavior, modifying parameters, changing operating modes and overriding automatic operation.

- **Process analysis.** Evaluate the operation of units, circuits and the overall plant, as part of an ISCS supervision task, or under direct request by the user.

- **Equipment operation.** Assess the operation of each piece of equipment as needed by other supervisory tasks, (determine if a tank is overflowing, or a valve is plugged, for instance).

- **Fault diagnosis.** Detect abnormal operation and diagnose possible causes such as defective equipment responsible for the anomaly observed.

- **Special operations.** Guide the operation of the process during special operating periods such as during start-up or shut-down, or when process layout is being modified.

- **Quantitative Interface.** Translate quantitative information to the appropriate qualitative representation, and vice versa, as required by the symbolic module, the meta-level or the user.

iii) Quantitative Module

- **Controller operation.** Provide a bank with different algorithms of control, their associated identification and parameter estimation mechanisms, and basic indices to assess their operation.

- **RCS interface.** Handle the transference of information between the ISCS and the RCS and the updating of the database on the ISCS.

- **Mathematical models.** Provide mathematical models of different process units and the environment. Simulate these models and make the results available as required during the operation of the ISCS or directly by the user.

- **Optimization routines.** Provide numerical optimization algorithms in support of the overall process optimization functions of the ISCS, or as required by the user.

- **Process analysis.** Analyze the operation of the process through balances or statistical analyses as required during the operation of the ISCS or directly by the user.

4.2.3 Regulatory Control Tasks

Although not part of the design of the ISCS, a brief description of the tasks associated with the RCS are mentioned here. The main tasks the RCS is responsible for are:

- **Continuous Process Control.** Keep control variables operating on setpoints or targets despite disturbances affecting the operation of the process.
- **Batch or Sequential Control.** Follow operating procedures and steps according to a specified sequence or operating program.
- **Interlocking.** Provide the mechanisms of interlocking between different equipment involved, facilitating operations such as start-up, shut-down, or emergency procedures.
- **Alarming.** Provide the basic mechanisms for detection of abnormal situations such as variables out of range, and mechanisms to communicate this to the plant operator and to the ISCS.
- **Manual Operation.** Provide the mechanisms for a direct manipulation of equipment and final control elements by the operator.

4.3 Hardware and Software Requirements

4.3.1 General Software Specifications

There are two basic options regarding the software available to build an ISCS. These are:

i) Use an expert system (ES) development software package, which has most of the capabilities required for building an ISCS, and which provides a high level programming interface for the development of the remaining facilities or characteristics required.

ii) Use an artificial intelligence (AI) programming language such as LISP or PROLOG -including C++; or use a specialized software package available in the market such as Knowledge Kraft or Envisage. The latter alternative provides different facilities especially useful at the development stage (e.g. a versatile user interface, debugging capabilities, etc.).

The major requirement for a software package, essential to supervisory control applications, is the need to handle the concept of time: the software package should handle temporal reasoning and have the ability to perform tasks in real-time fashion, e.g. sampling process variables at specific time intervals. Another requirements is the need for an open architecture in order to allow custom-modification if needed, e.g. integration of user-developed modules with those already available in the software package. A final requirement is the need for interconnection with the RCS level, which usually is running on a different machine.

4.3.2 General Hardware Specifications

Special computers have been designed for use in AI supporting languages such as LISP. LISP machines include the Symbolics 3600 series machines (Symbolics, Inc.), the Lambda series machines (LISP Machines, Inc.), the Xerox 1100 series (Xerox Corporation), and the Explorer (Texas Instruments Inc.). These manufacturers have their own versions of LISP, which are native to their machines.

The processing power of these specialized LISP machines has made them the ideal choice for large-scale, LISP-based projects. Many of the medium and large ES shells were originally developed to run on these type of machines. However, the recent trend is to modify those original products or offer new ones running on conventional machines. Economical reasons as well as the desire to reach other potential users and applications have been the causes for such a trend. Thus, nowadays, there

are many ES shells and development environments running on conventional mini and microcomputers.

The specific type of computer will depend on the software package selected for the development of the system as well as on the requirements of the specific application.

4.4 Potential AI Techniques Involved in the Design of an ISCS

There are numerous techniques available that deal with different aspects of artificial intelligence, and researchers continue proposing new ones. Only a few of these techniques are being used by researchers in intelligent control (IC); the majority still remain within the academic boundaries of the AI world.

The focus of this research project is qualitative modeling and its potential integration into the design of a real-time ISCS. Qualitative models provide an ISCS with the mechanisms to handle and reason upon heuristic knowledge about behavior of physical elements. The next chapters describe in detail a pseudo-qualitative modeling approach, its incorporation into an ISCS prototype, and the application of this prototype to a full scale industrial plant for validation of the approach proposed.

In addition to qualitative modeling, two other AI aspects, namely uncertainty handling and learning—also potential candidates for incorporation into the design of an ISCS—were briefly studied in the course of this research (see sections 3.3.2 and 3.3.3). Appendix A presents some of the major aspects that resulted from these additional studies. Time constraints have contributed to the non-implementation of these components at this stage; they are left for future research projects.

A simplified framework to handle uncertainty based on a subjective interpretation of probability is proposed and described in Appendix A.1. Under this interpretation, probability is viewed as the actual degree of belief in a given proposition held by some real individual at some specific time

(Kyburg and Smokler 1964; Weatherford 1982). This approach overcomes major limitations found in the application of a more traditional interpretation, which views probability as a relative frequency of a number of repetitive events.

Learning theory and available learning techniques are too immature to work with critical paths of knowledge within an ISCS. Thus, an alternative approach proposed in this study is the incorporation of learning to enhance the user interface (see Appendix A.2 for details). Under this approach, the system does not intervene directly with the process; instead, it learns how to handle concepts involved in the interaction with operators. A user interface rich in high-level concepts gives an opportunity to establish a more "intelligent" communication with the system.

CHAPTER 5

PSEUDO-QUALITATIVE MODELING APPROACH

5.1 Introduction

A central aspect of human reasoning is the representation of knowledge and the representation of the physical world we need to deal with. People, it has been said, walk around with a mental representation of the external world in their heads. This mental model allows them to try out various alternatives in a given situation before taking a final decision, to react to future situations before they arise, to utilize the knowledge of past events in dealing with the present and the future, and to learn from their experience so they can deal in a better way next time they face a similar situation. Scientists have been puzzled by human reasoning for ages, trying to find out what sort of representation and reasoning mechanisms people employ.

Mathematics is one of the modeling techniques most commonly used in formal scientific studies, especially in engineering fields. A literature review will show that there is not a single area of knowledge without mathematical models, representing at least the most important and fundamental aspect of the field. Researchers agree, however, that these are not the kinds of model people use in their everyday life. People seem to reason fluently about many phenomena without any information on the mathematical relationship that may describe these phenomena.

An example of the above situation can be found in the operation of an industrial plant. It is not uncommon to find that an operator has overridden an automatic control system which was unable to cope with particular operating circumstances. Does the operator have a mathematical representation

of the behavior of the process?—very unlikely. For example, an operator only needs to recognize the possibility of overflow in a tank to quickly decide on an appropriate corrective action. He does not need to spend time using a full mathematical model to predict when an overflow will occur or how much liquid will spill or how big the resulting puddle might be. This ability to handle partial, incomplete information and to reason at a low level of resolution, is one of the key characteristics that we need to incorporate into an intelligent computer-based system.

5.2 Alternative Techniques to Qualitative Modeling

Formalisms, such as rules, frames, and semantic networks used in today's expert systems are important alternatives to handle knowledge in the way mentioned above (Fischler and Firschein 1987; Bowerman and Glover 1988, Luger and Stubblefield 1989). However, due to the explicit characteristics of these formalisms, i.e., every possible behavior that may occur must be deduced and explicitly represented, the models obtained are referred to as "shallow models." For example, a rule-based model to represent the input/output behavior of a process may consist of a set of rules of the form:

{ IF *input* **condition** THEN *output* **behavior** }

This set of rules needs to consider every possible **condition** for *input* such as **high, low, increasing,** or **decreasing.** For each of these conditions we need first to determine the resulting behavior of the *output,* and then write down the corresponding rule. Thus, the model obtained is an explicit list of **condition-behavior** pairs associated with the operation of the process. In general, a different set of rules has to be obtained for each particular process being modeled.

Researchers in Qualitative Physics have proposed various formalisms that exhibit suitable characteristics to handle incomplete information (Forbus 1985, 1986a & 1988; de Kleer and Brown

1985; Kuipers 1986; Leitch 1990). A qualitative model (QM) built using these formalisms may be viewed as a "deep model," i.e., although the relationship between variables is explicit, the behavior of the output is unknown until a simulation of the model is performed. Under one of these formalisms, the input/output behavior of a process could be expressed as:

$$input \; \alpha \; output$$

read as "the *output* is **qualitatively proportional** to the *input*." This qualitative proportionality α, is a representation of the phenomena that define the relationship between these two variables. A qualitative simulation will then solve this model and predict the behavior of the *output* according to the values the *input* may take.

Several approaches have been proposed within qualitative physics to obtain a general framework to represent a physical reality in a qualitative fashion. De Kleer and Brown (1985 & 1986), and Williams (1984) describe their qualitative models in terms of "components" and "connections." Kuipers builds qualitative models starting from mathematical descriptions, and then he uses his own simulation algorithms to obtain predicted behavior (Kuipers 1984, 1985 & 1986). Bhaskar and Nigam (1990) use dimensional analysis as a method for qualitative reasoning; they do not require an explicit knowledge of the physical laws, but only knowledge of the dimensional representation of the relevant variables. Other researchers propose a semi-quantitative representation combining qualitative and quantitative reasoning as an alternative to pure qualitative reasoning approaches (Widman 1989; Simmons 1986). Finally, Forbus proposes a "qualitative process" ontology to deal with qualitative reasoning. In this ontology any physical reality is seen, in general, as a "process" with inputs and outputs through which it interacts with other processes and with the environment (Forbus 1984 & 1986b).

5.3 A Pseudo-Qualitative Approach to Qualitative Modeling

A pseudo-qualitative approach to qualitative modeling is proposed in this research. Under this approach, a QM is defined as a mathematical model with a hybrid structure capable of handling both numerical and qualitative information (see Figure 5.1).

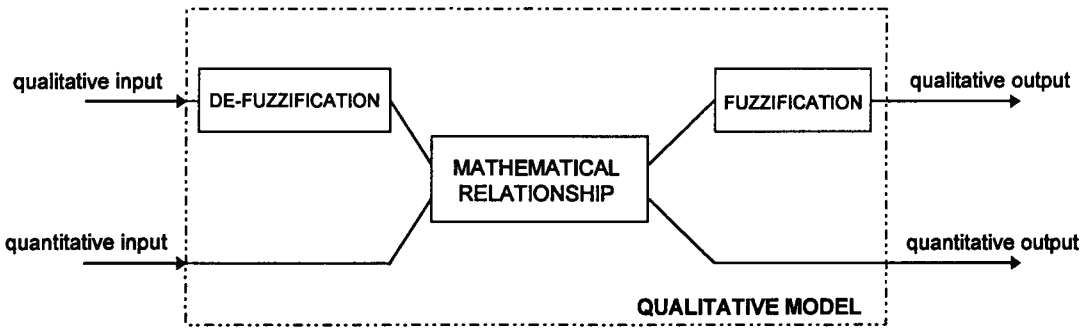


Figure 5.1 General structure of a QM under the pseudo-qualitative approach proposed.

A de-fuzzification algorithm is used when the information received by the model is purely qualitative; no pre-processing is required when the information received is quantitative. The information is mathematically processed to obtain the output of the model, which is presented in both qualitative and quantitative formats.

The accuracy of the QM is determined by the requirements of the specific application and by the information available to build the model. An application may only require representation of simple heuristic knowledge such as: "the output of the process is proportional to its input." In another case, we may need to represent a more accurate input/output behavior given by a specific mathematical expression.

As shown later on in this section, a QM under this pseudo-qualitative approach offers the same level of flexibility as a QM using a pure qualitative approach. The use of mathematical relationships as

the core of a QM does not impose any additional constraint. In fact, this pseudo-qualitative approach offers considerable advantages over its purely qualitative counterparts as discussed in section 5.5.

The major elements of this QM approach, described below, are: variable representation, fuzzification and de-fuzzification, steady state QM primitives, and dynamic QM primitives.

i) Variable Representation

A process variable is represented in both quantitative and qualitative formats. The qualitative value is represented by a set of qualitative terms that describe both the sign and dynamic trend of the variable. Figure 5.2 shows the set of qualitative terms proposed and their definition.

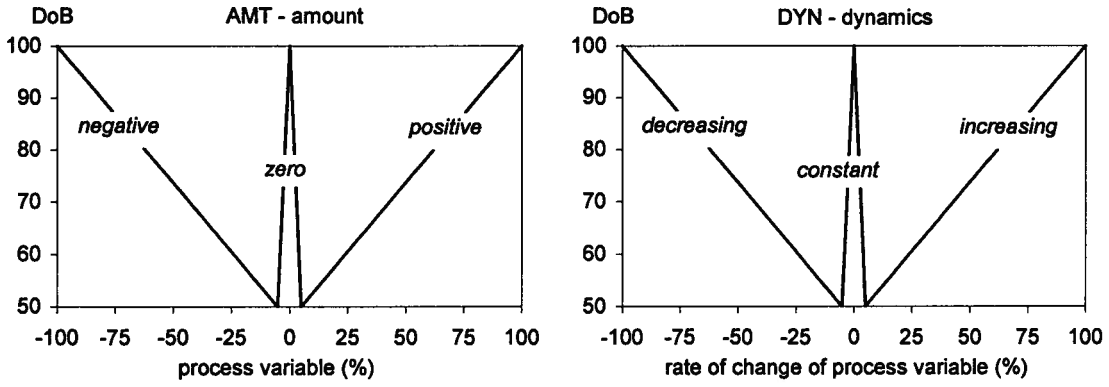


Figure 5.2 Fuzzy Sets that define qualitative information of variables.

Each of the qualitative terms shown in Figure 5.2 corresponds to a fuzzy set that associates the numerical value of the variable with a degree of belief (DoB). The DoB, in the context of pseudo-qualitative modeling, is a numerical value between 50 and 100. The set of qualitative terms shown in Figure 5.2, together with DoB values, suffice to uniquely specify the numerical value of a variable. The numerical value of variables is expressed as a percentage of the range of values that a variable may take.

Using this formalism, for example, a variable *V* that is at steady state and whose value is 50% can be qualitatively expressed as

AMT(*V* ***positive*** DoB: 74.4) DYN(*V* ***constant*** DoB: 100)

is interpreted as a person's certainty about the appropriateness of using a particular qualitative term to express the value of a variable. For example, if we indicate that the value of a variable is "high" and we are absolutely certain about that, we would assign a DoB equal to 100 to that assertion. A DoB equal to zero would have indicated our complete certainty that the variable is "not high."

The internal 3-term representation of a variable shown in Figure 5.2 can be directly translated into any user-defined set of qualitative terms such as "*positive low*," "*positive medium*" and "*positive high*." A user-defined representation may also involve a different interpretation of a DoB, such as that where a DoB is interpreted as a person's certainty about the appropriateness of using a particular qualitative term to express the value of a variable.

ii) Fuzzification/De-Fuzzification Procedure

In the context of a supervisory system, most of the information required by a QM is read directly from the process database; hence, it is already available in a numerical format. If the information is received in a qualitative format, it would need to be de-fuzzified before being used by the QM. The first step of this procedure consists of mapping the information available onto the set of qualitative terms shown in Figure 5.2. The second step consists of direct translation of these terms into numerical values using the same qualitative definitions shown in Figure 5.2.

The fuzzification procedure, i.e., to convert a numerical value into qualitative terms, is similar to the de-fuzzification procedure indicated above but in the reverse order.

iii) Steady State Primitives

The two primitives proposed to build steady state QMs that describe the input/output behavior of a process are:

output = **qm_function** (*input*) ; **qm_function** is a user-defined mathematical relationship between the *input* and the *output* of the process.

qm_table ($input_k$, $output_k$) ; **qm_table** is a look-up table that associates input/output pairs of the form ($input_k$, $output_k$).

Let us assume that we want to build a single-input, single-output steady state QM of a process. If the only information available indicates that the output is "proportional" to the input, we could write a QM based on the **qm_function** primitive as

QM_1: $output = K \bullet input$ (%) ; K= 1 (default value)

K can be tuned, if needed, when new information about its value is available.

If the input/output relationship is known to be a quadratic one, the QM would be written as (see Figure 5.3).

QM_2: $output = K \bullet (input)^2$ (%) ; K = 1/100 (default value)

K can be adjusted to fit any specific input/output quadratic behavior.

In general, the user can define **qm_function** as the most appropriate mathematical expression that represents the known input/output behavior of the process. In that case, the QM would take the form

QM_3: *output* = **qm_function** (*input*) (%) ; with **qm_function** a mathematical expression.

The second primitive, **qm_table**, allows the user to define the steady state behavior of a process as a set of input/output pairs, without specifying a mathematical relationship. A quadratic input/output behavior, as represented by a **qm_table** with three input/output pairs, **qm_table(3)**, could give the results shown in Figure 5.3. As a comparison, Figure 5.3 also shows the results of modeling the same behavior with a quadratic **qm_function**.

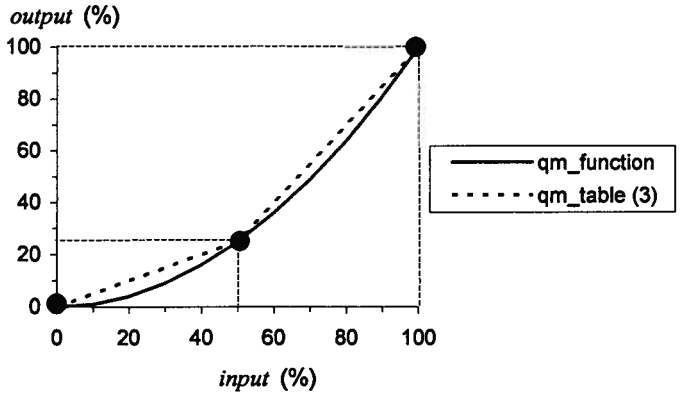


Figure 5.3 Quadratic input/output behavior represented qualitatively using primitives **qm_function** and **qm_table(3)**.

In the case of an application that does not require a high degree of accuracy, the results obtained from the two QMs shown in Figure 5.3 could be considered to be essentially the same.

iv) Dynamic QM Primitives

A dynamic QM provides time-dependent information about the behavior of a process. A dynamic QM would indicate if a variable is "constant," "increasing," or "decreasing," as well as the value of the output variable at any given time.

A dynamic mathematical model is at the center of a dynamic QM. This mathematical model predicts the evolution of a variable as it reaches a steady state value that has been previously obtained by a steady state QM of the same process. The dynamic QM primitive developed is based on a first order

dynamic model; primitives based on higher order dynamic models could be developed in a similar manner.

The response to a step input of a first order dynamic system with delay takes the form shown in Figure 5.4. This response can be mathematically written as

$$output(t) = Out_{ss}(1 - e^{-(t - t_d)/\tau}) \qquad ; t \geq t_d \tag{5.1}$$

where t_d : delay time

τ : time constant, time required for the output to reach 63.2% of the final steady state value

Out_{ss} : final steady state value that the output reaches

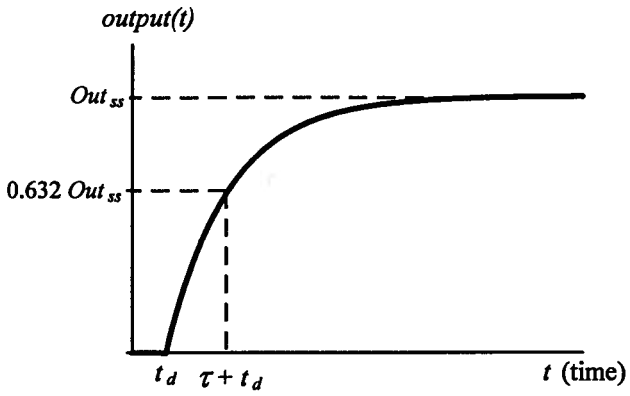


Figure 5.4 Behavior predicted by a first order dynamic model as a variable evolves from one steady state value to another.

The recursive form of equation (5.1) that gives the value of *output(t)* at each sampling time is (Franklin G.F. and Powell J.D. 1980, chapter 3).

$$Output_k = Output_{k-1}\, e^{-T_o/\tau} + (1 - e^{-T_o/\tau})\, Out_{ss(k-d-1)} \tag{5.2}$$

where $Output_k$ = predicted output value at the current sampling instant

$Output_{k-1}$ = predicted output value at the previous sampling instant

$Out_{ss(k-d-1)}$ = steady state value that the output will reach, as predicted (d+1) sampling instants prior to the current time

d = delay time, expressed as number of sampling instants

T_0 = sampling time (same units as τ)

Equation (5.2) is the final expression of the dynamic QM primitive. The user has to provide only two parameters at the time of building a QM for a specific process: τ, the time constant of the process, and d, the delay time expressed as number of sampling instants. Out_{ss} is provided by a steady state QM of the process.

5.4 Building a QM of a Head Tank

This section presents a case study, based on simulation results, to determine the validity of the proposed QM approach. A head tank with the configuration shown in Figure 5.5 is the process selected for this study. This simple and well-understood process was purposely chosen for this study as it helps to visualize clearly and demonstrate the principles of the approach proposed.

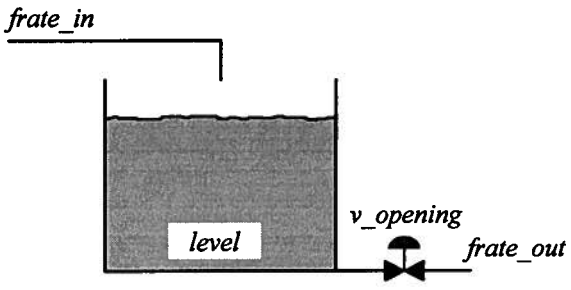


Figure 5.5 Head Tank analyzed in the Case Study.

The variables of interest are ***level***, level of the fluid in the tank; ***frate_in***, flowrate of fluid entering the tank; ***frate_out***, flowrate of fluid leaving the tank; and ***v_opening***, the opening of the valve. ***frate_in*** is considered as the manipulated variable, and ***v_opening*** as a parameter that disturbs the behavior of the tank. This study assumes that measurements of all the variables around the tank are available, including the opening of the valve.

A QM that describes the behavior of ***level*** is proposed and simulated using ProcessVision, a real-time SCADA software package. The source code of the knowledge base associated with this simulation study, as implemented in ProcessVision, is presented in Appendix C. The simulation results from the QM are analyzed against those obtained from a mathematical model of the head tank—the latter assumed to represent the real behavior of the level of fluid in the tank.

i) Steady State Qualitative Model

Let us assume that the only information we have about the steady state relationship between ***level*** and ***frate_in*** comes from heuristic knowledge. If this heuristic knowledge indicates that the level of fluid in the tank is directly proportional to the flowrate entering the tank, and inversely proportional to the opening of the valve, we could write the following model

$$level = K\left(\frac{frate\_in}{v\_opening}\right) \tag{5.3}$$

where K is a parameter that needs to be estimated. An estimate of K could be obtained from plant data or from additional heuristic knowledge about the behavior of the tank.

Now, if experimental data are available, we may be able to use a ***qm_table*** primitive, described in the previous section, to represent the behavior of the level of fluid in the tank. Let us assume that we have information regarding three steady state points for "*low*", "*medium*" and "*high*" values of the

variables. With this information we may represent the behavior of ***level*** by a ***qm_table(3)*** as shown in Figure 5.6, valid for a specific valve opening. Additional QMs for different valve openings could also be constructed and implemented, if required.

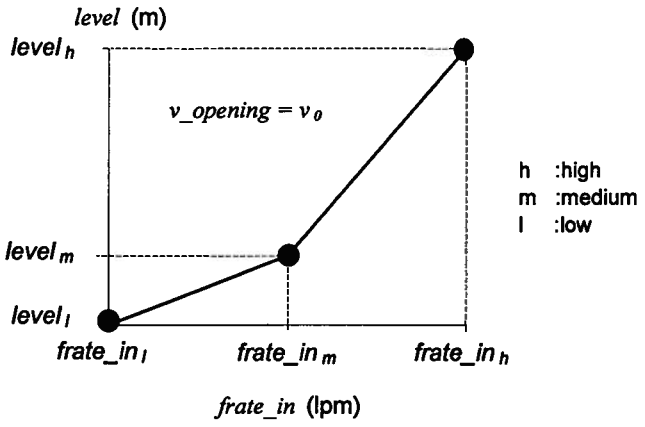


Figure 5.6 QM for the level of fluid in the tank, based on a ***qm_table(3)*** primitive.

As more knowledge about the operation of the tank and about the specific relationships that hold among the variables is gathered, we could update the initial model with more "accurate" ones. For example, if a mathematical model of the tank is available, we may replace equation (5.3) with a cuadratic steady state model of the form:

$$level = \left(\frac{frate\_in}{K_R \cdot v\_opening} \right)^2 \tag{5.4}$$

Equation (5.4) was implemented in ProcessVision using the QM primitive ***qm_function*** described in the previous section.

ii) Dynamic Qualitative Model

Using a dynamic QM primitive of the form presented in equation (5.2), the evolution of the level of fluid in the tank is given by

$$level_k = level_{k-1}\, e^{-T_0/\tau} + (1 - e^{-T_0/\tau})\, level_{ss(k-1)} \tag{5.5}$$

where $level_k$ and $level_{k-1}$ are the level values at the current and previous sampling instants respectively; $level_{ss(k-1)}$ is the steady state level value predicted by the steady state QM of equation (5.4) at the previous sampling instant; τ is the time constant of the tank; and T_0 is the sampling time.

iii) Mathematical Model

A mathematical model that is generally accepted to represent the behavior of the level of fluid in the head tank of Figure 5.5 is

$$K \cdot A \frac{dL}{dt} + R(v)\sqrt{L} = F_{in} \qquad \text{[lpm]} \tag{5.6}$$

where L = level of fluid in the tank, [m]

F_{in} = flowrate of fluid entering the tank, [lpm]

A = cross sectional area of the tank, [m^2]

$R(v)$ = resistance to flowrate of fluid leaving the tank, [lpm / $\sqrt{m}$]. This resistance depends also on factors such as valve type, valve size, pipe size, and downstream pressure.

v = valve opening, [%]

K = 1000, a constant to adjust the dimensions of equation (5.6), [l/m^3]

$R(v)$ is assumed proportional to the opening of the valve, although other correlation may be used if required (e.g., if the valve is non linear). $R(v)$ is given by

$$R(v) = K_R \bullet v \tag{5.7}$$

where K_R is a parameter that comprises all other factors that determine the resistance to the flowrate of fluid leaving the tank. Replacing $R(v)$ in equation (5.6), and incorporating $C = K \bullet A$, the capacity of the tank, we obtain that the model of the tank of Figure 5.5 is given by

$$C\frac{dL}{dt} + K_R\, v\, \sqrt{L} = F_{in} \qquad \text{[lpm]} \tag{5.8}$$

The modified Euler's method for numerical integration (Stark 1970), is used to implement and simulate equation (5.8). Applying the modified Euler's method to equation (5.8) we obtain the following recursive expression:

$$L_k = L_{k-1} + \tfrac{1}{2}[f(L_{k-1}, F_{in(k-1)}, t_{k-1}) + f(L_k^*, F_{in(k)}, t_k)] \cdot T_\Delta \tag{5.9}$$

where k, k-1 = indices of current and previous sampling instants

T_Δ = integration step

L_k^* = a preliminary estimate of L_k given by

$$L_k^* = L_{k-1} + f(L_{k-1}, F_{in(k-1)}, t_{k-1}) \cdot T_\Delta$$

and f is a re-written version of equation (5.8) given by

$$f(L, F_{in}, t) = \frac{dL}{dt} = -\frac{K_R}{C} v \sqrt{L} + \frac{1}{C} F_{in}$$

Equation (5.9) was implemented in ProcessVision; its simulation results are assumed to represent the true behavior of the level of the fluid in the tank.

iv) Simulation Results

The following values for the parameter of the tank (chosen arbitrarily) were used during this simulation study:

$A = 1.0\ m^2$ $\quad K_R = 0.943\ (lpm / \sqrt{m}\ /\%)$ $\quad v = 75\ \%$ (nominal valve opening)

Tuning of the steady state and dynamic QMs (equations (5.4) and (5.5)), requires the value of two parameters: K_R and τ. Assuming that we do not know the numerical value of the parameters of the tank, we would need to simulate equation (5.9) to obtain K_R and τ (in a real-world case, we would need to perform some experimental tests around the tank). From the simulation results using the above parameters, we obtain that $K_R = 0.943\ (lpm / \sqrt{m}\ /\%)$ and $\tau = 32$ min.

Different simulation tests were performed to compare the results provided by the QMs proposed to those obtained from the mathematical model of the tank. Although this comparison involved only numerical values, the results predicted by QMs were also available in qualitative terms at any given time during the simulation.

Figure 5.7 shows the simulation results for a valve opening of $v = 75\ \%$. During this test ***frate_in*** takes the values 100 lpm, 50 lpm, and 75 lpm.

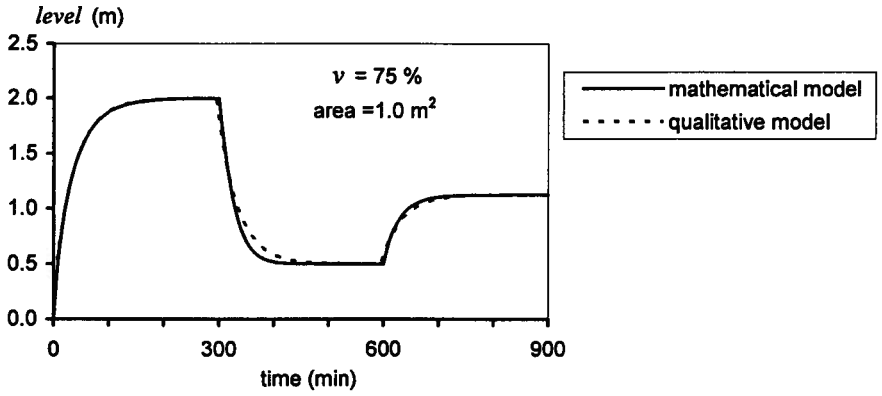


Figure 5.7 Actual and predicted level of fluid in the tank; ***frate_in*** takes the values 100, 50, and 75 lpm.

In order to study the effect of different valve openings on the ability of the QMs to predict the behavior of the tank level, tests were performed for valve openings of $v = 50$ % and $v = 100$ %. The rest of the parameters of the tank for this test are the same as those used in the previous test. The results are shown in Figure 5.8.

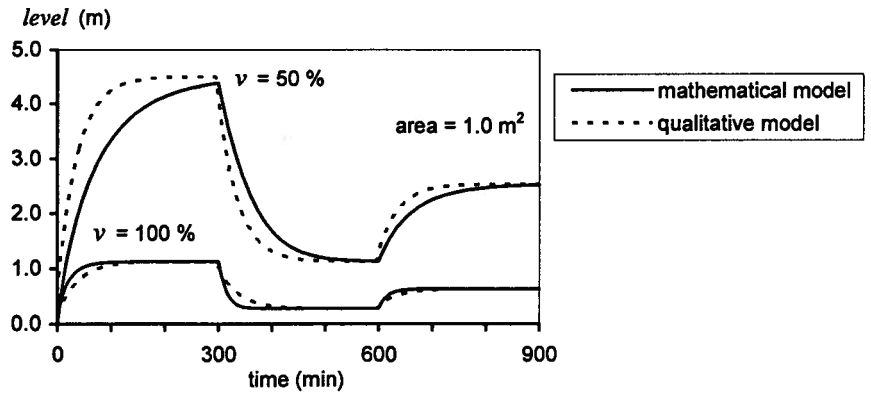


Figure 5.8 Simulation results for two valve openings: $v = 50$ % and $v = 100$ %; ***frate_in*** takes the values 100, 50, and 75 lpm.

A simulation test to study the ability of QMs to predict the behavior of similar processes without re-tuning parameters was also performed. This test simulated the case where a QM was applied to

tanks with different cross sectional area. Figure 5.9 shows the results obtained for areas of 0.5 m^2 and 2.0 m^2 and a valve opening of 75 %; the results for the original tank with an area of 1.0 m^2 are shown in Figure 5.7.

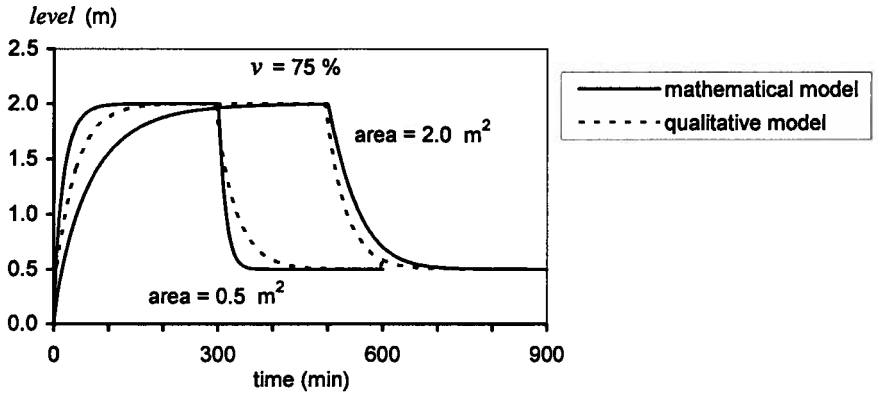


Figure 5.9 Simulation results on two similar tanks but with different areas; ***frate_in*** takes the values 100 and 50 lpm.

v) Analysis of Simulation Results

From the simulation results presented above, we can observe that:

- A well-tuned QM can accurately predict the behavior of a process (see Figure 5.7). This accuracy depends also on other factors such as the quality of the knowledge represented by the QM. We cannot expect to have the same results with a QM built from simple heuristic knowledge than with one derived from a mathematical model of the process.

- The performance of a QM deteriorates when process parameters change (see Figure 5.8). This deterioration could be corrected by tuning the QM as parameters change (as the valve opening changes, in this case), or by establishing a correlation between these parameters and the variables in the model. There is no general solution to this problem; each case must be treated separately.

The prediction errors shown in Figure 5.8 can be corrected however, by incorporating the valve opening into the original QM of the tank. Only the dynamic QM (equation (5.5)) needs to be modified; the steady state QM (equation (5.4)), already considers the valve opening. From equation (5.8) (assuming that we know the mathematical model of the tank), we obtain that the corrected time constant of the tank, as a function of the valve opening, is given by the following QM

$$\tau \propto \tau_0 \left(\frac{v_0}{v}\right)^2 \tag{5.10}$$

where τ_0 = 32 min. and v_0 = 75 %, are the original values for the time constant and the opening of the valve, respectively. Equation (5.10) is incorporated into equation (5.5) to obtain a corrected QM of the tank shown in Figure 5.5.

The simulation test of Figure 5.8 was repeated using the corrected QM, providing the results shown in Figure 5.10.

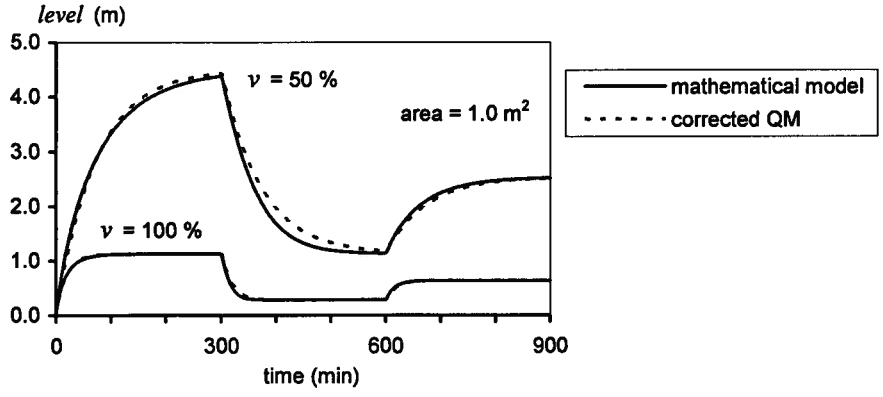


Figure 5.10 Simulation results obtained from a corrected QM of the tank.
frate_in takes the values 100, 50, and 75 lpm.

From the results shown in Figure 5.10, we can see that the prediction errors have been eliminated. These results demonstrate that the complexity of a QM can be adjusted in order to expand the range of validity of the model, i.e. the set of operating condition in which the model accurately predict the behavior of a process.

- The simple first order dynamic QM primitive proposed can accurately predict the dynamic behavior of a process (see Figure 5.7). The accuracy, however, is sensitive to the tuning of the QM (see Figure 5.8), and it may also be affected by the actual order of the dynamic behavior of the process being modeled.

- A QM that has been accurately tuned for one specific process may behave poorly if applied to another process (compare the results of Figure 5.7, for a tank with a cross sectional area of 1.0 m^2, to those shown in Figure 5.9 for tanks with different areas). QMs should be appropriately tuned for each application to achieve good results.

  Following a similar procedure to that described above for the case of the valve opening, the original QM could be corrected to take the area of the tank into consideration. The corrected QM eliminates the prediction errors shown in Figure 5.9.

5.5 Potential and Limitations of the Approach Proposed

i) Handling Heuristic Knowledge

One of the most important features of qualitative modeling is that a QM can be built with a minimum of information. We do not need to know in detail the kind of phenomena that defines the behavior of a process to start building a QM, nor do we require a previous mathematical model of the process – heuristic knowledge may suffice. The accuracy of the prediction, however, will reflect the accuracy of the information employed to build the model.

The pseudo-qualitative approach proposed offers similar flexibility to that of the pure qualitative approach as proposed by Forbus (Forbus, 1984 & 1985). Both approaches can handle QMs based on heuristic knowledge such as qualitative relationships of the form "the output is proportional to the input of the process". However, the pseudo-qualitative approach proposed can also handle QMs based on very accurate information. Under this approach, for example, a QM could be built as a direct representation of a mathematical model, as in the case of the open tank described in the previous section.

ii) Resolution and Ambiguities

An important feature of QMs is their ability to handle information at a low level of resolution. A QM can handle variables whose values are expressed in qualitative terms such as "small," "medium," and "large." Both inputs and outputs of a QM would be expressed in similar terms. If the above resolution satisfies the requirements of an application, any additional information about the values of the variables involved would be irrelevant for that application.

Working at a low level of resolution, however, limits the capacity of QMs based on purely qualitative approaches to represent a more complex behavior such as one described by a non-linear relationship. A three-level resolution QM cannot represent properly a quadratic relationship between two variables, V_1 and V_2, as shown by the shaded areas in Figure 5.11. In this case, a generalized definition of fuzzy concepts was used, i.e. the definition of fuzzy concepts is based on percentage values and is the same for each variable. This three-level resolution QM cannot reliably differentiate between a linear and a non-linear relationship.

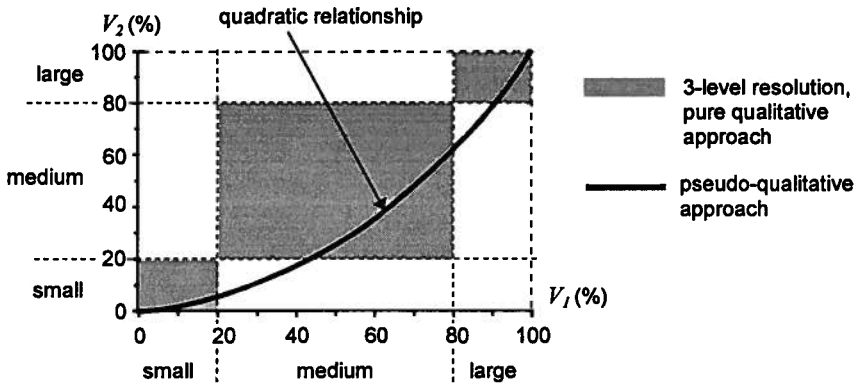
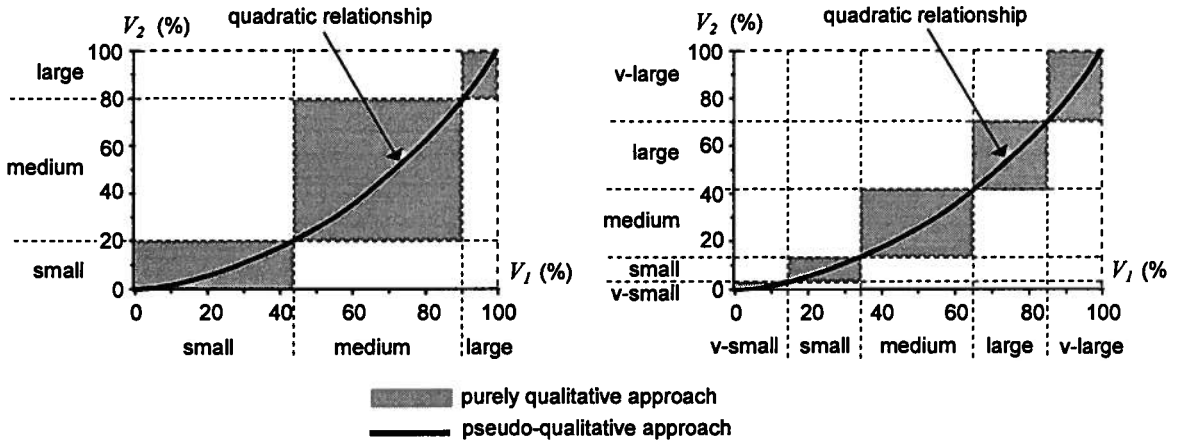


Figure 5.11 Representation of a quadratic relationship under the pseudo-qualitative approach proposed (continuous line) and under a purely qualitative one (shaded areas), using a generic definition of fuzzy concepts.

The pseudo-qualitative approach proposed does not exhibit the representational problems found in purely qualitative approaches. It offers a high resolution for the internal representation of knowledge and, at the same time, an external level of resolution that responds to the specific requirements of an application. A quadratic behavior between two variables, for example, would be internally represented by a mathematical relationship as shown by the continuous line in Figure 5.11. The input/output information, however, could still be expressed in terms of the low resolution values: "small," "medium," and "large."

The ability of a pure QM to represent nonlinear behavior can be improved by redefining the fuzzy concepts to fit the nonlinearity. As shown in Figure 5.12(a), the improvement obtained is not substantial. Increasing the resolution is another alternative to improve the representational ability of a pure QM. A five-level resolution gives the results shown in Figure 5.12(b). From these results we can see that some improvement is obtained by increasing the resolution. However, changing the resolution implies changing the fuzzy concept definition as well as the heuristic rules associated with the variables involved in the model. Also, as the resolution increases so does the complexity of the mechanisms that handle this information. As shown in Figure 5.12, the results from the improved

versions of a pure QM are inferior to those from a pseudo-qualitative approach. The latter correspond exactly to the mathematical representation of a quadratic relationship.



(a) 3-level resolution QM (b) 5-level resolution QM

Figure 5.12 Representation of a quadratic relationship using a pure QM at different levels of resolution and the pseudo-qualitative approach proposed.

Low level resolution representation also brings ambiguity problems to purely qualitative formalisms. A QM based on these formalisms, for example, cannot differentiate between two "positive medium" variables. The pseudo-qualitative approach proposed is not affected by these ambiguity problems; a QM under this approach uses a degree of belief associated with the qualitative description of a variable to uniquely identify the value of each variable.

iii) Qualitative Models and IF-THEN Rules

Qualitative modeling is a powerful mechanism to represent knowledge about process behavior at low levels of resolution. It should not, however, be considered as a direct rival to any other technique that may be employed in a given application. Each technique has its own strengths, weaknesses, and specific areas of application in which they are more suitable than other available techniques.

Qualitative models may not be the recommended alternative in an application that only requires the representation of one simple linear steady state proportional relationship; IF-THEN rules may suffice. However, if the application involves the representation of a more complex behavior such as the dynamic behavior of the tank shown in Figure 5.5, qualitative models would certainly be an appropriate option.

iv) Uncertainty in Qualitative Modeling

Purely qualitative formalisms do not handle uncertain information in the characterization of variables: variables have "crisp" values, e.g. a variable is either definitely "small" or definitely "medium". This becomes a drawback in an application in which qualitative modeling is combined with other representation techniques that work with uncertain information.

The pseudo-qualitative approach proposed in this research, unlike purely qualitative approaches, handles uncertainty. Uncertainty, under this pseudo-qualitative approach, is represented by a degree of belief associated with each piece of knowledge handled by the QM. This is another important feature of the approach proposed.

5.6 Incorporation of Qualitative Modeling into an ISCS

The validity of the pseudo-qualitative modeling approach proposed has been demonstrated. The flexible structure of a QM allows the representation of behavior derived from simple heuristic knowledge, as well as a more accurate behavior obtained from existing mathematical model.

Simulation results showed that a QM, built using the pseudo-qualitative modeling approach, can accurately predict the behavior of a physical process. The structure of a QM and the mechanisms for

information representation allow on-line simulation and handling of uncertain information; these are two important features that allow the incorporation of QMs into an ISCS.

The next step in this research is the incorporation of qualitative modeling into a real-time ISCS. This involves the development of an ISCS prototype and its application to a real-world situation. This next stage will further establish the validity of the approach proposed and the feasibility of its application to an industrial plant.

CHAPTER 6

DESIGN AND IMPLEMENTATION OF AN ISCS PROTOTYPE

6.1 Objective and Scope

The main objective for developing this intelligent supervisory control system (ISCS) prototype is to demonstrate the feasibility of applying a qualitative modeling approach within a system designed for a full-scale industrial operation.

This prototype was applied to the C Line grinding circuit at Highland Valley Copper (HVC) in Logan Lake, B.C., Canada, to assist in process supervision. The initial focus has been to monitor and detect tonnage restrictions that affect circuit production.

The system is designed to provide on-line advice via an intelligent SCADA interface to the shift supervisors in the control room regarding the occurrence of circuit restrictions. At this stage the system does not intervene directly with the operation of the process. Appropriate actions are left to the discretion of plant operators.

Periodic reports summarize diagnostic results; these are used to assess the extent and frequency of production-limiting bottlenecks. Changes in operating philosophy are expected to derive from this analysis. On-line information, together with reports summaries, can reduce the decision-making time. This can lead to removal of a restriction fast enough to reduce tonnage losses. Average losses due to grinding circuit restrictions range up to 5000 tons per week for the C-line circuit.

6.2 Background

Once per week, an experienced metallurgist analyses the weekly output of trend graphs together with shift log reports to establish the extent, frequency and occurrence of production losses due to circuit restrictions. The approach used is highly heuristic, perhaps even subjective, leading to inconsistent analyses and reliance on a single individual to perform the work.

It was considered that development of an on-line system capable of providing minute-to-minute analysis using the knowledge of the expert could result in several gains:

- more consistent and reliable analysis.
- faster turnaround of data.
- reduction in the extent and frequency of certain delays.
- potential improvement in the automatic tonnage control system.

Accordingly, this system had been developed based on the expert knowledge of Hans Raabe, who at the time of initiation of the project, February 1995, was the plant metallurgist at HVC.

6.3 General Structure

The ISCS runs on a PC (UBC-PC) and is interfaced to the HVC-PC computer that runs the Bailey DCS (Distributed Control System) currently in operation at HVC (see Figure 6.1). The ISCS accesses the Bailey database to obtain on-line information about the operation of C-line. The data are used by the ISCS reasoning system to detect and monitor restrictions that affect the operation of the process.

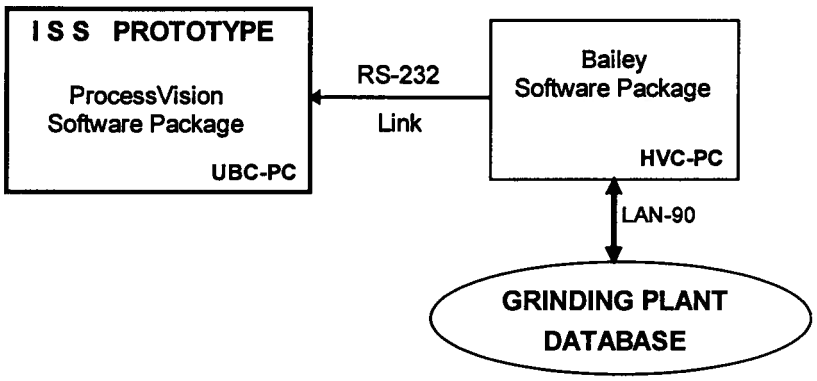


Figure 6.1 General structure of the implementation of the ISCS prototype.

The ISCS has been implemented using the software package ProcessVision from Comdale Technologies, Toronto, Canada.

A file with required process data is created on the HVC-PC and transferred from this computer to the UBC-PC via an RS-232 serial link. Current data updates take place on about a 60 second time interval. A communication routine on the UBC-PC updates the ProcessVision database with the information received. Direct access to the Bailey network will eventually be setup once the HVC-PC has upgraded its operating system.

6.4 ProcessVision Software Package

ProcessVision is a modular software package for development of knowledge-based real-time applications. ProcessVision runs on a standard PC under the QNX 4.2 operating system from Quantum Software of Kanata, Ontario. Version 5.3, used in this application, is a multi-tasking development toolkit in which all sub-tasks are assigned to separate modules; data transfer, message transfer, alarming, scheduling events, trend analysis, and knowledge processing. The software provides facilities to interface with existing control systems, field I/O and PLCs.

Information in ProcessVision is represented by keyword triplets of the form:

object.attribute.value :DoB

or *{class}.attribute.value* :DoB

where DoB is the degree of belief (assigned by the user or estimated by the system) in the knowledge represented by the keyword triplet.

In this format, a developer can conceptualize knowledge in an object-oriented manner. This approach provides a structured manipulation and organization of objects by type or "classes" of objects. A piece of equipment or a process variable can be considered as an object represented by the object token of a keyword triplet. Specific characteristics of an object, such as the area or height of a tank, are represented by the attribute token; quantification of these attributes is executed by the value token of the keyword triplet. An example of a keyword triplet is:

tank.area.small DoB: 85

This triplet indicates that "the area of the tank is small", and that the degree of belief associated with this piece of information is 85%.

Comdale/C, a real-time expert system, is the module that enables ProcessVision to handle symbolic information. This module deals with heuristics about the behavior of a process as well as information captured from the expertise of human operators. Comdale/C can reason with this knowledge to make decisions on the best actions to be taken or to advise the operator on the qualitative state of the process. Other modules that comprise ProcessVision include a historical database, a network administrator, an alarm administrator, an explanation facility, and a graphical user interface. The modularity of an application together with the features of each individual module

allow a system to be designed to mimic the heterogeneous functions used by a human to reason about a complex process.

6.5 Tonnage Restrictions in the C-Line Grinding Circuit at HVC

i) Process Description

Feed for C-line is reclaimed from a stockpile by five variable-speed feeders (see Figure 6.2). Primary grinding is achieved by a Semi-Autogenous Grinding (SAG) mill (43'x16') equipped with grate discharge. The mill feeds two grizzlies where the undersize is split out. Oversize returns to the SAG mill, while the undersize is sent to two ball mill discharge pumps. Mill power derives from two variable-speed 4700 KW motors. Feed tonnage ranges from 1200 to 2000 tph.

Secondary grinding occurs in two 16.5'x27' ball mills operated in closed circuit with a cluster of ten 760 mm cyclones. Cyclone overflow discharges by gravity to the flotation plant where copper minerals are extracted.

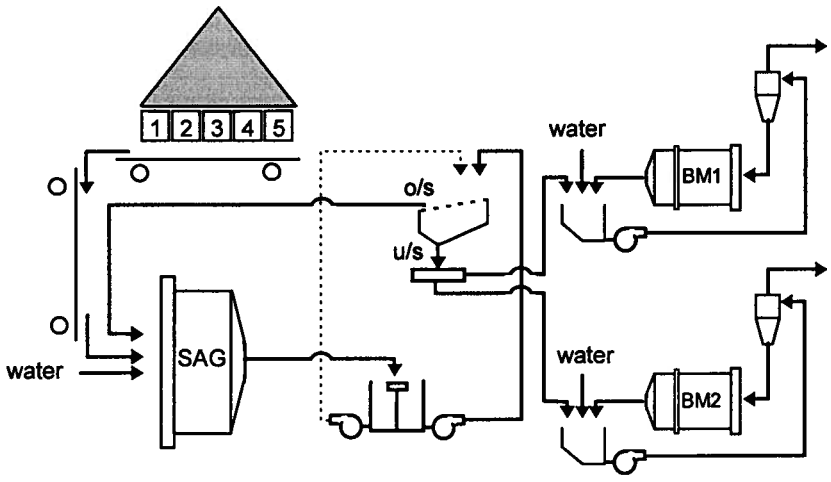


Figure 6.2 C Line Grinding Circuit at HVC.

A Bailey Distributed Control System interfaces with the grinding plant. The global objective of the control strategy is to maximize tonnage at maximum power draw. Fresh feed is adjusted by one of the two main control loops: SAG mill power draw or SAG bearing pressure. A switching logic program determines which loop manipulate the fresh feed at any given time. There is also a second algorithm to adjust the speed of the SAG mill in order to maintain a constant mill inventory of ore. All other major process variables (water addition, cyclone conditions, etc.) are also monitored and adjusted by the control system.

ii) Tonnage Restrictions

A tonnage restriction exists when the tonnage fed to the mill cannot be maintained (it must be decreased) or cannot be increased beyond a specific value due to the presence of certain operating conditions. Tonnage restrictions prevent the control system from achieving its goal of maximum tonnage; thus, a tonnage loss is attributed to their occurrence.

The most frequent type of restriction occurs when the tonnage cannot be maintained at a specific value. Two cases can be distinguished:

- The operator imposes a upper limit on the fresh feed setpoint. Although the SAG mill could process higher tonnage, the operator restricts its value due to specific operating conditions, e.g. when one ball mill is down for maintenance or when the plant tailings discharge box level is high.

- The fresh feed controller cannot achieve the required setpoint value. In this case, the fresh feed flowrate drops below the setpoint value due to operating problems in the feeding system, e.g. some feeders are down or some chutes are plugged.

The operator imposes a high limit on the fresh feed setpoint by reducing the maximum tonnage value in the hi-limit block (see Figure 6.3). Thus, although the main controller (power or bearing pressure) requires a higher fresh feed flowrate, the setpoint to the fresh feed controller is limited to the high limit value imposed by the operator. The SAG mill speed is also modified when the tonnage is being limited.

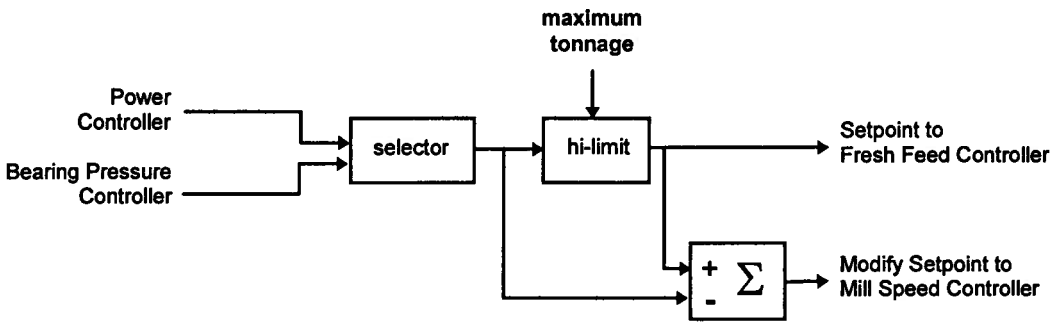


Figure 6.3 Setting a high limit for the fresh feed setpoint.

The second type of restriction occurs when the tonnage has reached the absolute maximum that can be delivered to C-line. Tonnage is limited at this level even though the SAG mill could still handle more ore. This is generally known as a "soft ore" restriction. Raabe (1994) indicates that losses due to this restriction are the most difficult ones to estimate.

A different tonnage restriction, much more difficult to detect, may occur when a feeder with fine ore goes down but the rest of the feeders can continue supplying the required tonnage (Raabe 1994). Feeders draw from different points in the stockpile into which ore is discharged from the primary crusher. Particle segregation takes place naturally in this stockpile meaning that certain feeders tend to deliver coarser ore than others (in the case of C-line, feeders #1 and 5 supply coarser ore than feeders # 2, 3 and 4). Raabe argues that the mill throughput may be greatly affected if the proportion of coarse ore in the feed increases when a feeder goes down. This restriction may eventually disappear as the stock pile material naturally readjusts to a new equilibrium.

H. Raabe (1994) developed a heuristic procedure to evaluate the occurrence of restrictions, to determine possible causes, and to estimate the tonnage lost in each case. The procedure is performed manually on a weekly basis requiring intensive examination of trend graphs as well as information from the control room.

Much of the procedure is based on subjective judgment; hence, different people will obtain different results. As well, the knowledge is only available well after the restrictions have occurred. It was considered that an on-line system to perform this analysis automatically could help to standardize the procedure and decrease the frequency and duration of such restrictions. The application thus, attempts to mimic how experienced plant metallurgists think about and interpret circuit upsets rather than modeling data extracted from the plant directly.

6.6 Design Aspects of the ISCS Prototype

The following is a brief description of major aspects of the design of the ISCS prototype. The main aspects of the knowledge base used in the system can be viewed in Appendix B.

i) Knowledge Representation

The ISCS uses keyword triplets as the basic element for knowledge representation. Information such as status of equipment, which requires a True/False value, is represented by logical keyword triplets. Other information such as values of process variables are represented by both numerical and qualitative keyword triplets (see chapter 5 for more details). The following are some examples of the knowledge represented by keyword triplets on this application.

mill-power.real-value.@f		; numerical value of measured variables
sag-mill.status.running	(True/False)	; equipment status
mill-power.real-value.high	(DoB)	; qualitative value of variables

ii) Diagnosing Tonnage Restrictions

A complete classification of all the major tonnage restrictions that affect the operation of the C-line grinding circuit, and possible causes of these restrictions were obtained during discussions with HVC personnel. This information along with the heuristic procedures involved in diagnosing tonnage restrictions were incorporated into the ISCS. This heuristic knowledge was implemented using IF-THEN rules and "procedures" (a structured knowledge representation element offered by ProcessVision).

The ISCS updates its knowledge base with information read from the process database at regular time intervals. The reasoning system uses this information together with the heuristic rules to detect the occurrence of tonnage restrictions, determine potential causes and estimate tonnage losses. Figure 6.4 shows a diagram with the sequence of major tasks performed by the ISCS at each sampling time.

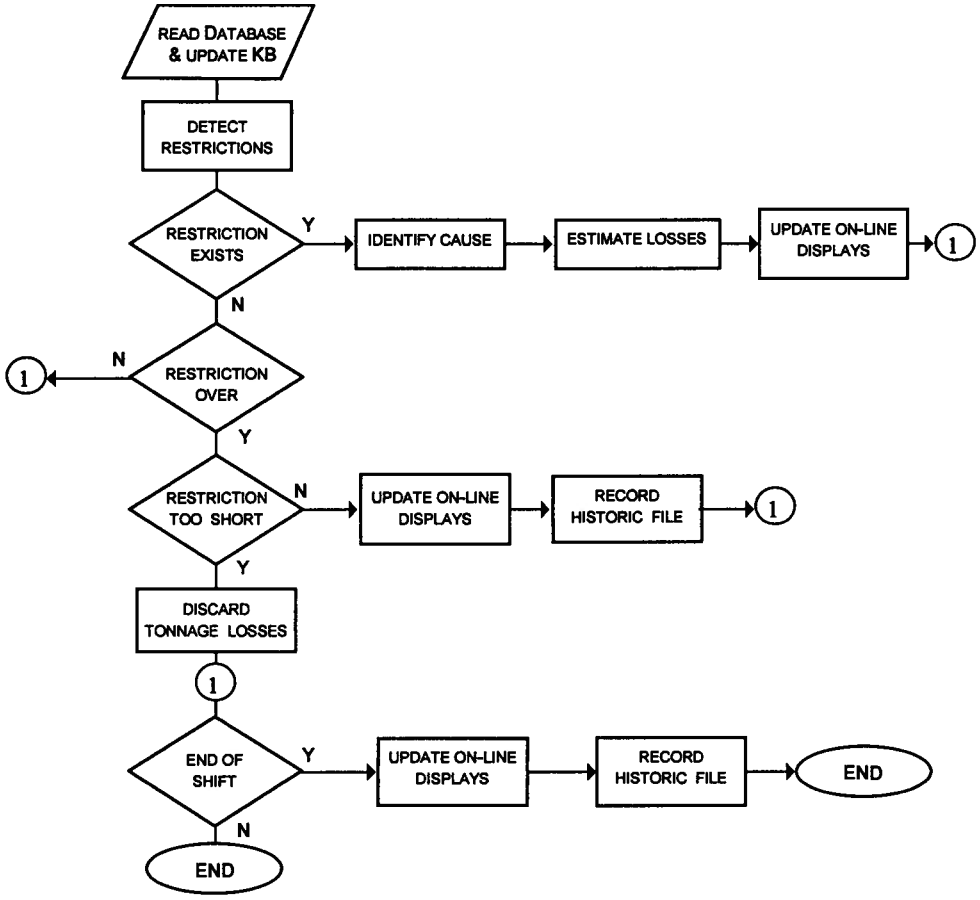


Figure 6.4 Sequence of major tasks performed by the ISCS.

The following is a list of the restrictions that the ISCS prototype can detect. A brief description of the heuristics associated with the diagnosis of each tonnage restrictions is also given.

RR_FEED_LIMIT: The operator imposes a high limit to the fresh feed flowrate to the C-line grinding circuit.

Detection: The system will indicate that this restriction occurs when the following conditions are met: the input to the high limit block that limits the tonnage is higher than its output value, and this output value is lower than the maximum tonnage (target value) specified for the C-line grinding circuit, (see Figure 6.3).

RR_FEED_CUT: Fresh feed is cut completely, but the SAG mill continues running.

Detection: The system will indicate that this restriction occurs when all feeders stop, while the SAG mill continues running.

RR_SOFT_ORE: Tonnage increases until it reaches the maximum value (target value) specified for the C grinding line.

Detection: The system will indicate that this restriction occurs when the following conditions are met: the tonnage reaches the high limit (the absolute maximum value that can be delivered to C-line), the bearing pressure remains relatively stable, and the power draw drops below the setpoint value.

RR_FEEDER_DOWN: One or more feeders with fine ore are down (feeders #2, 3 or 4), but the rest of the feeders can supply the required tonnage.

Detection: The system will indicate that this restriction occurs when the following conditions are met: a feeder with fine ore is down, the mill power draw is at its setpoint value, but the tonnage drops from the average value it had before the occurrence of the restriction. (The mill throughput decreases due to a coarser fresh feed).

RR_ORE_SUPPLY: The fresh feed flowrate cannot reach its setpoint value.

Detection: The system will indicate that this restriction occurs when the fresh feed flowrate drops below its setpoint value and does not appear to return to it.

RR_SAG_STOPS: SAG mill stops. (Mill stops are not considered as restrictions; they will be detected and recorded but no tonnage lost will be estimated)

Detection: The system will indicate that this restriction occurs when the SAG mill speed is zero or the SAG mill motor control signal is OFF.

After detecting a restriction, the ISCS prototype uses its heuristic knowledge to determine the cause of the restriction detected. Table 6.1 shows some of the possible causes for each of the restrictions the prototype handles. The ISCS provides the mechanisms which allow the user to confirm/correct on-line some of the findings and results reported by the system.

Table 6.1 Restrictions handled by the ISCS prototype and their possible causes.

CAUSE / RESTRICTION	SOFT ORE	PLUGGED CHUTES	SAG CONTROLS	BALL MILLS	BM PUMPS	FLOTATION	TAILINGS	OTHER
RR_SOFT_ORE	YES[1]							
RR_FEED_CUT			YES	YES	YES			YES [2]
RR_FEED_LIMIT			YES	YES	YES	YES	YES	YES [2]
RR_FEEDER_DOWN								YES [3]
RR_ORE_SUPPLY		YES						YES [4]
RR_SAG_STOPS				YES [5]				YES [6]

[1] YES, in each case, means that the corresponding cause could be responsible for the restriction detected.

[2] The restriction could be due to other problems such as BM cyclones.

[3] One or more feeders with fine ore are down (feeders #2, 3 or 4). Undetermined feeder problems.

[4] Low ore supply or feeder problems could be the cause of this restriction.

[5] Both ball mills are down.

[6] Problems with the SAG pump or other unclassified problems may have caused this restriction.

iii) Estimating Tonnage Lost

The regulatory control system automatically compensates for losses resulting from brief restrictions: an initial drop in tonnage is compensated by a sharp increase in tonnage when the restriction is

removed. Hence, a tonnage loss is only recorded when the restriction affects operation for a period of time longer than 10 min.

Estimation of tonnage loss is based on the difference between a predicted and a measured value of the fresh feed to the SAG mill. The predicted value is the tonnage rate assumed to be processed by the mill if the restriction had not occurred. For some restrictions, the predicted value is obtained from an analysis of tonnage trends before the occurrence of the restriction. For other restrictions, the predicted value derives from appropriate qualitative models of the process.

Every restriction, except for the "soft ore" one, is characterized by a tonnage reduction, i.e. the tonnage during the restriction is lower than the average value before the occurrence of the restriction. In this case, the predicted tonnage value during a restriction corresponds to the average tonnage value before the occurrence of the restriction (for a period without restrictions). Thus, the tonnage lost is estimated as the accumulated difference (area) between the predicted value and the actual tonnage measured during the restriction (see restriction 1 in Figure 6.5).

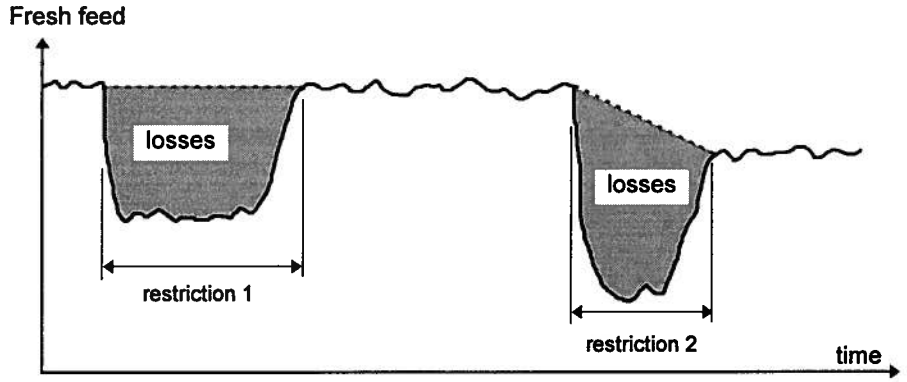


Figure 6.5 Estimating tonnage losses from historic trends (schematic trends).

When the restriction is over, the tonnage may stabilize at a value different than the average value before the occurrence of the restriction. If this occurs, the ISCS assumes that the predicted tonnage

value is an inclined line connecting average tonnage values before and after the restriction (see restriction 2 in Figure 6.5).

Tonnage losses due to a "soft ore" restriction are much more difficult to estimate. There is no simple answer as to what tonnage value might be possible if a high limit restriction did not exist (see Figure 6.6).

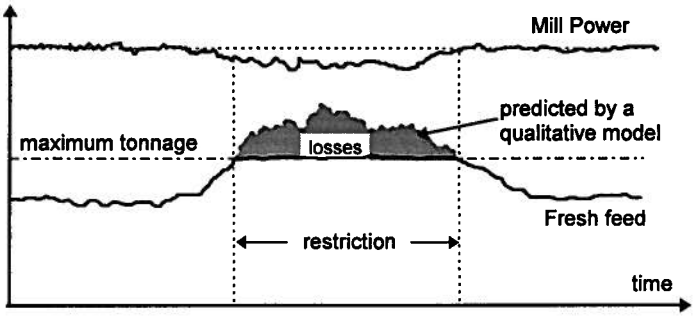


Figure 6.6 Estimating tonnage lost due to a "soft ore" restriction (schematic trends).

As shown in Figure 6.6, during a "soft ore" restriction the power draw drops as the fresh feed reaches the absolute maximum value that can be delivered to C-line. Thus, tonnage losses are estimated as the additional tonnage required to maintain the power draw at its setpoint value (shaded area in Figure 6.6). Metallurgists use historic trends to determine the drop in the power draw and the duration of the restriction. From this information, they would estimate the approximate tonnage losses associated with the restriction. Raabe (1994) notes, however, that these heuristics procedures to estimate losses are based on pure "guesstimating" techniques; hence, different people obtain different results.

Conventional rule-based systems lack appropriate tools to implement heuristic procedures such as one to estimate losses during a "soft ore" restriction. A promising alternative to overcome this lack of representation capabilities is offered by the qualitative modeling approach proposed in this

research. Under this approach, a QM that represents the relationship between the power draw and the fresh feed can be implemented. The QM can be built directly from the heuristic procedure currently employed at HVC or from information provided by a mathematical model of the process. This QM estimates a tonnage value that accounts for the difference between the actual mill power draw and its setpoint value. Thus, the difference between the tonnage predicted by the QM and the actual tonnage measured will correspond to the tonnage lost due to the "soft ore" restriction.

A QM that mimics the heuristic procedure employed by metallurgists at HVC to estimate tonnage losses has been incorporated into the ISCS. This QM can be expressed as

$$Tonnage\_loss \quad \alpha \quad (Power\_draw\_setpoint - Power\_draw)$$

The proportionality constant used by the ISCS for this model corresponds to the ratio between the maximum tonnage rate that can be delivered to C-line and the power draw setpoint.

The ISCS accumulates the estimated losses over the period the restriction affects the operation. The ISCS displays these losses in real-time as it accumulates them while the restriction is present; it also reports and records the final estimate when the restriction is over. Appendix B presents the overall steps required to incorporate additional QMs into the ISCS should they be required in the future. Appendix D the source code of the QM to estimate tonnage losses as incorporated into the ISCS.

iv) User Interface

The ISCS provides on-line information associated with the supervision of the process and findings from its restriction diagnosis. This output includes type of restriction detected, classification of restriction cause, estimate of tonnage lost, and accumulated losses per shift. The ISCS also provides mechanisms to allow the user to confirm or correct on-line some of the findings and results reported

by the ISCS. Should the system begin to diagnose a fault that has not really occurred, the user can override the system and prevent an erroneous prediction.

Figure 6.7 and Figure 6.8 show two of the displays the ISCS offers to interact with the user. Figure 6.7 is a display with an overall view of the process. This display shows the value of relevant variables and the status of the equipment involved; this displays also has a message window to inform about the occurrence of restrictions. This displays provides the mechanism for the user to override ISCS findings. Figure 6.8 shows a displays where the ISCS reports tonnage loss per category. Losses are reported for current shift, previous shift and accumulated-to-date. When a restriction occurs, the ISCS highlights the corresponding row, and reports the duration and estimated losses in real-time as the restriction progresses. The accumulated-to-date information is also presented as percentage of the total losses in the circuit.

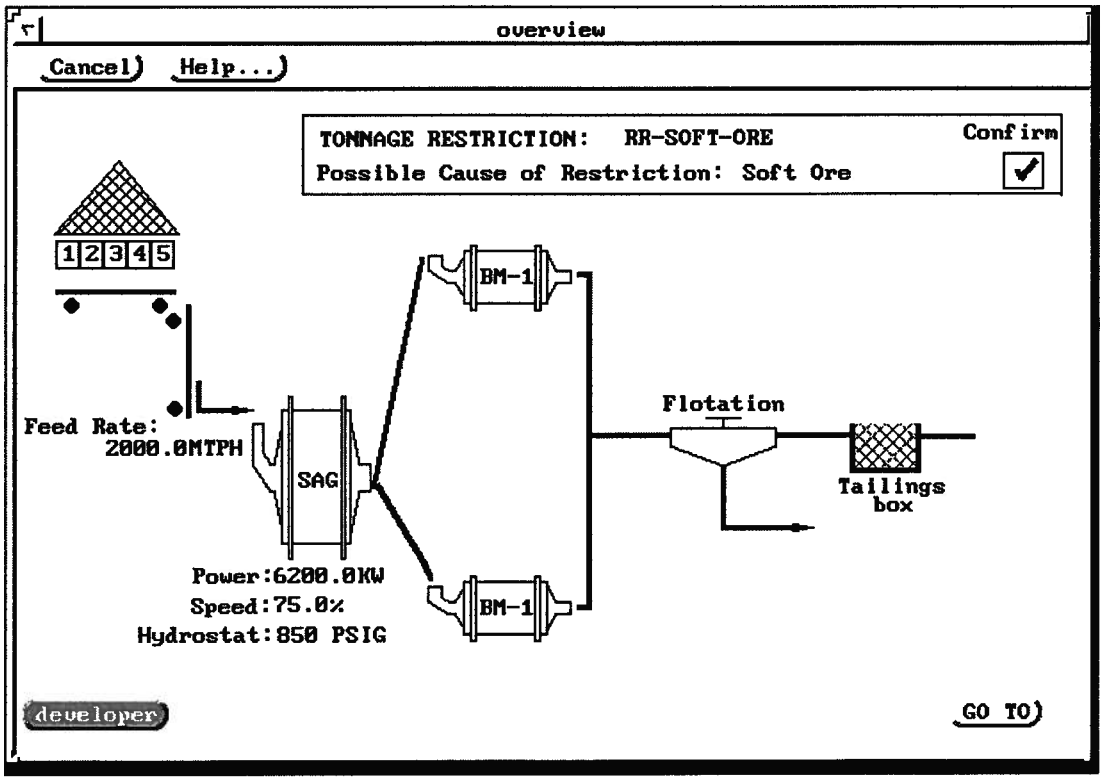


Figure 6.7 User Interface: Process Overview Display.

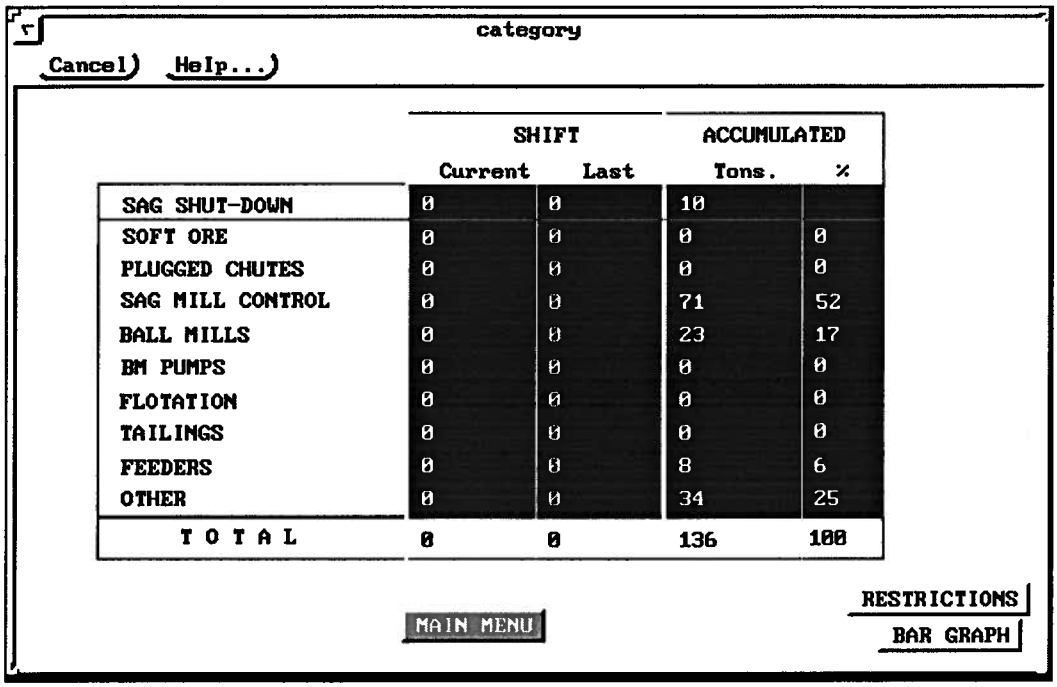


Figure 6.8 User Interface: Tonnage loss per Category.

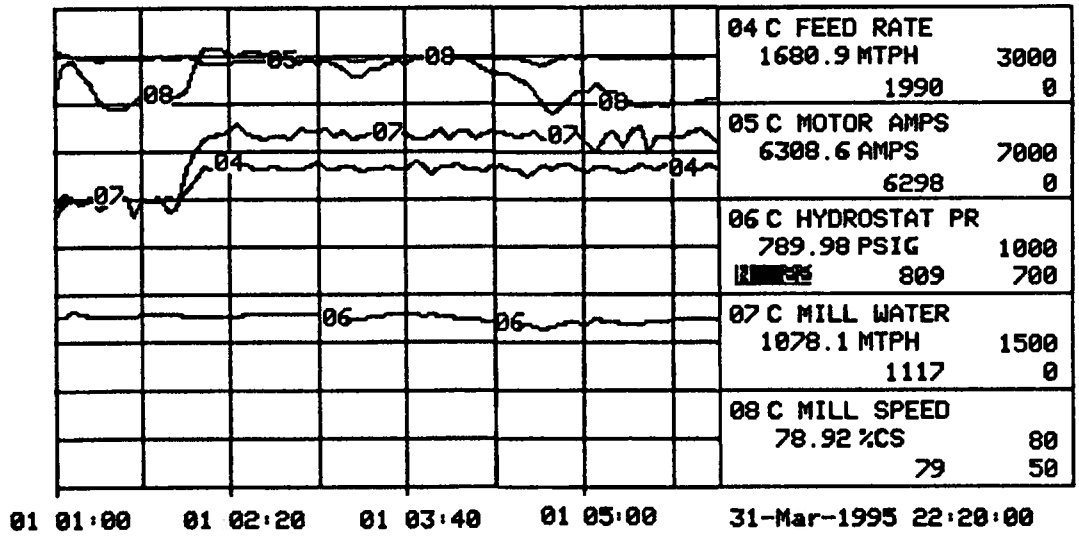
The ISCS generates reports at the end of each shift with a summary of supervision results. A file with a record of all relevant events is also generated by the ISCS. This file can be accessed by the plant metallurgist to assist in daily or weekly analysis.

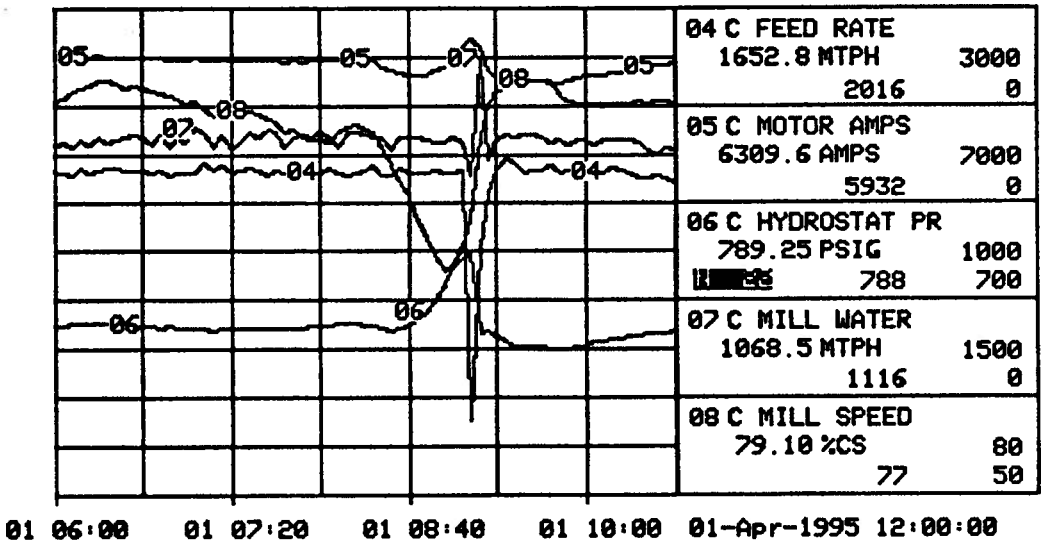
6.7 Experimental Results

The ISCS ran at HVC for an evaluation period of six weeks. The diagnostic results obtained from the ISCS were stored on files to be analyzed at the end of the evaluation period. Operators did not have access to the system nor to the results during this period.

Trends of relevant process variables were obtained to analyze the capabilities of the ISCS system to diagnose tonnage restrictions. The following are some of the results provided by the ISCS, together with trends of process variables and a list of major events for the periods of interest.

i) Soft ore restriction (RR_SOFT_ORE)





8:22 Feeder #5 stops
9:07 All feeders stop (the power draw reaches its high-high limit)
9:25 Feeder #5 starts

Figure 6.9 Soft ore restriction: April 1, 1995, day shift.

The ISCS reported the following restrictions during this period of time.

TIME	RESTRICTION	DURATION	TONNAGE LOST
02:06 — 09:07 hrs.	RR_SOFT_ORE	7 hrs. 1 min.	103 tons
09:07 — 09:15 hrs.	RR_ORE_SUPPLY	8 min.	128 tons (too short)
09:15 — 10:19 hrs.	RR_SOFT_ORE	many very short restrictions	-------

The soft ore restriction that started at 02:06 hrs. was adequately diagnosed by the ISCS. From an analysis of the historic trends, a metallurgist would have estimated 80 tons lost for this restriction, slightly lower than that estimated by the ISCS.

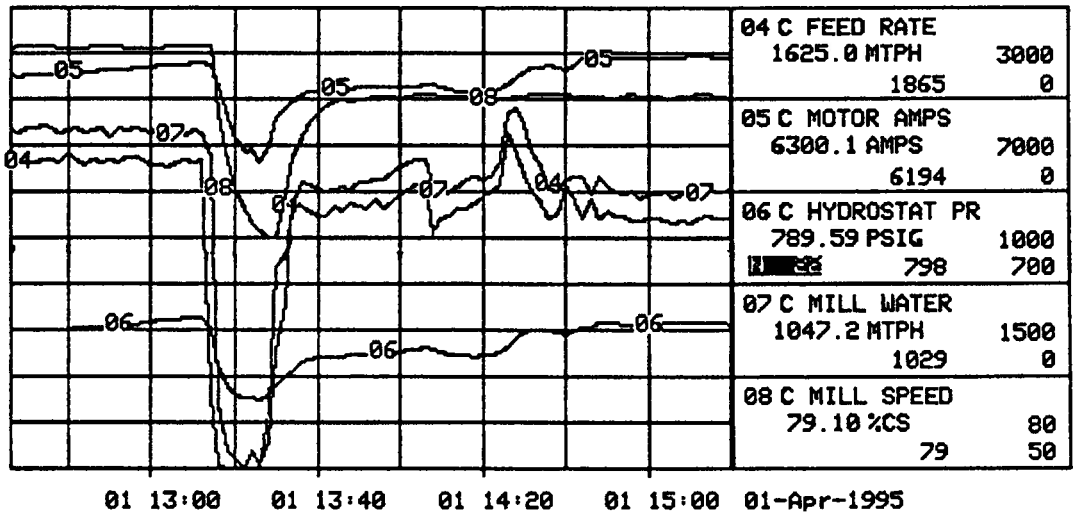
At 9:07 hrs., the power draw reached the absolute maximum limit, at which point all the feeders were automatically turned off by the interlocking system. The ISCS correctly detected and reported this restriction as an RR_ORE_SUPPLY. The ISCS estimated a loss of 128 tons; however, since the restriction duration was shorter than 10 min., these losses were discarded. A metallurgist might have also discarded the losses caused by this short restriction.

From a close analysis of the operating conditions around this short restriction, we conclude that this restriction and the associated 128 tons lost could have been prevented. Given the appropriate heuristic knowledge, the ISCS could analyze the evolution of process variables to determine when this type of restriction is likely to happen. At that time, the ISCS could either warn the operators or intervene directly with the operation of the process. This would give the ISCS the ability to not only estimate losses caused by restrictions but also prevent their occurrence.

Between 9:15 and 10:19 hrs., the ISCS detected a series of very short soft ore restrictions. None of these restrictions was longer than the minimum time pre-specified (10 min.); thus, the ISCS did not accumulate the corresponding tonnage losses. A metallurgist, however, would likely have reported

the occurrence of a single soft ore restriction during this period, with losses of approximately 100 tons.

ii) Fresh feed to the mill is cut (RR_FEED_CUT)



13:12 hrs. Feeder #2, 3, 4 and 5 stop
13:28 hrs. Feeder #2, 3, 4 and 5 start
13:50 hrs. Feeder #5 stops
14:05 hrs. Feeder #4 stops
14:22 - 14:48 hrs. All feeders start and stop several times.

Figure 6.10 Restriction RR_FEED_CUT: April 1, day shift.

The ISCS reported the following restrictions during this period of time.

TIME	RESTRICTION	DURATION	TONNAGE LOST
13:13 - 13:15 hrs.	RR_ORE_SUPPLY	2 min.	39 tons (too short)
13:15 - 13:27 hrs.	RR_FEED_CUT	12 min.	351 tons
13:27 - 13:29 hrs.	RR_ORE_SUPPLY	2 min.	34 tons (too short)
14:06 - 14:22 hrs.	RR_ORE_SUPPLY	16 min.	42 tons

Metallurgists considered the restriction that occurred at 13:13 hrs. as a single restriction that lasted until 13:29 hrs. The ISCS, however, reported the occurrence of three restrictions. Before and after the main restriction RR_FEED_CUT, the ISCS indicated the occurrence of RR_ORE_SUPPLY restrictions.

The total losses estimated by the ISCS for the restrictions detected between 13:13 and 13:29 hrs. are 424 tons, which is very close to the value of 450 tons estimated by an expert from the trends measurements.

Metallurgists differ about whether the entire period between 13:29 and 14:22 hrs. should be considered as a restriction, or just the period between 14:02 and 14:22 hrs. This would obviously lead to different estimate of tonnage losses. Losses for the entire period are estimated to be approximately 170 tons; for the shorter period the estimate is 40 tons.

The ISCS reported the occurrence of a restriction for the period between 14:06 and 14:22 hrs.; it reported losses of 42 tons—very close to a metallurgist's estimate. The ISCS heuristics can be tuned to follow either view as supported by the metallurgists.

iii) Fresh Feedrate High Limit (RR_FEED_LIMIT)

This restriction did not occurred during the period of evaluation of the ISCS.

iv) Feeders are down (RR_FEEDER_DOWN)

Some feeders tend to take coarser ore than others due to natural particle segregation in the stockpile. This causes a change in the size composition of the fresh feed changes when a feeder goes down. Variations in size composition may produce either a decrement or an increment in the tonnage that

can be processed by the mill, as the ore becomes coarser or finer respectively. In the case shown here, the tonnage increases after a feeder goes down.

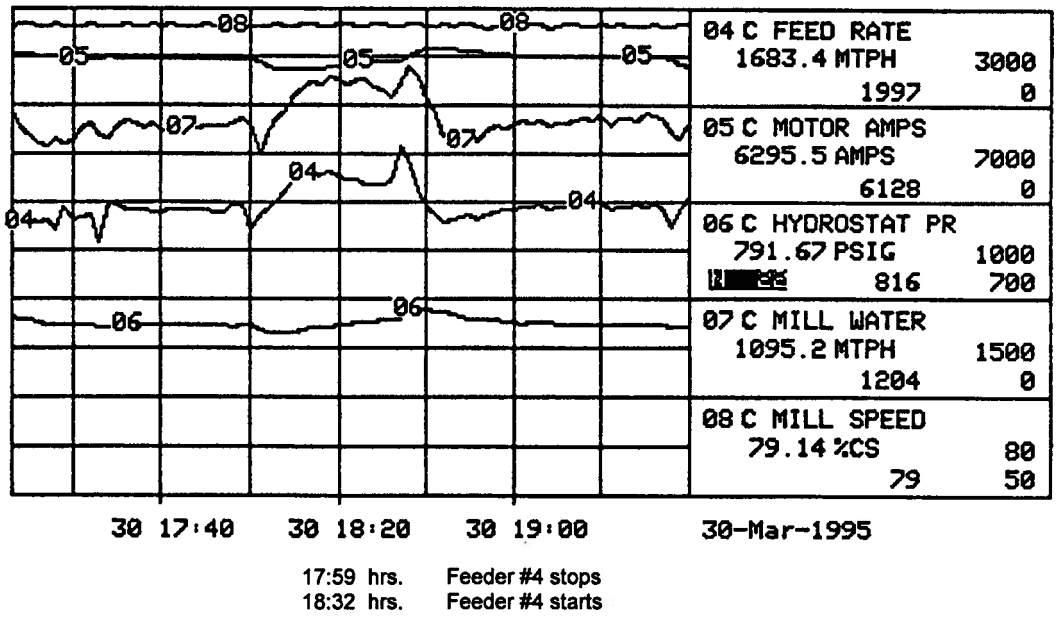


Figure 6.11 Restriction RR_FEEDER_DOWN: March 30, day shift.

The ISCS reported the following restrictions during this period of time.

TIME	RESTRICTION	DURATION	TONNAGE LOST
17:59 - 18:03 hrs.	RR_ORE_SUPPLY & RR_FEEDER_DOWN	few short restrictions	-------
18:03 - 18:32 hrs.	RR_FEEDER_DOWN	28 min.	0 tons

Figure 6.11 shows the case of feeder #4 going down. This feeder normally delivers finer ore, so we expected a decrease in the tonnage processed by the mill after this feeder went down. However, the tonnage increased on this particular occasion.

The ISCS appropriately detected this potential restriction when feeder #4 went down. When the restriction was lifted at 18:32 hrs., the ISCS reported zero losses.

A non-experienced metallurgists may overlook the irregularities shown in the historic trends for this period, indicating that there is no restriction. An experienced metallurgist, however, would be able to detect the occurrence of this restriction. After a careful analysis of this period of time, he would note that there are no tonnage losses associated with the restriction. Thus, agreeing with the results reported by the ISCS.

During the initial 4 min. of this restriction, between 17:59 and 18:03 hrs., the ISCS indicated that short RR_ORE_SUPPLY & RR_FEEDER_DOWN restrictions occurred a couple of times.

v) Problems with ore supply (RR_ORE_SUPPLY)

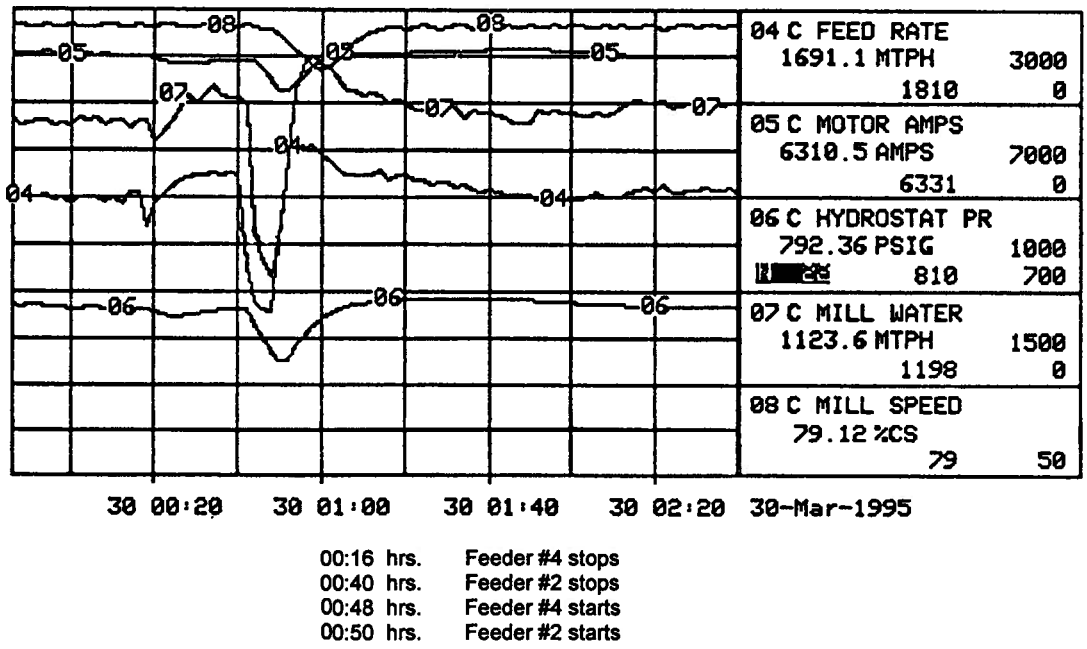


Figure 6.12 Restriction RR_ORE_SUPPLY. March 30, night shift.

The ISCS reported the following restrictions for this period of time.

TIME	RESTRICTION	DURATION	TONNAGE LOST
00:17 - 00:21 hrs.	RR_FEEDER_DOWN & RR_ORE_SUPPLY	several alternating short restrictions	--------
00:21 - 00:41 hrs.	RR_FEEDER_DOWN	20 min.	2 tons
00:41 - 00:51 hrs.	RR_ORE_SUPPLY	10 min.	97 tons
00:51 - 01:00 hrs.	RR_SOFT_ORE	9 min.	9 tons (too short)

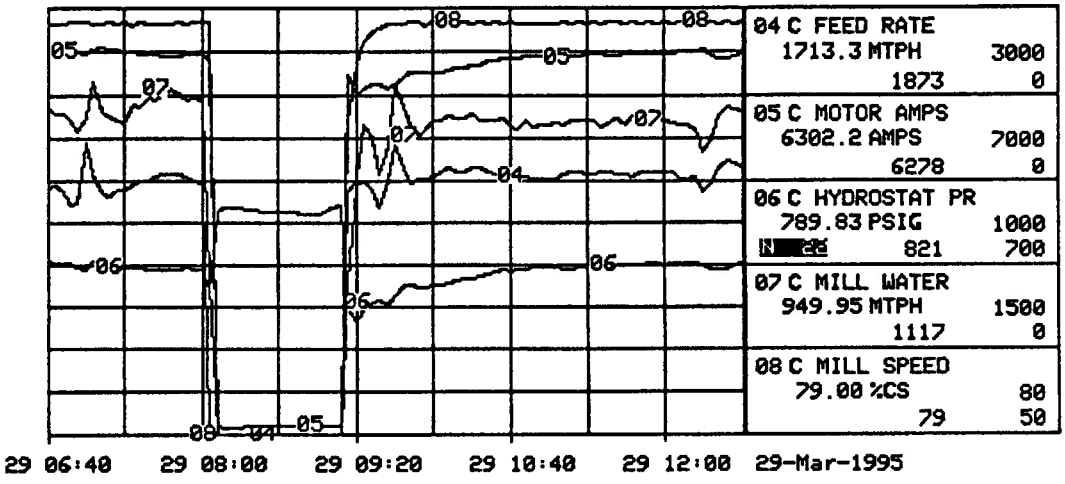
When feeder #4 went down at 00:16 hrs., the size composition of the feed to the mill changed. As a result of this, the tonnage processed by the mill increased (similar to the restriction shown in Figure 6.11, discussed earlier). The ISCS detected this restriction and reported a loss of 2 tons. These losses occurred right after the feeder went down, before the control system could compensate for the sudden drop in fresh feedrate.

When feeder #2 went down at 00:40 hrs., the remaining feeders could no longer continue supplying the tonnage rate requested. As a result, the fresh feedrate to the mill dropped, and so did the power draw as shown in Figure 6.12.

The ISCS correctly detected this restriction, reporting a tonnage loss of 97 tons. A metallurgist, on the other hand, would roughly estimate a loss of 110 tons, slightly higher than that reported by the ISCS. The difference in the estimated losses is caused by the highly subjective aspect of this problem, i.e. the estimation of the fresh feed rate that could have been processed by the mill if feeder #2 had not stopped. The metallurgist estimated a loss of 110 tons by assuming that the mill would continue to process the same tonnage average that existed just prior to the time feeder #2 went down. The ISCS, on the other hand, obtained its estimate by assuming that the mill would process the same tonnage rate that it was processing before feeder #4 went down. The ISCS uses a period of

operation with no restriction, such as the one before the restriction caused by feeder #4, to estimate the tonnage that the mill may continue processing.

vi) Mill is shut down (RR_SAG_STOPS)



08:01 hrs. Feeder #2, 3, 4 and 5 stop (feeder #1 had already stopped)
08:04 hrs. SAG mill stops
08:40 hrs. Both ball mills stop
09:13 hrs. SAG mill and ball mills start
09:15 hrs. Feeder #2, 3, 4 and 5 start

Figure 6.13 Restriction RR_SAG_STOPS: March 29, day shift.

The ISCS reported the following diagnostic results

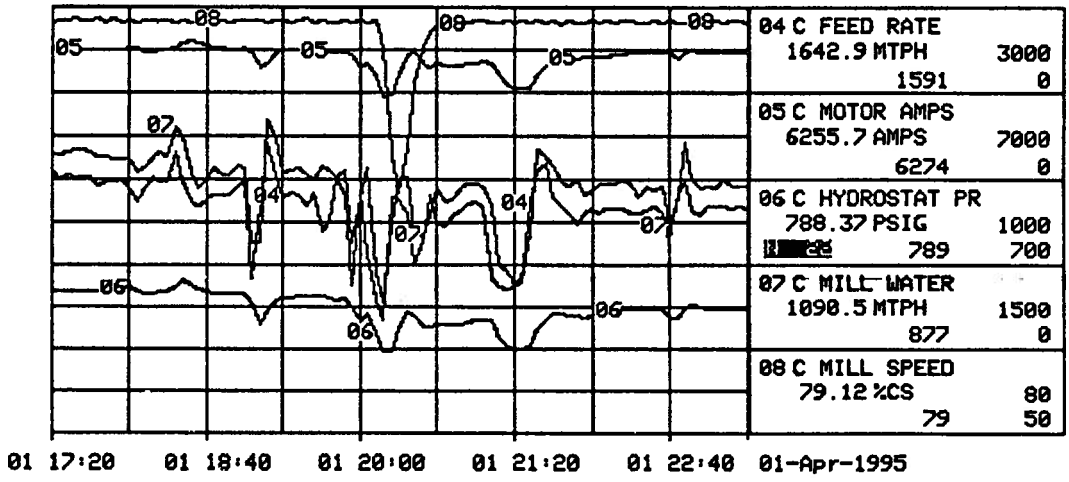
TIME	RESTRICTION	DURATION	TONNAGE LOST
08:02 - 08:03 hrs.	RR_ORE_SUPPLY	1 min.	25 tons (too short)
08:03 - 08:03 hrs.	RR_FEED_CUT	0 min.	--------
08:03 - 09:13 hrs.	RR_SAG_STOPS	1 hr. 10 min.	2112 tons
09:13 - 09:15 hrs.	RR_FEED_CUT	2 min.	45 tons (too short)
09:15 - 09:19 hrs.	RR_ORE_SUPPLY	4 min.	31 tons (too short)

The shut down of the mill involves stopping all feeders. As shown in Figure 6.13, the feeders were stopped just prior to the shut down of the mill. The start-up of the mill is a reverse procedure: the mill is re-started first with no feed, and then the feeders are re-started.

This shut down procedure caused some difficulties for the ISCS. The ISCS, instead of reporting only one restriction, reported five restrictions: the main RR_SAG_STOPS restriction, plus two short restrictions just before and two just after the main restriction. The ISCS reported a loss of 2112 tons associated with the main restriction; it discarded the losses caused by the short restriction, which amount to 101 tons.

A metallurgist would have considered the whole period from 08:02 hrs. until 09:19 hrs. as a single restriction. Tonnage losses estimated from the historic trends for this period amount to 2250 tons. This estimate is very close to the total losses of 2213 tons that the ISCS would have reported if all five restrictions had been considered as one single restriction.

vii) Operating period with several disturbances



18:00 - 21:49 hrs. All feeders start and stop many times

Figure 6.14 Many restrictions due to plugged chutes. April 1, day and night shifts.

The ISCS reported the following major restrictions during this period of time.

TIME	RESTRICTION	DURATION	TONNAGE LOST
18:01 - 18:22 hrs.	RR_FEEDER_DOWN	21 min.	7 tons
20:02 - 20:16 hrs.	RR_ORE_SUPPLY	14 min.	125 tons
20:16 - 20:35 hrs.	RR_FEEDER_DOWN	19 min.	61 tons
20:38 - 21:06 hrs.	RR_FEEDER_DOWN	28 min.	15 tons
21:06 - 21:29 hrs.	RR_ORE_SUPPLY	23 min.	177 tons
		Total loss:	**385 tons**

It is not an easy task to detect restrictions and estimate tonnage losses for operating periods such as that shown in Figure 6.14. A metallurgist would have identified two major restrictions: one at 20:01 hrs. and another at 21:06 hrs. Tonnage losses associated with these two restrictions are estimated to be 100 tons and 150 tons, respectively. Some metallurgists may also report a rough estimate of overall losses due to the other short restrictions present during this period. They would not, however, estimate this on a detailed restriction-by-restriction basis.

The ISCS reported five major restrictions during the period shown in Figure 6.14; it also discarded several short restrictions detected during this period. The total losses reported by the ISCS were 385 tons, higher than those reported by a metallurgist. The major reason for this difference is that the ISCS does a much more detailed diagnosis of the operation of the circuit; a metallurgist is likely to consider major restrictions only.

6.8 Evaluation of Results

The following is a summary of the findings obtained from an analysis of the results presented in the previous section.

i) Overall ISCS Performance

The overall performance of the ISCS is very encouraging. The system could appropriately detect and diagnose every restriction that affected the operation of C-line during the evaluation period. Tonnage losses estimated by the ISCS are very close to those obtained from historic trends, reported by HVC personnel.

The current set of heuristics incorporated into the ISCS covers every restriction of interest to HVC personnel. It does not consider, however, all possible circumstances in which a restriction may occur. The ISCS, for example, detects a high tonnage limit (RR_FEED_LIMIT) when the operator uses the high-limit block to constrain production (see Figure 6.3); the ISCS cannot detect this restriction properly if the operator imposes this high limit by other means, e.g. by turning the power loop to manual operation and manipulating directly the output of the fresh feed controller.

The performance of the ISCS was analyzed on a restriction-by-restriction basis. It was difficult to do this for a long operating period due to problems with the HVC-PC, the computer that provides the ISCS with access to the Bailey database. The HVC-PC crashed randomly every few days throughout the evaluation period. Because of these crashes, the ISCS sometimes missed restrictions that produced significant production losses. These hardware problems have since been eliminated and the system has operated essentially free of hardware and software errors from April 1 to May 1, 1995.

The period from March 26 to March 31, 1995, was chosen for evaluation of system output as it represented a period of time for which reliable data were available. The trend charts of this period

were analyzed manually using the heuristic approach of Hans Raabe. This comparison appears in Table 6.2.

Table 6.2 Diagnostic results provided by the ISCS. Period: March 26 - 31, 1995

RESTRICTION CAUSE	NUMBER OF RESTRICTIONS		DURATION hh:mm		TONNAGE LOST tons	
	ISCS	METALLURGIST	ISCS	METALLURGIST	ISCS	METALLURGIST
SAG STOPS [1]	5	5	34:45	35:30	54649	56940
FEEDER PROBLEMS	9	2	03:50	01:05	1019	1050
CHUTES	13	4	02:44	00:57	423	485
SOFT ORE	7	0	02:07	00:00	55	0
OTHER	3	1	00:57	00:35	8	450
NOT DETECTED [2]	2	------	00:45	-------	550	-------
TOTAL	**34**	**7**	**09:38**	**2:37**	**2055**	**1985**

(1) This is detected by the ISCS but not integrated with the other causes of restriction. HVC keeps a separate record of it.

(2) The ISCS missed two restrictions as explained in the text.

The total tonnage losses reported by HVC for this period was 2100 tons, which compared with the manual total in Table 6.2, demonstrates the variability of different metallurgists conducting this analysis. The important point, however, is that the ISCS estimate is clearly within the range of these two manually derived predictions.

The difference between the results provided by the ISCS, those reported by HVC, and those obtained using Raabe's heuristics (see Table 6.2), is due to three factors. First, The ISCS performs a much more meticulous diagnosis; it does not only report restrictions responsible for significant losses but also those that only cause minor losses. Metallurgists, however, tend to consider losses caused by major restrictions only. This accounts for the big difference in both number and duration of restrictions shown in Table 6.2. During this period of evaluation, for example, the ISCS detected a

number of small soft ore restrictions, which produced estimated total losses of 55 tons. HVC personnel, however, overlooked these small restrictions and their corresponding losses. Second, the set of heuristics in the ISCS is incomplete: it does not cover all the possible circumstances in which a restriction may occur. During the evaluation period, HVC operators imposed a high limit to the setpoint of fresh feed by a mechanism not covered by the heuristics available in the ISCS (they turned the power draw loop to manual mode and manipulated the setpoint of the fresh feed controller). Hence, the ISCS could not appropriately detect this problem. This restriction was responsible for a loss of 450 tons, as estimated from historic trends.

Finally, there were some problems with the HVC-PC, the computer that interfaces with the Bailey database. The HVC-PC was down when a restriction occurred preventing the ISCS from detecting it. Losses due to this restriction were estimated at 100 tons.

ii) Using QM to estimate tonnage losses

This application has provided a good opportunity to demonstrate the capabilities of qualitative modeling.

Every aspect of the diagnostic problem involves qualitative information and subjective analysis—a challenge to any knowledge-based system. The most difficult and subjective aspect of this application is the estimation of tonnage losses due to soft ore restrictions. How much tonnage can the mill process, after the fresh feed has reached the high limit target specified for C-line, in order to keep the power draw at its setpoint value? (see Figure 6.9). Different people have different answers to this problem. This is an aspect that causes serious difficulties to conventional knowledge-based systems; they do not provide an appropriate tool to handle this kind of problem. The results of the application of the ISCS demonstrate that QMs, under the pseudo-qualitative modeling approach proposed in this research, can appropriately handle such problems.

A steady state QM implemented into the ISCS proved to be an essential tool to estimate losses caused by soft ore restrictions. This QM represents the mental model of a metallurgist regarding the relationship between the fresh feed to the mill and the power draw. The mental model can be expressed as follows: the additional tonnage required to bring the power back to its setpoint value is proportional to the difference between the actual power draw and its setpoint value. The ISCS uses this QM to predict a fresh feed value that could have been processed by the mill during the occurrence of the soft ore restriction. Finally the ISCS estimates the losses based on the difference between the predicted fresh feedrate value and the high limit imposed on the fresh feed setpoint.

iii) Revision of the Current Set of Heuristics

The ISCS detected every restriction that affected the operation of C-line (provided that they were covered by its current set of heuristics). Most of these restrictions were appropriately detected, a few, however, caused some difficulties for the ISCS. As discussed below, the deficiencies found in the ISCS heuristics can be solved by modifying/expanding the current set of heuristics.

The ISCS split some soft ore restrictions into several shorter ones as in the case shown in Figure 6.9. According to the ISCS report, the restriction in Figure 6.9 occurred intermittently during this period, which in strict terms, is correct. The setpoint for the fresh feed oscillated continuously around the target tonnage rate for C-line during this period. Current heuristics associated with soft ore restrictions are sensitive to this oscillation; thus the ISCS switched the restriction condition on and off. A metallurgist, however, would conclude that it is more appropriate to consider that only one restriction occurred during this period. This problem can be eliminated by modifying current ISCS heuristics in order to reduce their sensitivity to the oscillations described earlier.

In some cases, the ISCS reported the occurrence of several shorter restrictions when only a single restriction occurred. When the fresh feed was cut (see Figure 6.10), for example, the ISCS first indicated a restriction due to feeders (feeders stopped), then a restriction due to ore supply (the fresh feed dropped below its setpoint), and finally, the restriction due to a fresh feed cut. Similar splitting occurred when the restriction was lifted and the fresh feed restored. The ISCS requires additional heuristics that instruct the system that shorter restriction may be detected before and after the occurrence of a main one, but all such irregularities correspond to the same restriction. These new heuristics could be arranged as supervisory heuristics dedicated to the detection of specific restrictions.

The ISCS also misdiagnosed some restrictions. This normally occurred following the removal of a major restriction. When the mill restarts after a shut down, for example, the control system would request a high value of fresh feed to bring the power draw back to its setpoint value. In some cases, due to requests by the control system, the fresh feed may reach the maximum tonnage value allowable for C-line. When this happened, the ISCS reported the occurrence of a soft ore restriction. If we analyze only the value of relevant variables during that time, ignoring that the mill has just re-started, we would also conclude that a soft ore restriction is affecting the operation. However, considering the above in the right context, we would conclude that what has occurred is the natural response of the control system and not that of a soft ore restriction. This could be solved by fine-tuning the existing heuristics and adding others designed to supervise and monitor partial diagnostic results.

The 10 minute-criteria incorporated into the ISCS to decide whether or not a restriction is too short to be considered needs to be revised. Depending on specific cases, a metallurgist may include a restriction shorter than 10 min. and discard others longer than 10 min. The ISCS should exhibit a similar degree of flexibility.

iv) Future expansion of the Heuristic Knowledge

At this stage the ISCS operates as an real-time advisory system; it does not intervene directly with the process. The ISCS provides on-line reports about restrictions affecting circuit operation; appropriate actions are left to the discretion of plant operators. A future expansion of the ISCS is to give the system the ability to intervene with the operation in order to prevent the occurrence of some tonnage restrictions.

For example, the ISCS could override the control system to prevent the cut in tonnage shown in Figure 6.9. This upset occurs because the conventional control system cannot distinguish between a high limit imposed on the tonnage by operators and one due to a soft ore restriction. In both cases, the control system reduces mill speed in order to maintain a constant mill inventory of ore. In the case of a soft ore restriction, this action eventually produces a rapid increase in power draw as the mill loses its grinding capacity (see Figure 6.9). When the mill power draw reaches the absolute maximum allowed, the interlocking system stops the feeders causing the upset to circuit operation. The ISCS, however, is able to distinguish between these two situations. Appropriate heuristics could be incorporated into the ISCS to override the existing control system before this upset occurs. This would result in reduction of tonnage losses due to restrictions and in stable feed tonnage.

CHAPTER 7

FINAL CONCLUSIONS AND RECOMMENDATIONS

7.1 Final Conclusions

i) Pseudo-Qualitative Modeling

While other researchers have proposed incorporation of heuristics with mathematical models for supervisory or adaptive control, this work demonstrates that such components can, in fact, be integrated into a singular overall modeling approach, called Pseudo-Qualitative Modeling. This methodology will allow system developers to attack the problem of real-time supervision even when they do not have a mathematical model or understanding of a complex process. Alternatively, should such models exist, it is easy to incorporate these within the arena of Pseudo-Qualitative Modeling.

- Pseudo-Qualitative Modeling is proposed as a suitable method to build supervisory control systems that requires heuristic knowledge and/or input and/or mathematical models.

ii) Components of Pseudo-Qualitative Modeling

Pseudo-Qualitative Modeling offers the same versatility as purely qualitative approaches to handle models based on heuristic knowledge. In addition, the approach proposed can handle well defined and accurate models including mathematical models. In fact, the approach provides a framework to integrate qualitative and numerical models into a knowledge-based system.

A pseudo-QM is conceived as a model with a hybrid structure capable of handling both numerical and qualitative information. Qualitative information is represented by a set of fuzzy concepts that can be directly translated into any other user-defined set, which facilitates the integration of QMs with other elements of an ISCS. Pseudo-QMs offer qualitative primitives that allow the representation of both steady state and dynamic behavior.

- A pseudo-QM can appropriately represent the behavior of complex processes. Its parameters can be tuned as more knowledge becomes available or by mechanisms that adjust its parameters in relation with process variables.

- A pseudo-QM can be fully integrated with other elements of knowledge representation and reasoning within the environment of an ISCS.

iii) Application of Pseudo-Qualitative Modeling to a real world process

Qualitative models, under the approach proposed, proved to be a valuable tool to represent and handle heuristic models. A QM incorporated into the ISCS mimics the heuristic knowledge that metallurgists possess about the relationship between power draw and fresh feed into the mill. The use of this QM was essential in estimating tonnage losses caused by operating restrictions that affected the operation of the process.

The ISCS provides periodic reports that can be used to assess the extent and frequency of production-limiting bottlenecks. On-line information together with report summaries, can reduce the decision-making time. This can ultimately lead to a reduction of tonnage losses, which range up to 5000 tons per week in C-line. In addition, the ISCS provides consistent criteria to deal with this diagnostic problem and can save about 5 hours per week of a metallurgist's time.

- The application of the ISCS to the diagnosis of tonnage restrictions in a grinding line at HVC has demonstrated the validity of the proposed pseudo-qualitative modeling approach. The feasibility of incorporating this technique into a commercial real-time SCADA system widely used in industrial applications has been proven.

- An ISCS, with the structure and elements proposed in this research, can effectively be applied to a real-time full-scale industrial operation. In the HVC application, the ISCS dealt with aspects that are difficult to tackle with tools provided by current knowledge-based systems. The ISCS could mimic human expertise and heuristic knowledge to handle all the subjective aspects involved in this application.

- ProcessVision was found to be a powerful tool for the development and implementation of the ISCS. Its versatility played an important role in reducing the development time of this system. The modularity of ProcessVision together with features of each individual module, were essential factors for the system to be designed to mimic the heterogeneous functions used by a human to reason about complex processes.

iv) Handling Uncertainty

Probability has not been widely accepted by researchers in intelligent control mainly due to a lack of an appropriate approach to handle heuristic knowledge and fuzzy concepts. A simplified framework to handling uncertainty based on subjective probability is proposed along the lines described in Appendix A.1. This approach is suitable to represent and handle concepts such as those required for ISCS process supervision applications.

This approach has various similarities to existing numerical approaches to certainty; its major difference is that it employs the widely respected probability calculus to handle uncertainty. In

summary, this is a contribution of another item to the arsenal from which we can select the most appropriate tool for a given application.

7.2 Recommendations for Future Research

Research should continue along the following lines:

- Expand the ability of current pseudo-qualitative modeling approach to include subjective probability analysis described in Appendix A.1.

- Develop a philosophy on intelligent learning along the lines outlined in Appendix A.2. Initially the focus should be on explanation and justification, moving on to adaptation of linguistic definitions in response to heuristics, then eventually leading to development of pseudo-QMs that can add new knowledge elements.

- Extension of the ISCS developed for HVC to supervisory control to intervene with the operation of the control system to eliminate abnormal operations such as tripout of feeders.

- Addition to the HVC ISCS system of new heuristics to identify the onset of certain restrictions and to combine several short intermittent restrictions into a single major.

- Examination of other AI methodologies that might interact effectively with pseudo-QM within the environment of an ISCS; methodologies such as artificial neural networks, genetic algorithms and cluster analysis for such problems as pattern recognition, optimization and classification.

CHAPTER 8

CLAIMS TO ORIGINAL RESEARCH

I claim the following items as original research deriving from this work:

- The proposition that heuristic knowledge together with numerical models can be integrated into an approach for real-time supervision of direct control of a complex process. Such a technique is referred to as Pseudo-Qualitative Modeling.

- The definition of the components and structure of a Pseudo-Qualitative Model, and verification of this approach to model a simple well-understood process (head tank).

- The incorporation of a pseudo-qualitative modeling facility into a commercial SCADA software package to create an ISCS system to monitor in real-time a full-scale industrial process.

- The proposition of an uncertainty handling technique, based on subjective interpretation of probability that can be used in an ISCS system employing pseudo-qualitative modeling.

NOMENCLATURE

i) List of Symbols [1]

AMT	: Amount
CNT	: Constant
DEC	: Decreasing
DYN	: Dynamics
INC	: Increasing
NEG	: Negative
POS	: Positive
min	: minimum
max	: maximum
$\neg$	: NOT (negation)
$\wedge$	: (logical) AND
α	: Qualitative proportionality
	: Belongs to
μ	: Membership function for fuzzy sets
τ	: Time constant

ii) Acronyms [1]

AI	: Artificial Intelligence
DCS	: Distributed Control System
DoB	: Degree of Belief
EBL	: Explanation-Based Learning
ES	: Expert System
HVC	: Highland Valley Copper
IC	: Intelligent Control

---

(1) Other symbols and acronyms used in the text are described whenever they are used.

ISCS	: Intelligent Supervisory Control System
KB	: Knowledge Base
KR	: Knowledge Representation
OR	: Operations Research
PC	: Personal Computer
PID	: Proportional-Integral-Derivative
QM	: Qualitative Model
RCS	: Regulatory Control System
SAG	: Semi-autogenous grinding
SCADA	: Supervisory Control And Data Acquisition
UI	: User Interface

iii) Units

cm	: Centimeters
hp	: Horse power
hrs.	: Hours
KB	: Kilo Bytes
KW	: Kilo Watts
lpm	: Liters per minute
m	: Meter
min.	: Minute
MB	: Mega Bytes
% crit	: Percent of critical speed

BIBLIOGRAPHY

Abdulmajid B.A. and Wynne R.J., (1991). "An Extended Qualitative Controller", *IEE Int. Conference on Control'91*, Edinburgh, UK, March, pp 606-611.

Andersen T.R. and Nielsen S.B., (1985). "An Efficient Single Output Fuzzy Control Algorithm for Adaptive Applications", *Automatica*, Vol 21, No 5, pp 539-545.

Anderson J.A. and G.E. Hinton, (1981). "Models of Information in the Brain". In Hinton G.E. & Anderson J.A., (Eds.)., *Parallel Models of Associative Memory*. Lawrence Erlbaum Associate, Publ., NJ, USA.

Antsaklis P.J., Passino K.M. and Wang S.J., (1989). "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues", *Journal of Intelligent and Robotic Systems*, Vol. 1, pp 315-342.

Antsaklis P.J., Passino K.M. & Wang S.J., (1990). "An Introduction to Autonomous Control Systems", *IEEE Int. Symposium on Intelligent Control*, Philadelphia, PA, USA, September, pp 21-26.

Arjunan W. M., (1988). "Diagnosing Multiple Faults in Intelligent Controls and Automated Systems", *Third IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 93-97.

Åström K.J., Anton J. and Årzén K., (1986). "Expert Control", *Automatica*, Vol 22, N° 3, pp 277-286.

Åström K.J., (1989). "Toward Intelligent Control", *IEEE Control Systems Magazine*, April, pp 60-64.

Åström K.J. and Wittenmark B., (1989). *Adaptive Control*, Addison-Wesley Publishing Co.

Bartrum J., Bowler A. and Butcher G., (1986). "SAG Mill Operations at Kidston Gold Mines", *Minerals and Metallurgical. Processing*, May, pp 96-103.

Berenji H., (1990). "Neural Networks and Fuzzy Logic in Intelligent Control", *IEEE Int. Symposium on Intelligent Control*, Philadelphia, PA, USA, September, pp 916-920.

Bhaskar R. and Nigam A., (1990), "Qualitative Physics Using Dimensional Analysis", *Artificial Intelligence*, Vol 45, No 1-2, September, pp 73-111.

Bösser T.E., (1985). "Learning to control Complex Technical Systems", in Willumeit H.P., (Ed.), *Human Decision Making and Manual Control*, Elsevier Science Publ., pp 319-328.

Bowerman R. and Glover D.E., (1988). *Putting Expert Systems into Practice*. Van Nostrand Reinhold Co.

Burrows M.J. Carriere J.D., Leung J. and Laguitton D., (1989). "An Expert System Designed to Improve Reliability of Assay Information in a Flotation Plant", *21st CMP Conference*, Ottawa, ON, Canada, January.

Cheeseman P., (1985). "In Defense of Probability", *Proc. of the Ninth IJCAI*, Los Angeles, CA, USA, August, pp 1002-1009.

Cheeseman P., (1986). "Probabilistic Versus Fuzzy Reasoning", in Kanal L.N. and Lemmer J.F. (Eds.), *Uncertainty in Artificial Intelligence*, Elsevier Science Publ. B.V., North Holland, pp 85-102.

Cifuentes E.I. and Mular A.L., (1992). "An Introduction to Qualitative Modeling and its Potential in the Development of Intelligent Control Systems", *Conference and Exhibition on Industrial Automation*, Montreal, PQ, Canada, June, pp 1.5-1.8.

Clarke D.W., (1981). "Implementation of Adaptive Controllers", In Harris C.J. and Billings S.A., (Eds.), *Self-tuning and Adaptive Control*, Peter Peregrinus, UK.

Clarke D.W. and Gawthrop P.J., (1981). "Implementation and Application of Microprocessor-Based Self-Tuners", *Automatica*, Vol 17, No 1, pp 233-244.

Cohen P.R., (1985). *Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach*, Pitman Publ. Inc.

Corea R., Tham M.T. and Morris A.J., (1992). "Qualitative Modeling - An Application to Supervisory Control of a Distillation Column", *IEEE Int. Symposium on Intelligent Control*, Glasgow, Scotland, UK, August, pp 586-591.

Crosscope J.R. and Bonnell R.D., (1988). "A Dynamically Evolving Learning Network for Intelligent Control", *IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 529-533.

D'Ambrosio B., (1989a). "Extending the Mathematics in Qualitative Process Theory", *Int. Journal of Intelligent Systems*, Vol 4, pp 55-80.

D'Ambrosio B., (1989b). *Qualitative Process Theory Using Linguistic Variables*, Springer-Verlag.

Davis R., (1985). "Diagnostic Reasoning Based on Structure and Behavior", *Artificial Intelligence*, Vol 24, pp 347-410.

DeJong G., (1988). "An introduction to Explanation-based Learning", in Shrobe H.E., (Ed.), *Exploring Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., Ch. 2.

de Kleer J, (1976). *Local Methods for Localizing Faults in Electronic Circuits*, MIT, Cambridge, MA, USA.

de Kleer J. and Brown J.S., (1985). "Qualitative Physics Based on Confluences", in Hobbs J.R. & Moore R.C., (Eds.), *Formal Theories of the Commonsense World*, Ablex Publ. Corp., Norwood, NJ, USA, pp 109-183.

de Kleer J. and Brown J.S., (1986). "Theories of Causal Ordering", *Artificial Intelligence*, Vol 29, No 1, pp 33-61.

de Kleer J. and Williams B.C., (1987). "Diagnosing Multiple Faults", *Artificial Intelligence*, Vol 32, pp 97-130.

Drury C. G., (1976). "Human Performance in Manual Process Control", in Sheridan T.B. and Johannsen G. (Eds.), *Monitoring Behavior and Supervisory Control*, Plenum Press, New York, pp 359-369.

Dubois D. and Prade H., (1988). "The Treatment of Uncertainty in Knowledge-Based Systems Using Fuzzy Sets and Possibility Theory", *Int. Journal of Intelligent Systems*, Vol 3, pp 141-165.

Efstahiou H.J., (1990). "Introduction to Knowledge-Based Systems for Process Control", Ch. 2 in McGhee J., Grimble M.J. & Mowforth P., (Eds.), *Knowledge-Based Systems for Industrial Control*, Peter Peregrinus Ltd., London, UK.

Eggert J. and Benford P., (1994). "Development and implementation of an expert system for grinding control at The Dome Mine," *26th Meeting of the Canadian Mineral Processors*, Ottawa, Ontario, Canada, January.

Fischler M. and Firschein O., (1987). *Intelligence. The Eye, the Brain and the Computer,* Addison-Wesley Publ. Co.

Flintoff B.C., and Mular A.L. (Eds.), (1992). *A Practical Guide to Process Controls in the Minerals Industry*, University of British Columbia, Brenda Mines Ltd. and MINTEC, Ch 2.

Forbus K.D., (1984). "Qualitative Process Theory", *Artificial Intelligence*, Vol 24, No 1-3, December, pp 85-168.

Forbus K.D., (1985), "The role of Qualitative Dynamics in Naive Physics", in Hobbs J.R. & Moore R.C., (Eds.), *Formal Theories of the Commonsense World*, Ablex Publ. Corp., Norwood, N.J., USA, pp 185-226

Forbus K.D., (1986a). "Qualitative Physics: Past, Present, and Future", in Shrobe H.E. & American Association for AI, (Eds.), *Exploring Artificial Intelligence: Survey Talks from the National Conferences on AI*, Morgan Kaufmann Publ., Inc., California, USA, pp 239-296.

Forbus K.D., (1986b). "Interpreting Measurements of Physical Systems", *AAAI-86*, Philadelphia, PA, USA, August, pp 113-117.

Forbus K.D., (1988). "Commonsense Physics: A Review", *Annual Review of Computer Science*, pp 197-232.

Forsyth R and Rada R., (1986). *Machine Learning: Applications in Experts Systems and Information Retrieval*, Ellis Horwood Ltd.

Forsyth R.(Ed.), (1989). *Machine Learning. Principles and Techniques,* Chapman and Hall.

Fortescue T.R., Kershenbaum L.S. and Ydstie B.E., (1981). "Implementation of Self-Tuning Regulators with Variable Forgetting Factors", *Automatica*, Vol 17, No 6, pp 831-835.

Fox J., (1986). "Three Arguments for Extending the Framework of Probability", in Kanal L.N. & Lemmer J.F., (Eds.), *Uncertainty in Artificial Intelligence*, Elsevier Science Publ. B.V., North Holland, pp 447-458.

Franklin G.F. and Powell J.D, (1980). *Digital Control of Dynamic Systems.* Addison-Wesley Publishing Co.,

Fu K.S., (1971). "Learning Control Systems and Intelligent Control Systems: An Intersection of Artificial Intelligence and Automatic Control", *IEEE Trans. on Automatic Control*, Vol AC-16, No 1, pp 70-72.

Fuentes A., Cifuentes E.I., (1988). "Design of a Selftuning Regulator with Feed Forward Signals". *VII Brazilian Conference on Automation*, Aeronautic Technical Institute, Sao Paulo, Brazil, August.

Geng Z. and Jamshidi M., (1988). "Design of Self-Learning Controllers Using Expert System Techniques", *IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 551-558.

Hales L.B., Burdett B.W. and Gilbert S.R., (1982). "The History and Development of Grinding Control", in Mular A.L. and Jergensen (Eds.), *Design and Installation of Comminution Circuits*, Ch 44, SME-AIME, Littleton, CO, USA.

Hales L.B. and Pate W.T., (1985). "Supervisory and Optimal Control Strategies for Mineral Processing Plants", *Conference of Metallurgists*, Vancouver, BC, Canada. August.

Hales L.B., Vanderbeek J.L. and Herbst J.A., (1988). "Supervisory Control of a Semi-Autogenous Grinding Circuit", *Int. Journal of Mineral Processing*, Vol 22, pp 297-312.

Hamscher W. and Davis R., (1984). "Diagnosing Circuits with State: An Inherently Under Constrained Problem", *AAAI-84*, Austin, TX, USA, pp 142-147.

Harris C.A. and Meech J.A., (1987). "Fuzzy Logic : a Potential Control Technique for Mineral Processing", *CIM Bulletin*, September, pp 51-59.

Herbst J.A., Pate W.T. and Oblad A.E., (1989). "Experiences in the use of Model Expert Control Systems in Autogenous and Semi Autogenous Grinding Circuits", in Mular A.L. and Agar G.E. (Eds.), *Advances in Autogenous and Semiautogenous Grinding Technology*, University of British Columbia, Vancouver, BC, Canada, September, pp 669-686.

Horwitz E. and Heckerman D., (1986). "The Inconsistent use of Certainty in Artificial Intelligent Research", in Kanal L.N. and Lemmer J.F., (Eds.), *Uncertainty in Artificial Intelligence,* Elsevier Science Publ. B.V., North Holland, pp 137-151.

Hunter D., (1986). "Uncertain Reasoning using Maximum Entropy Inference", in Kanal L.N. and Lemmer J.F., (Eds.), *Uncertainty in Artificial Intelligence*, Elsevier Science Publ. B.V., North Holland, pp 203-209.

Isaka S., (1990). "An Optimization Approach for Fuzzy Controller Design", *1990 American Control Conference*, San Diego, CA, USA, May, pp 1485-1492.

Isermann R. and Lachmann K.H., (1985). "Parameter-adaptive Control with Configuration Aids and Supervision Functions", *Automatica*, Vol 21, pp 625-638.

Isik C and Ciliz K, (1988). "A Two-level Neural Network System for Learning Control of Robot Motion", *IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 519-522.

Jerez J., Toro H. and Von Borries G., (1985). "Automatic Control of Semi-Autogenous Grinding at Los Bronces", *IFAC Symposium Automation for Mineral Resource Development*, Qeensland, Australia, pp 275-282.

Johnson-Laird P., (1989). "Mental Models", in Posner M.I. (Ed.), *Foundations of Cognitive Science*, MIT, Ch. 12, pp 469-499.

Jones A.H. , Porter B. and Fripp R.N., (1988). "Qualitative and Quantitative Approaches to the Diagnosis of Plant Faults", *IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 87-92.

Jones A.H., Fripp R.N. and Porter B., (1990a). "Intelligent Knowledge-Based Control of Multivariable Plants", *IEEE Int. Symposium on Intelligent Control*, Philadelphia, PA, USA, September, pp 786-793.

Jones A.H., Porter B., Fripp R.N. and Pallet S., (1990b). "Real-time Expert Systems for Diagnosis and Closed-Loop Control", *IEEE Int. Symposium on Intelligent Control*, Philadelphia, PA, USA, September, pp 1088-1094.

Jovic F., (1992). *Process Control Systems: Principles of Design, Operation and Interfacing*, Chapman & Hall, 2nd edition.

Karetnyk D.G., Grant E. and McGregor D.R., (1988). "A Review of Artificial Intelligence in Feedback Control", *IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 40-45.

Karr Ch., (1991). "Applying Genetics to Fuzzy Logic", *AI Expert*, March, pp 38-43.

Kokar M.M., (1988). "Machine Learning in a Dynamic World", (Ed. Panel discussion), *IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 500-507.

Krog S.R., (1979). "Automatic Control of Autogenous and Semiautogenous Mills", in Digre M. (Ed.), *Proc. of Autogenous Grinding Seminar*, University of Trondheim, Norway, May, pp EL1-EL20.

Kuipers B., (1984). "Commonsense Reasoning about Causality: Deriving Behavior form Structure", *Artificial Intelligence*, Vol 24, No 1-3, December, pp 169-203.

Kuipers B., (1985). "The Limits of Qualitative Simulation", *IJCAI-85*, Los Angeles, CA, USA, August, pp 128-136.

Kuipers B., (1986). "Qualitative Simulation", *Artificial Intelligence*, Vol 29, No 3, September, pp 289-338.

Kuipers B., (1987). "Qualitative Simulation as Causal Explanation", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol SMC-17, No 3, May/June, pp 432-444

Kuipers B., (1989). "Qualitative Reasoning with Causal Models in Diagnosis of Complex Systems", in Lawrence W., Loparo A.K. and Nielsen N.R., (Eds.), *Artificial Intelligence, Simulation and Modeling*, John Wiley & Sons, Inc.

Kyburg H.E. and Smokler H.E., (1964). *Studies in subjective Probability*, John Wiley & Sons, Inc.

Latour P.R., (1976). "The Hidden Benefits from Better Process Control", *Advances in Instrumentation*, Vol 31, Part 1, ISA-76 Annual Conference, pp 528;1-11.

Léa Sombé, (1990). "Reasoning Under Incomplete Information in Artificial Intelligence", *Int. Journal of Intelligent Systems*, Vol 5, No 4, September, Special Issue, pp 324-470.

Lee S. and Kim M.H., (1988). "Learning Expert Systems For Robot Fine Motion Control", *IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 534-544.

Lees F.P. and Sayers B., (1976). "The Behavior of Process Operators under Emergency Conditions", in Sheridan T.B. and Johannsen G., (Eds.), *Monitoring Behavior and Supervisory Control*, Plenum Press, NY, USA, pp 331-341.

Leitch R., (1988). "Qualitative Modeling of Physical Systems for Knowledge Based Control", in Denham M.J. and Laub A.J., (Eds.), *Advanced Computing Concepts and Techniques in Control Engineering*, NATO ASI Series, Springer-Verlag, pp 31-51.

Leitch R., (1990). "A Review of the approaches to the Qualitative Modeling of Complex Systems", in McGee J., Grimble M.J. and Mowforth P., (Eds.), *Knowledge-Based Systems for Industrial Control*, Peter Peregrinus Ltd., London, UK.

Luger G.F. and Stubblefield W.A., (1989). *Artificial Intelligence and the Design of Expert Systems*, The Benjamin/Cummings Publ. Co., Inc.

McDermott K., Cleyle P., Hall M. and Harris C.A., (1994). "Expert System for Control of No 4 Autogenous Mill Circuit at Wabush Mines," *26th Meeting of the Canadian Mineral Processors*, Ottawa, Ontario, Canada, January.

McDermott D. and Doyle J., (1980). "Nonmonotonic Logic I", *Artificial Intelligence*, Vol 3, pp 41-72.

McGregor D.R., Odetayo M.O. and Dasgupta D., (1992). "Adaptive Control of a Dynamic System using Genetic-based Methods", *IEEE Int. Symposium on Intelligent Control*, Glasgow, Scotland, UK, August, pp 521-525.

McManus J., Paul A.R. and Yu F., (1975). "Process Control in the Lornex Grinding Circuit", *CIM Bulletin*, May, pp 146-151.

Miller T.T. and Hewes R.P., (1988). "Real Time Experiments in Neural Network Based Learning Control During High Speed Nonrepetitive Robotic Operations", *IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 513-518

Milne R., (1987a). "Artificial Intelligence for On-Line Diagnosis", *IEE Proceedings*, Vol 134, Pt D, No 4, July, pp 238-244.

Milne R., (1987b). "Strategies for Diagnosis", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol SMC-17, No 3, May/June, pp 333-339.

Minton S., (1988). *Learning Search Control Knowledge. An Explanation-Based Approach*, Kluwer Academic Publishers, Boston, USA.

Minton S., Garbonell J.G., Knoblock C.A., Kuokka D.R., Etzioni O. and Gil Y., (1989). "Explanation-Based Learning: A Problem Solving Perspective", *Artificial Intelligence*, Vol 40, pp 63-118.

Mooney R.J. and Benett S.W., (1986). "A domain Independent Explanation-Based Generalizer", *AAAI-86*, Philadelphia, PA, USA, August, pp 551-555.

Mooney R.J., (1990). *A General Explanation-Based Learning Mechanism and its Application to Narrative Understanding*, Morgan Kaufmann Publishers, Inc., California, USA.

Mular A.L., (1989). "Process Control and Instrumentation in Mineral Processing Plants", in CIM, *Operation and Maintenance in Mineral Processing Plants*, Montreal, Ch 10.

Mular A.L. and Burkert A., (1988). "Automatic Control of Semiautogenous Grinding Circuits", *Report submitted to Brenda Business Development Corp., Brenda Mines Ltd., Kelowna, BC, Canada.*

Neapolitan R.E., (1990). *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, John Wiley & Sons, Inc.

Negoita C.V., (1985). *Expert Systems and Fuzzy Systems*. Menlo Park, CA., Benjamin/Cummings Publ. Co.

Negoita C.V., (1987). *Simulation, Knowledge-Based Computing, and Fuzzy Statistics*, Van Nostrand Reinhold.

North R.C., (1976). *The World that Could Be*, The Portable Stanford, Stanford Alumni Association, Stanford, CA., USA.

(Panel Discussion), (1987). "Intelligent Control: How it is Understood by AI and Systems Scientists, and How to Teach it to the Graduate Students", *IEEE Int. Symposium on Intelligent Control*, Philadelphia, PA, USA, pp 508-529

Pang D., Bigham J. and Mandani E.H., (1987). "Reasoning with Uncertain Information", *IEE Proceedings*, Vol 134, Part D, No 4, July, pp 231-237.

Partridge D. and Paap K.R., (1988). "Learning", Ch 7 in McTear M.F. (Ed.), *Understanding Cognitive Science,* Ellis Horwood Ltd., Chichester, England.

Pearl J., (1988). "Evidential Reasoning Under Uncertainty", in Shrobe H.E., (Ed.), *Exploring AI: Survey talks from the National Conferences on AI*, Morgan Kaufmann Publishers, Inc., CA, USA.

Perry R. and Hall M., (1994). "Selbaie Mines Expert System," *26th Meeting of the Canadian Mineral Processors*, Ottawa, Ontario, Canada, January.

Peters L., Beck K. and Camposano R., (1992). "Fuzzy Logic Controller with Dynamic Rule Set", *IEEE Int. Symposium on Intelligent Control*, Glasgow, Scotland, UK, August, pp 216-219.

Pikaar R.N., (1985). "Man-Machine-Interaction in Process Control", in Willumeit H.P., (Ed.), *Human Decision Making and Manual Control*, Elsevier Science Publ., pp 157-172.

Pipitone F., (1986). "The FIS Electronic Troubleshooting System", *IEEE Computer*, pp 68-75.

Raabe H., (1994). "Mill Tonnage Restrictions", Internal communication, Highland Valley Copper Mine, Logan Lake, B.C., Canada.

Rao M., Jiang T.S. and Tsai J.J.P., (1988). "Integrated Architecture for Intelligent Control", *IEEE Int. Symposium on Intelligent Control*, Arlington, VA, USA, August, pp 81-86..

Rao M. and Ying Y., (1990). "Intelligent Control: A New Decade for Pulp and Paper Processes", *IEEE Int. Symposium on Intelligent Control*, Philadelphia, PA, USA, September, pp 1095-1099

Rao M., (1992). "Integrated System for Intelligent Control", *Lecture Notes in Control and Information Sciences*, Springer-Verlag.

Rasmussen J., (1976). "Outlines of a Hybrid Model of the Process Plant Operator", in Sheridan T.B. and Johannsen G. (Eds.), *Monitoring Behavior and Supervisory Control*, Plenum Press, New York, pp 371-383.

Rasmussen J, (1986). *Information Processing and Human-Machine Interaction. An Approach to Cognitive Engineering*, Elsevier Science Publ. Co.

Ratte A.K., (1979). "Computer Control of No 2 Mill at Island Copper Mine", *CMP Regional Meeting*, Vancouver, BC, November.

Reiter R., (1987). "A Theory of Diagnosis from First Principles", Artificial Intelligence, Vol 32, pp 57-95.

Rogers J.A., (1985). "Optimizing Process and Economic Aims", *Chemical Engineering*, December, pp 95-99.

Sanchez I., Cifuentes E.I. and Sbarbaro D., (1988). "Selftuning controller operating under real conditions", *III Latin-American Congress on Automation, VIII Chilean Automatic Control Conference*, Viña del Mar, Chile, October.

Saridis G.N., (1977). *Self-Organizing Control of Stochastic Systems*, Marcel Dekker, NY.

Saridis G.N., (1985). "Foundations of the theory of Intelligent Controls", *IEEE Int. Symposium on Intelligent Control*, Troy, New York, USA, August, pp 23-28.

Saridis G.N., (1987). "Knowledge Implementation: Structures of Intelligent Control Systems", *IEEE Int. Symposium on Intelligent Control*, Philadelphia, PA, USA, pp 9-17.

Scarl E.A., Jamieson J.R. and Delaune C.I., (1987). "Diagnosis and Sensor Validation through Knowledge of Structure and Function", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol SMC-17, No 3, May/June, pp 360-368.

Shafer G., (1976). *A Mathematical Theory of Evidence*, Princeton, University Press.

Sharkey N.E., (1988). "Neural Network Learning Techniques", Ch 8 in McTear M.F. (Ed.), *Understanding Cognitive Science,* Ellis Horwood Ltd., Chichester, England.

Shavlik J.W., (1990). *Extending Explanation-Based Learning by Generalizing the Structure of Explanations*, Morgan Kaufmann Publishers, Inc., California, USA.

Sheridan T.B, (1976). "Toward a General Model of Supervisory Control", in Sheridan T.B. and Johannsen G., (Eds.), *Monitoring Behavior and Supervisory Control*, Plenum Press, New York, USA, pp 271-281.

Shortliffe H.E., (1976). *Computer-based medical consultations: Mycin*, American Elsevier, NY.

Simmons R., (1986). "Commonsense Arithmetic Reasoning", *AAAI-86*, Philadelphia, PA, USA, August, pp 118-124.

Singleton W.T., (1976). "The Model Supervisor Dilemma", in Sheridan T.B. and Johannsen G., (Eds.), *Monitoring Behavior and Supervisory Control*, Plenum Press, NY, pp 261-270.

Stallings, W., (1977). "Fuzzy Set Theory versus Bayesian Statistics", *IEEE Trans. On Systems, Man, and Cybernetics*, March, pp 216-219.

Stark P.A., (1970). *IIntroduction to Numerical Methods.* MacMillan Publishing Co., NY

Sugeno M., (1985). *Industrial Applications of Fuzzy Control*, (Ed.), Elsevier Science Publishers.

Sutton R., (1990). *Modeling Human Operators in Control System Design*, Research Studies Press Ltd.

Thompson T.F. and Clancey W.J., (1986). "A Qualitative Modeling Shell for Process Diagnosis", *IEEE Software*, March, pp 6-15.

Tong R.M., (1977). "A Control Engineering Review of Fuzzy Systems", *Automatica*, Vol 13, pp 559-569.

Tuffs P.S. and Clarke D.W., (1985). "Self-Tuning Control of offset: A Unified Approach", *IEE Proceedings*, Vol 32, Pt D, No 3, May, pp 100-110.

Vagenas N. and Grangholm S., (1991). "Developing and applying expert systems for fault diagnosis of mining equipment in Swedish mines", In Poulin R. et al. (Eds.), *Proc. of the 2nd Canadian Conference on Computer Applications in the Mineral Industry*, Vancouver, BC, Canada, September, pp 191-201.

van Delf J.H., (1985). "Cognitive Aspects of Human Monitoring Behavior", in Willumeit H.P., (Ed.), *Human Decision Making and Manual Control*, Elsevier Science Publ., pp 319-328.

Verbruggen H.B. and Åström K.J., (1990). "Artificial Intelligence and Feedback Control", in Rodd M.G., Li H. and Su S., (Eds.), *Artificial Intelligence in Real-Time Control 1989*, Proceedings of the IFAC Workshop, Shenyang, China, September, Pergamon Press, pp 1-11.

Weatherford R., (1982). *Philosophical Foundations of Probability Theory*, Routledge & Kegan Paul.

Weiss S.M., Kulikowski C.A., Amarel S. and Safir A., (1979). "A Model-Based Method for Computer-Aided Medical Decision-Making", *Artificial Intelligence*, Vol 11, pp 145-172.

Widman L., (1989). "Semi-Quantitative Close-Enough Systems Dynamics Models: An Alternative to Qualitative Simulation", in Lawrence W., Loparo A.K. and Nielsen N.R., (Eds.), *Artificial Intelligence, Simulation and Modeling*, John Wiley & Sons, Inc.

Williams B, (1984) "Qualitative Analysis of MOS Circuits", *Artificial Intelligence*, Vol 24, No 1-3, December, pp 281-346.

Wittenmark B. and Åström K.J., (1984). "Practical Issues on the Implementation of Self-tuning Control", *Automatica*, Vol 20, pp 595-605.

Zadeh L.A., (1973). "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes", *IEEE Trans. Systems, Man and Cybernetics*, Vol SMC-3, pp 28-44.

Zadeh L.A., (1986). "Is Probability Theory Sufficient for Dealing with Uncertainty in AI: A Negative View", in Kanal L.N. and Lemmer J.F. (Eds.), *Uncertainty in Artificial Intelligence*, Elsevier Science Publ. B.V., North Holland, pp 103-116.

Zadeh L.A., (1988). "Fuzzy Logic", *IEEE Computer*, April, pp 83-93.

APPENDIX A AI ELEMENTS THAT CAN BE INCORPORATED INTO AN ISCS

A.1 A Subjective Probability Approach to Handle Uncertainty

A.1.1 Alternatives to Reasoning Under Uncertainty

The approaches followed by artificial intelligence (AI) researchers to tackle this problem can be classified into three groups: logicist, neo-calculist, and neo-probabilist (Pearl 1988). The logicist approach attempts to deal with uncertainty using non-numerical techniques such as non-monotonic logic (McDermott and Doyle 1980), and the theory of endorsements (Cohen 1985). The neo-calculist approach uses numerical representations of uncertainty and believing somewhat that probability calculus is inadequate for the task, invents an entirely new calculus such as evidence theory (Shafer 1976), fuzzy logic (Zadeh 1973 & 1988; Negoita 1985 & 1987; Tong 1977), and certainty factors (Shortliffe 1976). The neo-probabilist approach remains within the framework of probability theory, while attempting to strengthen its interpretation to deal with AI tasks (Cheeseman 1986; Weatherford 1982; Hunter 1986).

There has been much controversy regarding which one of these approaches is most adequate or most appropriate to deal with reasoning under uncertainty. This discussion has been especially hot between fuzzy logic and probabilistic approach supporters (Cheeseman 1985 & 1986; Zadeh 1986; Stallings 1977; Hunter 1986). Cheeseman (1985) concludes his 'defense of probability', for instance, by saying that, "an AI system for reasoning under uncertainty should be possible based only on the basic laws of probability ... No other representation or calculus is necessary for reasoning under uncertainty." Zadeh (1986) argues that, "viewed as a language, classical probability theory is insufficiently expressive to cope with the multiplicity of kinds of uncertainty which one encounters in AI and, more particularly, in expert systems."

Other researchers, however, such as Léa Sombé group (1990), Pang et al. (1987) and Fox (1986), conclude that it seems pointless to think that a single formalism would be capable of representing uncertainty in commonsense reasoning. Fox, in particular, recommends that we should understand the different alternative approaches and represent their assumptions, advantages and weaknesses

explicitly, so that we can look at them not as rivals but as real alternatives to be used as circumstances demand.

In line with Fox's conclusions, this study focuses on the development of a "subjective" probability approach that seems suitable to handle uncertainty in intelligent control (IC) problems. The question of which method—probability or other—is the "best" or the "right" method to deal with uncertainty is not considered as an issue.

A.1.2 A Subjectivistic View of Probability

The subjectivistic view of probability considers probability as "the actual degree of belief in a given proposition held by a person at some specific time on the basis of given evidence" (Weatherford 1982; Kyburg and Smokler 1964, Chap. 1).

According to this view, probability represents a relation between a statement and a body of evidence. The numerical value attached to it represents nothing more than a "degree of belief." This value is not uniquely determined; a given statement may have any probability between 0 and 1, on a given evidence, according to the inclination of the person whose degree of belief that probability represents.

Subjective probability allows us to talk about probabilities of single events. If we need to know, for instance, the probability of "overloading the SAG mill during the next shift," we just ask the mill foreman; and his degree of belief is the probability value for this proposition. Possibly, his answer will be different from that of the mill operator. Subjectivists recognize that a person's opinion is the final authority, even though there can be as many probabilities as there are opinions. Classical probability theory, on the other hand, would lead us to the embarrassing conclusion that there is no probability for this proposition of overloading the mill.

A.1.3 Fuzzy Concepts Representation

In this section we will see how to represent and interpret fuzzy concepts such as "high," "low," "hard" or "coarse," under a subjective probability approach. Thus, suppose we consider a grinding circuit where the topic of interest is the hardness of the ore being treated. Asked for a description of the hardness of the ore, an operator may respond that "the ore is HARD." What does he really mean by "hard"? We have to recognize the inherent vagueness of concepts such as "hard," and the need to

specify the context in which they are being used. The result would be completely different if instead of ore we were talking about boiled eggs. After questioning the operator for more detail, we may be able to understand and represent his concept of "hard" when referring to the ore. In general, the representation of "hard" may take any form as long as it represents the coherent degree of belief of someone in the same situation analyzed. A possible representation of this concept is shown in Figure A.1

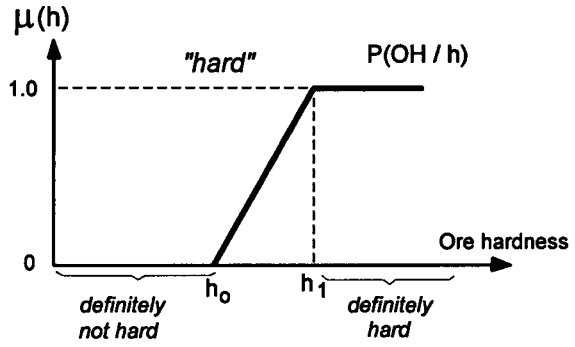


Figure A.1 Fuzzy concept "hard," P(OH/h).

Under probability formalisms, the proposition S1: "the ore is HARD," given by an operator, can be represented as:

S1: P(OH/h) = db(h) (A.1)

where: **h** is the specific value of the ore hardness; OH, "the ore can be referred to as HARD;" and db(h), a degree of belief (DoB) that may take any value between 0 and 1. A DoB equal to 1 would indicate that the ore is "definitely hard", while a DoB equal to 0 would indicate that the ore is "definitely not high."

A.1.4 Reasoning Under Uncertainty

We will see now how to incorporate fuzzy concepts, when represented and interpreted as indicated above, into a rule-based reasoning process. Let us consider the same scenario as in the previous section, where an operator is supervising the operation of a grinding mill. Let us assume that a piece of his heuristic knowledge can be represented by the rule:

R1: IF ***power_draw*** is "high"
THEN **the ore** is "hard"

where both the concept "high" for power draw and the concept "hard" for the ore hardness are fuzzy concepts. Let us assume that the operator's concept "high" for the power draw, P(JH/j), is as shown in Figure A.2; and that the concept "hard" for the ore hardness, P(OH/h), is the same as shown in Figure A.1.

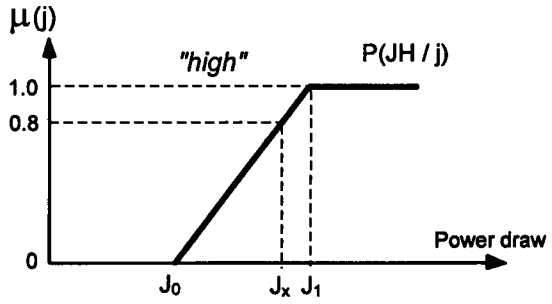


Figure A.2 Concept "high" for the power draw.

An approach to incorporate probability in a reasoning process, based on a method known as "causal belief networks" by Neapolitan (1990), is proposed. Under this approach, rule R1 can be written as:

$$P(OH/KJ) = P(OH/JH)\, P^*(JH/KJ) + P(OH)\,[\,1 - P^*(JH/KJ)\,] \quad ;P(JH/KJ) \geq 0.5 \qquad (A.2)$$

where $P^*(JH/KJ) = 2\,[P(JH/KJ) - 0.5]$

OH: proposition "the ore is hard"

KJ: information on the power draw - evidence to decide if the power draw is high or not

JH: proposition "the power draw is high"

P(OH): unbiased degree of belief in OH before knowing KJ.

Equation (A.2)gives the updated degree of belief in the conclusion of rule R1 when this rule is fired. To solve equation (A.2) we need to have an expert's opinion about P(OH/JH) in the knowledge base, and a value for P(JH/KJ) provided by the operator when rule R1 is instantiated.

A.1.5 Combining Evidence

When a rule has two or more antecedents we need a mechanism to combine these antecedents to determine the resulting degree of belief in the conclusion. To see how this can be done let us consider a rule with two antecedents of the form:

R2: IF A_1 AND A_2
THEN C

Let us assume that the antecedents $\mathbf{A_1}$ and $\mathbf{A_2}$ are known with some uncertainty, i.e. the information is actually given for $P(A_1/e_1)$ and $P(A_2/e_2)$, where $\mathbf{e_1}$ and $\mathbf{e_2}$ are the evidence used to infer $\mathbf{A_1}$ and $\mathbf{A_2}$ respectively; and that **C** is independent of the evidence $\mathbf{e_i}$, given that $\mathbf{A_i}$ or $\neg\mathbf{A_i}$ is known with certainty. Then R2 can be represented by the inference network shown in Figure A.3.

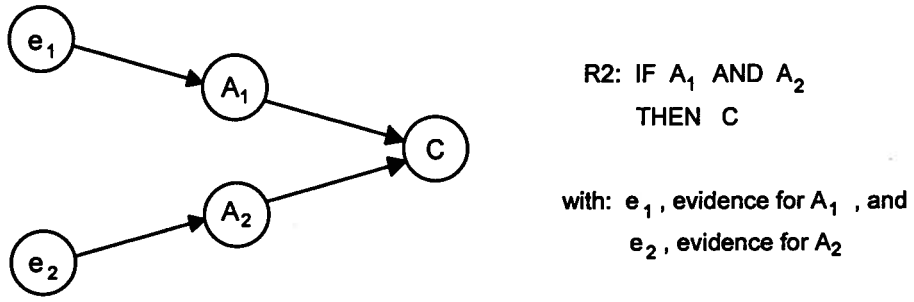


Figure A.3 Inference network for rule R2.

The degree of belief in the conclusion given the evidence $\mathbf{e_1}$ and $\mathbf{e_2}$, $P(C/e_1 \wedge e_2)$, can be obtained with Bayes' theorem as:

$$P(C/e_1 \wedge e_2) = \frac{P(e_1 \wedge e_2 / C)}{P(e_1 \wedge e_2)} P(C) \qquad (A.3)$$

In this situation with more than one antecedent to draw a conclusion we would like to be able to obtain independently $P(C/e_1)$ and $P(C/e_2)$, and then through a mathematical operation combine them to obtain the desired value of $P(C/e_1 \wedge e_2)$. However, as shown by Neapolitan (1990, Chap. 4), this is impossible without proper probabilistic assumptions.

Case A. Conditional Independence of the Evidence

$P(C/e_1 \wedge e_2)$ can be obtained in terms of $P(C/e_1)$ and $P(C/e_2)$ if we assume that $\mathbf{e_1}$ and $\mathbf{e_2}$ are conditionally independent, i.e. if $\mathbf{e_1}$ and $\mathbf{e_2}$ are independent when **C** occurs and when **C** does not occur. $P(C/e_1 \wedge e_2)$ can be obtained as follows:

i) Obtain $P(C/e_1)$ and $P(C/e_2)$ from

$$P(C/e_i) = P(C/A_i)\, P^*(A_i/e_i) + P(C)\, [\, 1 - P^*(A_i/e_i)] \qquad ;P(A_i/e_i) \geq 0.5 \qquad (A.4)$$

where $P^*(A_i/e_i) = 2\, [P(A_i/e_i) - 0.5]$

$P(C/A_i)$ and $P(C/\neg A_i)$ are given by an expert, and $P(A_i/e_i)$ are the values supplied at the time of instantiating the associated rule.

ii) Calculate the odds $O(C/e_1 \wedge e_2)$ from

$$O(C/e_1 \wedge e_2) = \lambda_{e1}\, \lambda_{e2}\, O(C) \qquad (A.5)$$

where $\lambda_{ei} = O(C/e_i) / O(C)$, $\quad O(C/e_i) = \dfrac{P(C/e_i)}{1 - P(C/e_i)}$ and $\quad O(C) = \dfrac{P(C)}{1 - P(C)}$

iii) Calculate the degree of belief in the conclusion of rule R2 given the evidence $\mathbf{e_1}$ and $\mathbf{e_2}$, $P(C/e_1 \wedge e_2)$, from

$$P(C/e_1 \wedge e_2) = \frac{O(C/e_1 \wedge e_2)}{1 + O(C/e_1 \wedge e_2)} \qquad (A.6)$$

From equation (A.5) we can see how to project this updating scheme for any number of pieces of evidence, provided the conditional independence assumption applies. Equation (A.5) is modular, so all we need is P(C), the initial degree of belief in the conclusion to determine O(C); then evaluate each $P(C/e_i)$ and $O(C/e_i)$ as evidence arrives; then use equation (A.5) to determine $O(C/e_1 \ldots \wedge e_i)$; and finally use equation (A.6) to update the degree of belief in the conclusion, $P(C/e_1 \ldots \wedge e_i)$.

Case B. Evidence is not conditionally independent

If we cannot assume that the pieces of evidence are conditionally independent, the updated degree of belief in the conclusion of rule R2 can be obtained from:

$$\begin{aligned} P(C/e_1 \wedge e_2) = \quad & P(C/A_1 \wedge A_2)\, P(A1/e1)\, P(A2/e2) + \\ & P(C/A1 \wedge \neg A2)\, P(A1/e1)\, [1 - P(A2/e2)] + \\ & P(C/\neg A1 \wedge A2)\, [1 - P(A_1/e_1)]\, P(A_2/e_2) + \\ & P(C/\neg A_1 \wedge \neg A_2)\, [1 - P(A_1/e_1)]\, [1 - P(A_2/e_2)] \end{aligned} \qquad (A.7)$$

An expert needs to provide the values for $P(C / A_1 \wedge A_2)$, $P(C / A_1 \wedge \neg A_2)$, $P(C / \neg A_1 \wedge A_2)$ and $P(C / \neg A_1 \wedge \neg A_2)$, which may not always be easy to estimate. This becomes a real problem as the number of pieces of evidence involved increases - the number of terms the expert would need to provide also grows exponentially. As in Case A, $P(A_1/e_1)$ and $P(A_2/e_2)$ are the only values that need to be provided when R2 is instantiated.

Another problem with equation (A.7) is that when there is no information about A_1 and A_2, i.e. $P(A_1/e_1) = P(A_2/e_2) = 0.5$, $P(C/e_1 \wedge e_2)$ takes the value

$$P(C/e_u) = [\, P(C/A_1 \wedge A_2) + P(C/A_1 \wedge \neg A_2) + P(C/\neg A_1 \wedge A_2) + P(C/\neg A_1 \wedge \neg A_2) \,] /4 \qquad (A.8)$$

which may not necessarily correspond to P(C), the previous degree of belief in the conclusion **C**. In the same way, we may expect that when $P(A_1/e_1)$ is provided but not $P(A_2/e_2)$; i.e. $P(A_2/e_2) = 0.5$, $P(C / e_1 \wedge e_2)$ would correspond to $P(C/e_1)$, the degree of belief in the conclusion given the evidence $\mathbf{e_1}$ alone. However, from equation (A.7) we obtain

$$\begin{aligned} P(C/e_1 \wedge e_u) = \quad & \tfrac{1}{2} [\, P(C/A_1 \wedge A_2) + P(C/A_1 \wedge \neg A_2)] \, P(A_1/e_1) + \\ & \tfrac{1}{2} [\, P(C/\neg A_1 \wedge A_2) + P(C/\neg A_1 \wedge \neg A_2)] \, [1 - P(A_1/e_1)] \end{aligned} \qquad (A.9)$$

which may not always correspond to $P(C/e_1)$. The same situation happens when $P(A_2/e_2)$ is known and $P(A_1/e_1)$ is 0.5.

This can be solved by splitting equation (A.7) into four equations one for each region shown in Figure A.4. In this case we have more independence on the values assigned to previous beliefs and updated ones. However, this increase of freedom has the disadvantage that the number of values the expert has to assign increases even more — so, perhaps it is not worth trying it. This problem is even more obvious when the number of antecedents is larger than two.

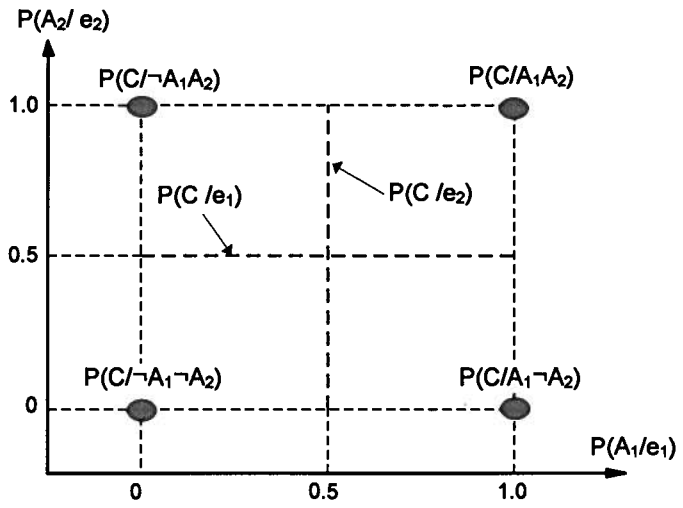


Figure A.4 Combining pieces of evidence when they are not conditionally independent.

A.1.6 Potential and Limitations of the Approach Proposed

In general, we can say that the subjective probability approach proposed in this research is suitable to represent and handle fuzzy concepts such as those found in human reasoning, when dealing with the supervision of a process. However, as for any other single method to deal with uncertainties in reasoning, it is not a *panacea* but an approach with its own inherent limitations.

The most interesting and obvious feature of this approach is that it uses probability calculus to handle concepts. According to a study by Horwitz and Heckerman (1986), probability gives a proper distinction between absolute belief and belief updates, when combining evidence to update a degree of belief in a conclusion. Another interesting aspect of the subjective approach proposed is the separation between the degree of belief in a proposition and in its negation.

One basic conclusion valid for each of the techniques proposed to deal with uncertainty is that none of them can equal the richness and flexibility of human reasoning in dealing with these kinds of problems. Probability, for example, is a widely respected tool for dealing with uncertainty. We showed in this research how a subjective probability approach can be used to incorporate uncertainty handling in the context of an ISCS. However, we cannot expect to represent under this single formalism every type of uncertainty problems that an ISCS has to deal with.

An approach supported by researchers is to integrate different techniques under the coordination of a "meta level" of knowledge about uncertainty. This meta level would decide which technique or group of techniques are more appropriate to deal with a particular problem.

A.2 Learning Capacity

Learning is still in an early stage of development, far too immature to let one of these techniques take full control of an ISCS and intervene with the operation of a process. A much more reasonable alternative is proposed in this study: the incorporation of learning into an ISCS to enhance the user interface. The system could start operating with a user interface that handles a basic set of linguistic concepts. As a result of a learning process, the system would slowly acquire new concepts organizing them into a hierarchical structure to represent more complex concepts. Ultimately, a user interface rich in high-level concepts gives the user the opportunity to establish a more "intelligent" communication with the system.

A.2.1 Alternative Techniques to Learning

There are several learning techniques proposed by AI researchers. Among them we can distinguish empirical and analytical learning techniques. Within empirical learning techniques we can mention symbolic induction learning, evolutionary (genetic) learning, and connectionist learning.. Analytical learning techniques are knowledge-intensive and require background knowledge; the basic inference type they use is deduction. The knowledge learned is basically a new representation for the input information; and not necessarily new knowledge (Forsyth 1989; Anderson and Hinton 1981; Partridge and Paap 1988; Sharkey 1988).

A characteristic of most learning techniques is that they follow a process of gradual learning. They do not take sufficient advantage of domain knowledge; they learn by abstracting common properties from a large number of positive and possibly negative instances of some concept (Forsyth and Rada 1986, Forsyth 1989). Some of these techniques, such as evolutionary learning and connectionist learning require hundreds of learning cycles before something useful is learned. However, when dealing with learning in the context of a real-time ISCS, there is not enough time available or enough cases of some sort to apply one of these techniques that require a large number of instances to learn a concept.

A.2.2 Explanation-Based Learning (EBL) Techniques

Several researchers have been working on learning techniques referred to as Explanation-Based Learning (EBL) techniques (DeJong 1988; Minton 1988; Minton et al. 1989; Mooney 1990; Mooney and Benett 1986; Shavlik 1990). These learning techniques overcome the barrier of requiring a large number of instances to learn a concept; researchers claim that learning can be achieved with even only one example. These features make EBL techniques the recommended choice for the incorporation of learning into the design of an ISCS.

In a standard EBL technique, the system starts with a "target" concept definition, knowledge about the domain of the concept to be learned and with a general criterion on how the knowledge should be represented. Then the system receives a "training" example from which it has to derive a more appropriate representation (explanation) of such concept. EBL techniques use the explanation of a very few examples (usually just one) to define the boundaries of a concept. The concept's definition is determined by a guided inspection using domain-knowledge on why an example worked, and not by similarities and differences between positive and negative examples, as in the case of induction techniques. The explanation identifies the relevant features of the example, which constitute sufficient conditions for describing the concept. The task of the system is to generalize such explanations so they are applicable to a range of similar situations. This allows explanations to be reused, hence improving the performance of the system.

A.2.3 An EBL Approach to Learn Concepts

Let us consider an example to describe in more detail EBL techniques. Let us consider that we are supervising the operation of a control loop, and that we want to teach the system how to recognize when "the control loop is operating OK." To make things easy, let us assume that the only aspect of the operation of the loop we are interested in, is the evolution of the controlled variable. If the controlled variable follows its setpoint, we would say that "the control loop is operating OK." We are not interested in a proper tuning of the controller or in any other aspect of the operation of the loop.

We said that EBL techniques use domain-knowledge to learn a concept. Thus the first thing we need is a domain theory concerning the aspects of the operation of the loop we are interested in. Using rule-based representation, we may write the domain theory for this example as shown in Figure A.5.

The limited theory shown in Figure A.5 defines the concepts **agrees-with, is-close-to** and **is-approaching-to,** when dealing with the behavior of two variables. For example, a variable var_1 **is-close-to** a variable var_2 (see rule R3 in Figure A.5), if the difference in their amounts is less than a pre-specified value. Nothing is said about the dynamics of these variables.

R1:	IF	$?var_1$	agrees-with	$?var_2$
	OR	$?var_1$	is-close-to	$?var_2$
	OR	$?var_1$	is-approaching-to	$?var_2$
	THEN	$?var_1$	follows	$?var_2$
R2:	IF	amount($?var_1$)	is-equal-to	amount($?var_2$)
	AND	dynamics($?var_1$)	is-equal-to	dynamics($?var_2$)
	THEN	$?var_1$	agrees-with	$?var_2$
R3:	IF	difference($?var_1$, $?var_2$)	is-less-than	diff-limit
	WITH	difference($?var_1$, $?var_2$)= amount($?var_1$) - amount($?var_2$)		
	THEN	$?var_1$	is-close-to	$?var_2$
R4:	IF	amount($?var_1$)	is-less-than	amount($?var_2$)
	AND	dynamics($?var_2$)	is	CNT
	AND	dynamics($?var_1$)	is	INC
	THEN	$?var_1$	is-approaching-to	$?var_2$
R5:	IF	amount($?var_1$)	is-greater-than	amount($?var_2$)
	AND	dynamics($?var_2$)	is	CNT
	AND	dynamics($?var_1$)	is	DEC
	THEN	$?var_1$	is-approaching-to	$?var_2$

where: $?var_i$: an instance of a process variable.
diff-limit: pre-specified value
INC/CNT/DEC: increasing/constant/decreasing

Figure A.5 Domain theory to compare values of process variables.

We also require a functional specification of the desired concept to be learned. In this case, this will be given by Q1 as follows:

Q1:	IF	?loop	**is-a**	control-loop
	AND	controlled-variable(?loop)	**follows**	setpoint(?loop)
	THEN	?loop	**is-operating**	OK

This functional representation of the concept can be interpreted as the "depth" at which the user wants to interact with the system. In other words, when the user interacts with the system and asks for an explanation why the system responds that a loop "is operating OK," the system should give an

explanation in terms of the corresponding controlled variable and setpoint of the loop, and use the concept **follows**, as in rule Q1.

Now we have to give the system an example of a case where a loop is operating properly (OK). This example is given in Figure A.6.

LIC-10 is-a CONTROL-LOOP
LIC-10 is-controlling level-of-fluid in TANK-10
level-10 is-controlled-variable-of LIC-10
sp-10 is-setpoint-of LIC-10
amount(*sp-10*) has-value POS-MEDIUM
dynamics(*sp-10*) has-value CNT
amount(*level-10*) has-value POS-SMALL
dynamics(*level-10*) has-value INC

where: POS: positive
INC/CNT: increasing/constant

Figure A.6 Example of a control loop operating OK.

The system then builds a proof-tree using the domain knowledge available. This proof tree is shown in Figure A.7.

From this proof tree the system obtains a generalized explanation for the example that can be used later in a similar situation. This generalization includes the elimination of irrelevant features such as the fact that LIC-10 is controlling the level of fluid in tank TANK-10, for instance. It also includes identity elimination, i.e. it removes unnecessary dependence on particular objects. The fact that *level-10* is the controlled variable of the loop is not important; what it is important is that this piece of knowledge is applicable to a controlled variable of a control loop. In the same way, there is nothing specific for the loop LIC-10; this example would work as well for any other loop - controlling level or any other variable. The generalized explanation for this example is shown in Figure A.8.

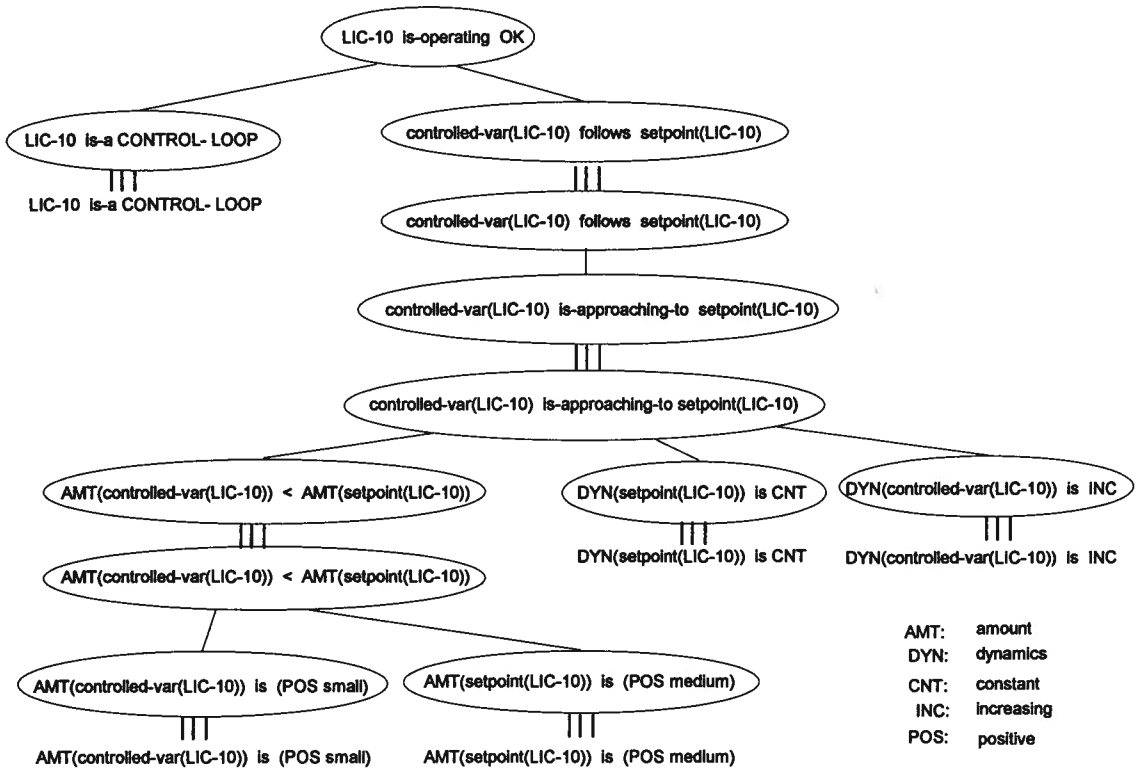


Figure A.7 Proof tree for the example of Figure A.6.

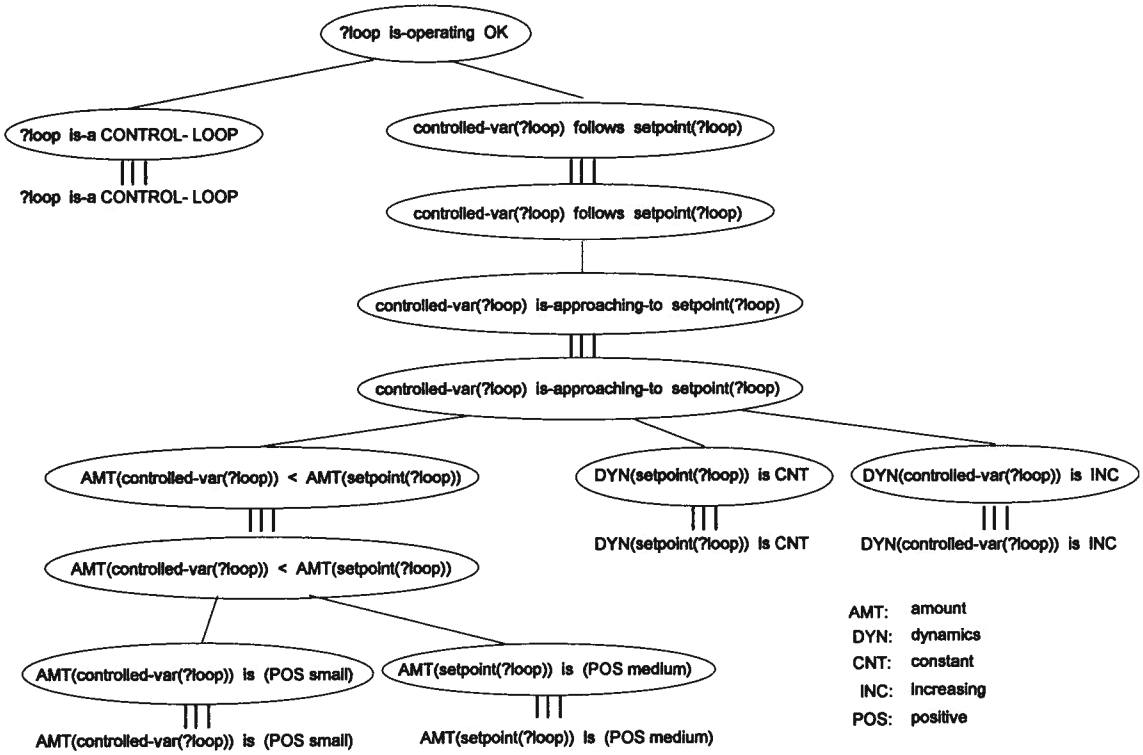


Figure A.8 Generalized explanation for the control loop case.

This generalized explanation is then used to obtain macro-rules which are more efficient representations of the concept or sub-concepts being taught to the system. A non-restricted macro-rule for this example is given schematically in Figure A.9 . The system will use this macro-rule when asked to assess the operation of a loop. And when asked for an explanation of why a specific loop is operating OK, the explanation will be in terms of the antecedents of this macro-rule - the other sub-goals have been eliminated.

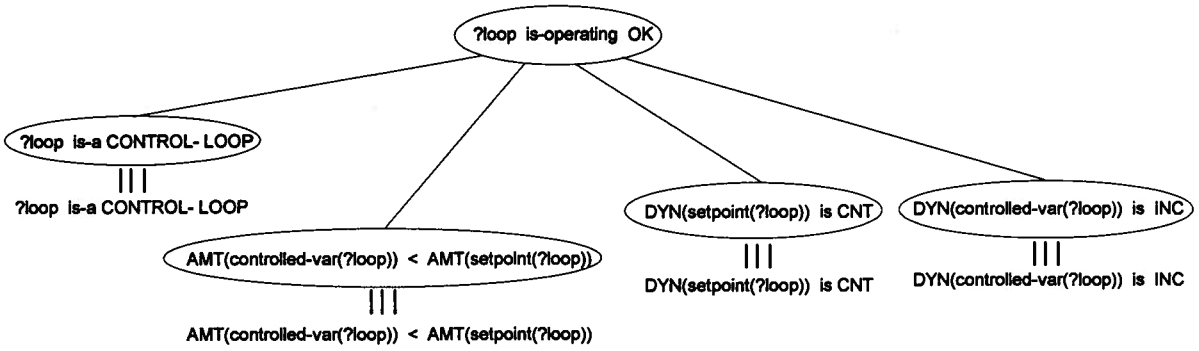


Figure A.9 Unrestricted generalization for the case of the control loop.

If we now consider the restrictions imposed by the functional representation given by rule Q1, we cannot eliminate the sub-concept **follows**. The system has to use the concept **follows** when providing an explanation about the behavior of the loop. Thus **follows** can be represented via a macro-rule derived from the generalized explanation of Figure A.8 and added to the unrestricted macro-rule of Figure A.9. Thus the new direct components of the concept the "loop is operating OK" is shown in Figure A.10.

From Figure A.9 and Figure A.10 we can see how it is possible for the user to control the "depth" at which we want to interact with the system. If the user chooses to interact with the system in the terms given by rule Q1, for instance, he cannot work with the unrestricted macro-rule shown in Figure A.9 because it does not have the sub-concept **follows**. If the user does not need the branches under this sub-concept, he can replace the sub-concept **follows** by an operational macro-rule with the results shown in Figure A.10. If the main concept is going to be part of a larger concept hierarchy, and none of the sub-concepts involved are required, the unrestricted operational representation of Figure A.9 will suffice.

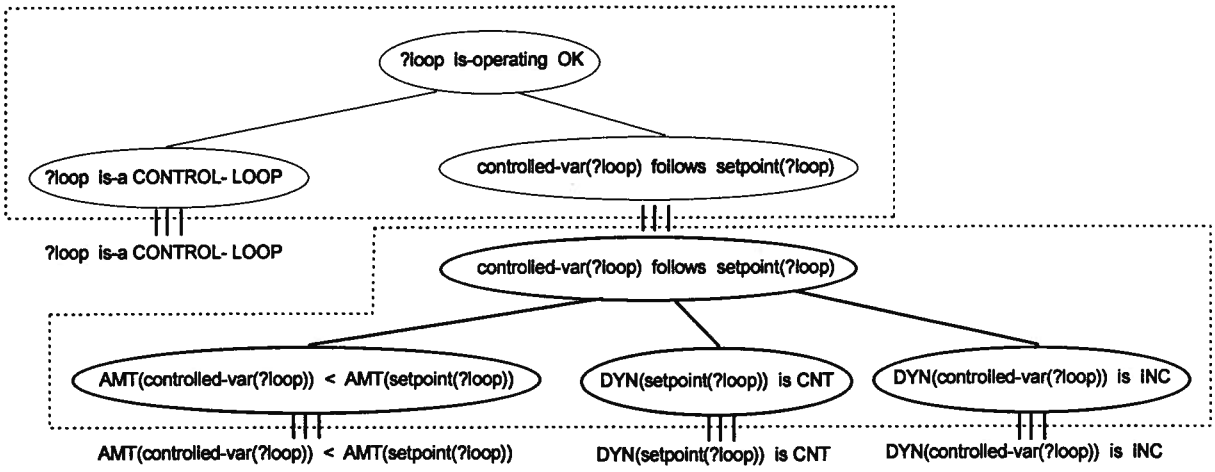


Figure A.10 Restricted generalization for the case of the control loop

The general idea of partial operational representation is schematically shown in Figure A.11. Depending on the level of abstraction specified by the user, some of the sub-concepts within the hierarchy are represented by operational macro-rules while other remain with a full representation. All the sub-concepts beneath the main concept are lost but with respect to this main concept hierarchy only; they remain in the knowledge base and can be incorporated back at any time.

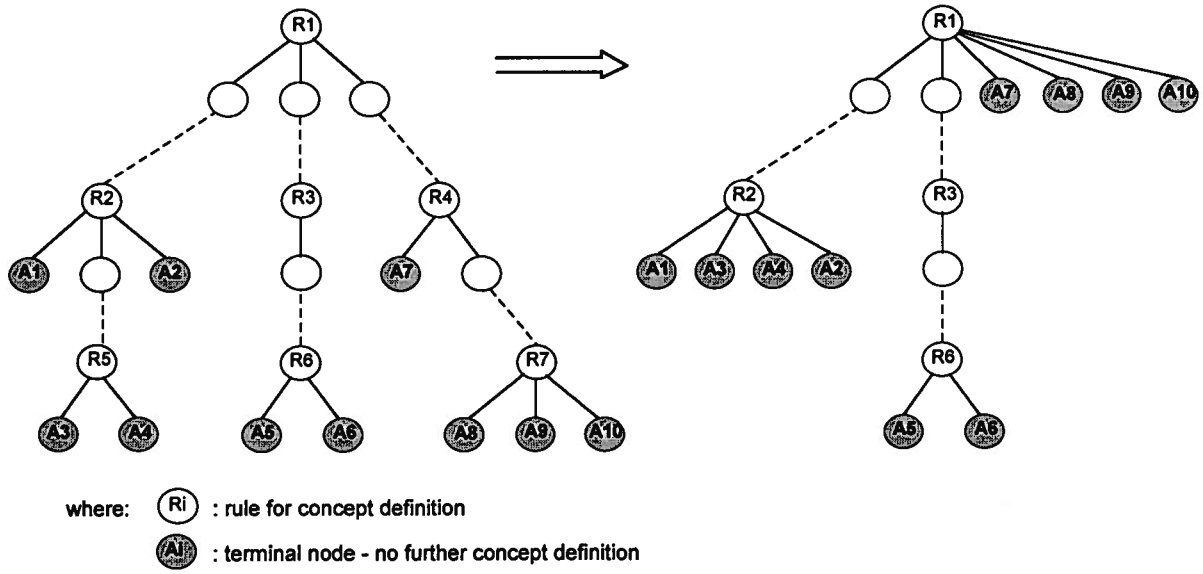


Figure A.11 User defined concept hierarchy

A.2.4 Potential and Limitations of the Approach Proposed

Two major features make this proposed learning approach a feasible and an attractive learning alternative in the design of an ISCS. The first feature is that this learning approach involves only the interaction with the user so the chances of spurious learning harming the process being supervised are minimized. The second feature is that the learning process does not require a high volume of data; in many cases learning can be obtained with only one instance of the learning goal.

An ISCS could start with a default set of concepts for the user interface. Then its learning capacity would allow it to learn new concepts and thus accommodate the user interface to suit the requirements of each individual user.

Another important feature of the learning approach is that it provides a flexible hierarchical structure for the concepts the system handles. Thus the user can decide the level of abstraction at which he wants to interact with the system.

Does the system learn concepts? Does the system "understand" a new concept and the instance when this concept can be used? Some researchers may respond that the system does "learns" concepts; however, others argue that what we call learning is no more than a different manipulation of symbolic information by the system. Despite limitation of current techniques, we should continue researching towards the goal of real "artificial learning."

APPENDIX B PROCEDURE TO BUILDING A QM INTO THE ISCS

i) Building a Steady State QM

The following steps are required for a developer to build a steady state qualitative model in ProcessVision.

Step 1: Edit class ***qm_function***. Add an object ***mss_name***, where ***name*** is the name of the QM to be created.

Step 2: Create a procedure with the name chosen in Step 1, ***mss_name***. This procedure defines the QM in terms of input/output variables of the form: *variable.input.@float* , *variable.output.@float*.

Step 3: If ***mss_name*** is a new model created in Step 2, and it needs to run every sampling instant, add the following line to the procedure ***run_model_ss***.

RUN_PROCEDURE ("***mss_name***")

The relative position of this line within the procedure is important: this line should be after the lines invoking models that provide information required by ***mss_name***.

If the QM to be created uses the ***qmss_table(3)*** primitive, follow the steps indicated below.

Step a.1: Same as Step 1 described above.

Step a.2: Create a procedure ***mss_name_definition*** with the following contents:

mss_name.input_l.@float = in_l
mss_name.output_l.@float = out_l
mss_name.input_m.@float = in_m
mss_name.output_m.@float = out_m
mss_name.input_h.@float = in_h
mss_name.output_h.@float = out_h

where (in_i, out_i) are the values that form the low (**l**), medium (**m**), and high (**h**) pairs that define the steady state QM. ***name*** is the name given to the model being built.

Step a.3: Create a procedure ***mss_name*** to assign values to the inputs and to read the output of the QM ***qm_table(3)***. This procedure is as follows:

qm_table3.input_l.@float = ***mss_name***.input_l.@float
qm_table3.input_m.@float = ***mss_name***.input_m.@float
qm_table3.input_h.@float = ***mss_name***.input_h.@float
qm_table3.output_l.@float = ***mss_name***.output_l.@float
qm_table3.output_m.@float = ***mss_name***.output_m.@float
qm_table3.output_h.@float = ***mss_name***.output_h.@float
qm_table3.input.@double = ***mss_name***.input1.@double
MACRO ("qmss_table3")
mss_name.output.@double = qmss_table3.output.@double

Step a.4: Same as Step 3 described above.

ii) Building a Dynamic QM

The steps required to build the first order dynamic model that predicts the dynamic behavior of a process are:

Step 1: Edit the class ***model_dyn***. Add the object ***mdyn_name***, where ***name*** is the name of the QM to be created.

Step 2: Create a procedure ***mdyn_name***, with ***name*** chosen in Step 1. This procedure contains the following information:

```
qm_dyn_first.output_ss.@double  = variable.pred_amt_ss.@float
qm_dyn_first.output_old.@double  = variable.pred_amt_old.@float
qm_dyn_first.time_s_state.@float  = variable.time_constant.@float
RUN_PROCEDURE ( "qm_dyn_first" )
variable.pred_amt.@float = qm_dyn_first.output.@double
```

where

variable = name of the output variable of the model.
variable.pred_amt.@float = predicted output value at the current sampling instant.
variable.pred_amt_old.@float = predicted output at the previous sampling instant.
variable.pred_amt_ss.@float = steady state output value predicted by a steady state QM.
variable.time_constant.@float = time constant of the process.
qm_dyn_first = procedure that solves the dynamic QM primitive.

Step 3: Add the following line to the procedure ***run_model_dyn***

RUN_PROCEDURE("***mdyn_name***")

The relative position of this line in the procedure is important. This model may require information provided by another dynamic model, and at the same time it may provide information to other models.

APPENDIX C KNOWLEDGE BASE FOR THE HEAD TANK

The following is the source code of the knowledge base associated with the simulation of the head tank (see section 5.4), as implemented in ProcessVision. The same structure of the different components provided by ProcessVision is maintained.

```
Class
@name = model_dyn
@object = mdyn_frate_out, mdyn_level1
endClass

Class
@name = model_ss
@object = mss_frate_out, mss_level1, mss_level2, mss_level3
@public = input1.@double, input2.@double, output.@double
endClass

Class
@name = p_var
@object = frate_in, frate_out, level
@public = diff_amt_pred.large, diff_amt_pred.medium, diff_amt_pred.small,
        diff_amt_pred.@float, p_amt.high, p_amt.low,
        p_amt.medium, p_amt.neg, p_amt.pos,
        p_amt.@float, p_amt_old.@float, p_amt_ss.@float,
        p_amt_status.@string, p_dyn.cnt, p_dyn.dec,
        p_dyn.inc, p_dyn.@float, p_dyn_status.@string,
        qmodel_ss.@string, rate_of_change.@float, real_amt.high,
        real_amt.low, real_amt.medium, real_amt.pos,
        real_amt.@float, real_amt_max.@float, real_amt_min.@float,
        real_amt_old.@float, real_amt_status.@string, real_dyn.cnt,
        real_dyn.dec, real_dyn.inc, real_dyn.@float,
        real_dyn_status.@string, std_amt.@float, time_s_state.@float
endClass

Class
@name = qm_table3
@object = mss_level2
@public = input1.@double, input_h.@float, input_l.@float,
        input_m.@float, output.@double, output_h.@float,
        output_l.@float, output_m.@float
endClass

Object
@name = a
@attribute = b.@string, b.@float, c.@string
endObject

Object
@name = analyze
@attribute = simulation.results
endObject

Object
@name = convert
@attribute = var.@string, varaible.@string, variable.@string
endObject
```

```
Object
@name = dummy
@attribute = convert.@string, eval_pred.@double, expo.@double,
        k1.@double, k2.@double, k3.@double,
        k4.@double, model.@string, pred_level.@double,
        procedure.@string, reasoning.@string, rotate.@string,
        std_amt.@float, var.@string
endObject

Object
@name = frate_in
@class = p_var
@attribute = diff_amt_pred.large, diff_amt_pred.medium, diff_amt_pred.small,
        diff_amt_pred.@float, p_amt.high, p_amt.low,
        p_amt.medium, p_amt.neg, p_amt.pos,
        p_amt.@float, p_amt_old.@float, p_amt_ss.@float,
        p_amt_status.@string, p_dyn.cnt, p_dyn.dec,
        p_dyn.inc, p_dyn.@float, p_dyn_status.@string,
        qmodel_ss.@string, rate_of_change.@float, real_amt.high,
        real_amt.low, real_amt.medium, real_amt.pos,
        real_amt.@float, real_amt_max.@float, real_amt_min.@float,
        real_amt_old.@float, real_amt_status.@string, real_dyn.cnt,
        real_dyn.dec, real_dyn.inc, real_dyn.@float,
        real_dyn_status.@string, std_amt.@float, time_s_state.@float
endObject

Object
@name = frate_out
@class = p_var
@attribute = diff_amt_pred.large, diff_amt_pred.medium, diff_amt_pred.small,
        diff_amt_pred.@float, p_amt.high, p_amt.low,
        p_amt.medium, p_amt.neg, p_amt.pos,
        p_amt.@float, p_amt_old.@float, p_amt_ss.@float,
        p_amt_status.@string, p_dyn.cnt, p_dyn.dec,
        p_dyn.inc, p_dyn.@float, p_dyn_status.@string,
        qmodel_ss.@string, rate_of_change.@float, real_amt.high,
        real_amt.low, real_amt.medium, real_amt.pos,
        real_amt.@float, real_amt_max.@float, real_amt_min.@float,
        real_amt_old.@float, real_amt_ss.@float, real_amt_status.@string,
        real_dyn.cnt, real_dyn.dec, real_dyn.inc,
        real_dyn.@float, real_dyn_status.@string, std_amt.@float,
        time_s_state.@float
endObject

Object
@name = level
@class = p_var
@attribute = diff_amt_pred.large, diff_amt_pred.medium, diff_amt_pred.small,
        diff_amt_pred.@float, p_amt.high, p_amt.low,
        p_amt.medium, p_amt.neg, p_amt.pos,
        p_amt.@float, p_amt_old.@float, p_amt_ss.@float,
        p_amt_status.@string, p_dyn.cnt, p_dyn.dec,
        p_dyn.inc, p_dyn.@float, p_dyn_status.@string,
        qmodel_ss.@string, rate_of_change.@float, real_amt.high,
        real_amt.low, real_amt.medium, real_amt.pos,
        real_amt.@float, real_amt_max.@float, real_amt_min.@float,
        real_amt_old.@float, real_amt_status.@string, real_dyn.cnt,
        real_dyn.dec, real_dyn.inc, real_dyn.@float,
        real_dyn_status.@string, std_amt.@float, time_s_state.@float
endObject
```

```
Object
@name = mdyn_frate_out
@class = model_dyn
endObject

Object
@name = mdyn_level1
@class = model_dyn
endObject

Object
@name = mss_frate_out
@class = model_ss
@attribute = input1.@double, input2.@double, output.@double,
        parameter.@float
endObject

Object
@name = mss_level1
@class = model_ss
@attribute = dummy1.@double, dummy2.@double, input1.@double,
        input2.@double, output.@double, parameter.@float
endObject

Object
@name = mss_level2
@class = qm_table3, model_ss
@attribute = input1.@double, input2.@double, input_h.@float,
        input_l.@float, input_m.@float, output.@double,
        output_h.@float, output_l.@float, output_m.@float,
        parameter.@float
endObject

Object
@name = mss_level3
@class = model_ss
@attribute = input1.@double, input2.@double, output.@double,
        parameter.@float
endObject

Object
@name = mss_tank_outlet
@attribute = input1.@double, input2.@double, output.@double
endObject

Object
@name = qm_addition
@attribute = input1.@double, input2.@double, output.@double
endObject

Object
@name = qm_difference
@attribute = input1.@double, input2.@double, output.@double
endObject

Object
@name = qm_division
@attribute = input1.@double, input2.@double, output.@double
endObject

Object
```

```
@name = qm_dyn_first
@attribute = dummy.@double, output.@double, output_old.@double,
        output_ss.@double, time_s_state.@float
endObject

Object
@name = qm_multiplication
@attribute = input1.@double, input2.@double, output.@double
endObject

Object
@name = qm_prop_square
@attribute = input.@double, output.@double
endObject

Object
@name = qm_proportional
@attribute = constant.@float, input.@double, output.@double,
        parameter.@float
endObject

Object
@name = qm_table3
@attribute = input.@double, input_h.@float, input_l.@float,
        input_m.@float, output.@double, output_h.@float,
        output_l.@float, output_m.@float, slope.@double
endObject

Object
@name = runx
@attribute = simulation.again
endObject

Object
@name = start
@attribute = simulation.yes
endObject

Object
@name = stop
@attribute = simulation.yes
endObject

Object
@name = system
@attribute = already.configured
endObject

Object
@name = tank
@attribute = area.@float, capacity.@float, height.@float,
        outlet_parameter.@float, outlet_restriction.@float, simulaiton.initialized,
        simulation.initialized
endObject

Object
@name = timex
@attribute = current.@integer, elapsed.@float, elapsed_old.@float,
        sampling.@float, sleep.@float
endObject
```

```
Object
@name = valve
@attribute = opening.@float
endObject

Procedure
@name = QM_simulator_coordinator
@do =
 RUN_PROCEDURE ( "convert_all_measurements" )
 RUN_PROCEDURE ( "run_model_ss" )
 RUN_PROCEDURE ( "run_model_dyn" )
 RUN_PROCEDURE ( "obtain_predicted_dynamics" )
 RUN_PROCEDURE ( "rotate_all_predictions" )
 MACRO ( "qual_status*" )
endProcedure

Procedure
@name = basic_data_processing
@do =
 TEXT ( "basic proc" )
 RUN_PROCEDURE ( "mss_tank_outlet_run" )
endProcedure

Procedure
@name = convert_all_measurements
@do =
 STRCOPY ( dummy.convert.@string, "level" )
 RUN_PROCEDURE ( "convert_real_var" )
 STRCOPY ( dummy.convert.@string, "frate_in" )
 RUN_PROCEDURE ( "convert_real_var" )
 dummy.convert.@string is "frate_out"
 RUN_PROCEDURE ( "convert_real_var" )
endProcedure

Procedure
@name = convert_real_var
@do =
 <dummy.convert.@string>.std_amt.@float  = ( <dummy.convert.@string>.real_amt.@float  -
<dummy.convert.@string>.real_amt_min.@float  ) / ( <dummy.convert.@string>.real_amt_max.@float  -
<dummy.convert.@string>.real_amt_min.@float  ) * 100
endProcedure

Procedure
@name = dynamics_predicted
@do =
 <dummy.var.@string>.p_dyn.@float  = RELATIVECHANGE ( <dummy.var.@string>.p_amt.@float  , 0,
<dummy.var.@string>.rate_of_change.@float  )
endProcedure

Procedure
@name = dynamics_real
@do =
 <dummy.var.@string>.real_dyn.@float  = RELATIVECHANGE ( <dummy.var.@string>.real_amt.@float  , 0,
<dummy.var.@string>.rate_of_change.@float  )
endProcedure

Procedure
@name = generate_files
@do =
 EXPORT ( "../tmp/s_time.rep+", "timex.elapsed.@float", 0, 100 )
 EXPORT ( "../tmp/l_pred.rep+", "level.p_amt.@float", 0, 100 )
```

```
 EXPORT ( "../tmp/l_real.rep+", "level.real_amt.@float", 0, 100 )
 EXPORT ( "../tmp/f_real.rep+", "frate_out.real_amt.@float", 0, 100 )
 EXPORT ( "../tmp/f_pred.rep+", "frate_out.p_amt.@float", 0, 100 )
endProcedure

Procedure
@name = global_manager
@do =
 DISABLEBACKWARDCHAIN ( TRUE )
 DISABLEFORWARDCHAIN ( TRUE )
 TEXT ( "0" )
 MACRO ( "continue*" )
 TEXT ( "1" )
 MACRO ( "initialization*" )
 TEXT ( "2" )
 RUN_PROCEDURE ( "read_data_coordinator" )
 TEXT ( "3" )
 RUN_PROCEDURE ( "QM_simulator_coordinator" )
 TEXT ( "4" )
 DISABLEBACKWARDCHAIN ( FALSE )
 DISABLEFORWARDCHAIN ( FALSE )
 RUN_PROCEDURE ( "reasoning_coordinator" )
 TEXT ( "5" )
 RUN_PROCEDURE ( "update_database" )
 RUN_PROCEDURE ( "update_simulation_time" )
endProcedure

Procedure
@name = initial_tank_simulation_values
@do =
 TEXT ( "initial tank sim values" )
 frate_in.real_amt.@float  = 100.000000
 frate_in.real_amt_old.@float  = 0.000000
 level.real_amt_old.@float  = 0.000000
 frate_out.real_amt_old.@float  = 0.000000
 valve.opening.@float  = 75.000000
endProcedure

Procedure
@name = initial_values
@do =
 level.p_amt_old.@float  = 0.000000
 timex.elapsed_old.@float  = 0.000000
 level.qmodel_ss.@string is "mss_level1"
endProcedure

Procedure
@name = initialize_files
@do =
 EXPORT ( "../tmp/f_real.rep", "frate_out.real_amt.@float", 0, 100 )
 EXPORT ( "../tmp/f_pred.rep", "frate_out.p_amt.@float", 0, 100 )
 EXPORT ( "../tmp/l_real.rep", "level.real_amt.@float", 0, 100 )
 EXPORT ( "../tmp/l_pred.rep", "level.p_amt.@float", 0, 100 )
 EXPORT ( "../tmp/s_time.rep", "timex.elapsed.@float", 0, 100 )
endProcedure

Procedure
@name = mathematical_simulation_coordinator
@do =
 TEXT ( "mathematical sim  coordinator" )
 MACRO ( "tank_simulation_init" )
```

```
 RUN_PROCEDURE ( "mss_tank_outlet_run" )
 MACRO ( "simulate_tank" )
endProcedure

Procedure
@name = mdyn_frate_out
@do =
 RUN_PROCEDURE ( "mss_tank_outlet_run" )
 frate_out.p_amt.@float  = tank.outlet_restriction.@float  * SQRT ( level.real_amt.@float  )
endProcedure

Procedure
@name = mdyn_level1
@do =
 qm_dyn_first.output_ss.@double  = level.p_amt_ss.@float
 qm_dyn_first.output_old.@double  = level.p_amt_old.@float
 qm_dyn_first.time_s_state.@float  = level.time_s_state.@float
 RUN_PROCEDURE ( "qm_dyn_first" )
 level.p_amt.@float  = qm_dyn_first.output.@double
endProcedure

Procedure
@name = mss_frate_out
@do =
 qm_proportional.input.@double  = mss_frate_out.input1.@double
 qm_proportional.constant.@float  = 1
 RUN_PROCEDURE ( "qm_proportional" )
 mss_frate_out.output.@double  = qm_proportional.output.@double
endProcedure

Procedure
@name = mss_frate_out_run
@do =
 mss_frate_out.input1.@double  = frate_in.real_amt.@float
 RUN_PROCEDURE ( "mss_frate_out" )
 frate_out.p_amt_ss.@float  = mss_frate_out.output.@double
endProcedure

Procedure
@name = mss_level1
@do =
 qm_prop_square.input.@double  = mss_level1.input1.@double
 RUN_PROCEDURE ( "qm_prop_square" )
 mss_level1.dummy1.@double  = qm_prop_square.output.@double
 qm_prop_square.input.@double  = tank.outlet_restriction.@float
 RUN_PROCEDURE ( "qm_prop_square" )
 mss_level1.dummy2.@double  = qm_prop_square.output.@double
 qm_division.input1.@double  = mss_level1.dummy1.@double
 qm_division.input2.@double  = mss_level1.dummy2.@double
 RUN_PROCEDURE ( "qm_division" )
 mss_level1.output.@double  = qm_division.output.@double
endProcedure

Procedure
@name = mss_level2
@do =
 qm_table3.input_l.@float  = mss_level2.input_l.@float
 qm_table3.input_m.@float  = mss_level2.input_m.@float
 qm_table3.input_h.@float  = mss_level2.input_h.@float
 qm_table3.output_l.@float  = mss_level2.output_l.@float
 qm_table3.output_m.@float  = mss_level2.output_m.@float
```

```
 qm_table3.output_h.@float  = mss_level2.output_h.@float
 qm_table3.input.@double  = mss_level2.input1.@double
 MACRO ( "qm_table3" )
 mss_level2.output.@double  = qm_table3.output.@double
endProcedure

Procedure
@name = mss_level2_definition
@do =
 mss_level2.input_l.@float  = 0
 mss_level2.output_l.@float  = 0
 mss_level2.input_m.@float  = 50
 mss_level2.output_m.@float  = 0.500000
 mss_level2.input_h.@float  = 100
 mss_level2.output_h.@float  = 2
endProcedure

Procedure
@name = mss_level3
@do =
 mss_level3.output.@double  = ( mss_level3.input1.@double  / tank.outlet_restriction.@float  ) ^ 2
endProcedure

Procedure
@name = mss_level_run
@do =
 <level.qmodel_ss.@string>.input1.@double  = frate_in.real_amt.@float
 <level.qmodel_ss.@string>.input2.@double  = valve.opening.@float
 <level.qmodel_ss.@string>.parameter.@float  = 1
 RUN_PROCEDURE ( level.qmodel_ss.@string )
 level.p_amt_ss.@float  = <level.qmodel_ss.@string>.output.@double
endProcedure

Procedure
@name = mss_tank_outlet
@do =
 mss_tank_outlet.output.@double  = mss_tank_outlet.input1.@double  * mss_tank_outlet.input2.@double
endProcedure

Procedure
@name = mss_tank_outlet_run
@do =
 TEXT ( "mss tank outlet run" )
 mss_tank_outlet.input1.@double  = tank.outlet_parameter.@float
 mss_tank_outlet.input2.@double  = valve.opening.@float
 RUN_PROCEDURE ( "mss_tank_outlet" )
 tank.outlet_restriction.@float  = mss_tank_outlet.output.@double
endProcedure

Procedure
@name = obtain_predicted_dynamics
@do =
 STRCOPY ( dummy.var.@string, "level" )
 RUN_PROCEDURE ( "dynamics_predicted" )
 dummy.var.@string is "frate_out"
 RUN_PROCEDURE ( "dynamics_predicted" )
endProcedure

Procedure
@name = obtain_real_dynamics
@do =
```

```
 STRCOPY ( dummy.var.@string, "level" )
 RUN_PROCEDURE ( "dynamics_real" )
 STRCOPY ( dummy.var.@string, "frate_in" )
 RUN_PROCEDURE ( "dynamics_real" )
 dummy.var.@string is "frate_out"
 RUN_PROCEDURE ( "dynamics_real" )
endProcedure

Procedure
@name = qm_addition
@do =
 qm_addition.output.@double = qm_addition.input1.@double + qm_addition.input2.@double
endProcedure

Procedure
@name = qm_difference
@do =
 qm_difference.output.@double = qm_difference.input1.@double - qm_difference.input2.@double
endProcedure

Procedure
@name = qm_division
@do =
 qm_division.output.@double = qm_division.input1.@double / qm_division.input2.@double
endProcedure

Procedure
@name = qm_dyn_first
@do =
 qm_dyn_first.dummy.@double = EXP ( - 3 / qm_dyn_first.time_s_state.@float * timex.sampling.@float )
 qm_dyn_first.output.@double = qm_dyn_first.output_old.@double * qm_dyn_first.dummy.@double + ( 1 -
qm_dyn_first.dummy.@double ) * qm_dyn_first.output_ss.@double
endProcedure

Procedure
@name = qm_multiplication
@do =
 qm_multiplication.output.@double = qm_multiplication.input1.@double * qm_multiplication.input2.@double
endProcedure

Procedure
@name = qm_prop_square
@do =
 qm_prop_square.output.@double = qm_prop_square.input.@double ^ 2
endProcedure

Procedure
@name = qm_proportional
@do =
 qm_proportional.output.@double = qm_proportional.input.@double * qm_proportional.constant.@float
endProcedure

Procedure
@name = qm_table3_high
@do =
 qm_table3.slope.@double = ( qm_table3.output_h.@float - qm_table3.output_m.@float ) / ( qm_table3.input_h.@float
- qm_table3.input_m.@float )
 qm_table3.output.@double = ( qm_table3.input.@double - qm_table3.input_m.@float ) * qm_table3.slope.@double +
qm_table3.output_m.@float
endProcedure
```

```
Procedure
@name = qm_table3_low
@do =
 qm_table3.slope.@double = ( qm_table3.output_m.@float - qm_table3.output_l.@float ) / ( qm_table3.input_m.@float
- qm_table3.input_l.@float )
 qm_table3.output.@double = ( qm_table3.input.@double - qm_table3.input_l.@float ) * qm_table3.slope.@double +
qm_table3.output_l.@float
endProcedure

Procedure
@name = re_configuration
@do =
 runx.simulation.again is FALSE
 timex.elapsed.@float = 0
endProcedure

Procedure
@name = read_data
@do =
 TEXT ( "read data " )
 GETPOINTDATA ( frate_in.real_amt.@float )
 GETPOINTDATA ( frate_in.real_amt_old.@float )
 GETPOINTDATA ( level.real_amt.@float )
 GETPOINTDATA ( level.real_amt_old.@float )
 GETPOINTDATA ( frate_out.real_amt.@float )
 GETPOINTDATA ( frate_out.real_amt_old.@float )
 GETPOINTDATA ( valve.opening.@float )
endProcedure

Procedure
@name = read_data_coordinator
@do =
 TEXT ( "read data coordinator" )
 RUN_PROCEDURE ( "read_data" )
 RUN_PROCEDURE ( "basic_data_processing" )
 RUN_PROCEDURE ( "mathematical_simulation_coordinator" )
 RUN_PROCEDURE ( "obtain_real_dynamics" )
endProcedure

Procedure
@name = reasoning_coordinator
@do =
 analyze.simulation.results is TRUE
 dummy.reasoning.@string is "level"
 MACRO ( "reas*" )
 FREERULE ( $Rule, "reas*" )
 dummy.reasoning.@string is "frate_out"
 MACRO ( "reas*" )
 analyze.simulation.results is FALSE
endProcedure

Procedure
@name = rotate_all_predictions
@do =
 STRCOPY ( dummy.rotate.@string, "level" )
 RUN_PROCEDURE ( "rotate_predicted_var" )
 STRCOPY ( dummy.rotate.@string, "frate_out" )
 RUN_PROCEDURE ( "rotate_predicted_var" )
endProcedure

Procedure
```

```
@name = rotate_predicted_var
@do =
 <dummy.rotate.@string>.p_amt_old.@float  = <dummy.rotate.@string>.p_amt.@float
endProcedure

Procedure
@name = run_model_dyn
@do =
 RUN_PROCEDURE ( "mdyn_level1" )
 RUN_PROCEDURE ( "mdyn_frate_out" )
endProcedure

Procedure
@name = run_model_ss
@do =
 RUN_PROCEDURE ( "mss_tank_outlet_run" )
 RUN_PROCEDURE ( "mss_level_run" )
 RUN_PROCEDURE ( "mss_frate_out_run" )
endProcedure

Procedure
@name = simulation_parameters
@do =
 timex.sampling.@float  = 5
 timex.sleep.@float  = 0.000000
endProcedure

Procedure
@name = start_simulation
@do =
 runx.simulation.again is TRUE
 PUTPOINTDATA ( "runx.simulation.again", 100 )
endProcedure

Procedure
@name = stop_simulation
@do =
 runx.simulation.again is FALSE
 PUTPOINTDATA ( "runx.simulation.again", 0 )
endProcedure

Procedure
@name = system_initialization
@do =
 RUN_PROCEDURE ( " simulation_parameters" )
 RUN_PROCEDURE ( "tank_definition" )
 RUN_PROCEDURE ( "initial_values" )
 RUN_PROCEDURE ( "mss_level2_definition" )
 RUN_PROCEDURE ( "initialize_files" )
endProcedure

Procedure
@name = tank_definition
@do =
 tank.area.@float  = 2.000000
 tank.capacity.@float  = tank.area.@float  * 1000
 tank.height.@float  = 2
 tank.outlet_parameter.@float  = 0.942809
 level.real_amt_max.@float  = 2
 level.real_amt_min.@float  = 0
 frate_in.real_amt_max.@float  = 100
```

```
 frate_in.real_amt_min.@float  = 0
 frate_out.real_amt_max.@float  = 100
 frate_out.real_amt_min.@float  = 0
 level.time_s_state.@float  = 110
 frate_out.time_s_state.@float  = 110
 level.rate_of_change.@float  = 5
 frate_in.rate_of_change.@float  = 5
 frate_out.rate_of_change.@float  = 5
endProcedure

Procedure
@name = tank_simulation
@do =
 dummy.expo.@double = - tank.outlet_restriction.@float  * SQRT ( level.real_amt_old.@float  ) / tank.capacity.@float  +
frate_in.real_amt_old.@float  / tank.capacity.@float
 dummy.pred_level.@double = level.real_amt_old.@float  + dummy.expo.@double  * timex.sampling.@float
 dummy.eval_pred.@double = - tank.outlet_restriction.@float  * SQRT ( dummy.pred_level.@double ) /
tank.capacity.@float  + frate_in.real_amt_old.@float  / tank.capacity.@float
 level.real_amt.@float  = level.real_amt_old.@float  + ( dummy.expo.@double  + dummy.eval_pred.@double ) *
timex.sampling.@float  / 2
 frate_out.real_amt.@float  = tank.outlet_restriction.@float  * SQRT ( level.real_amt.@float  )
endProcedure

Procedure
@name = tank_simulation_R_Kutta
@do =
 dummy.k1.@double = ( - tank.outlet_restriction.@float  * SQRT ( level.real_amt_old.@float  ) / tank.capacity.@float  +
frate_in.real_amt_old.@float  / tank.capacity.@float  ) * timex.sampling.@float
 dummy.k2.@double = ( - tank.outlet_restriction.@float  * SQRT ( level.real_amt_old.@float  + dummy.k1.@double / 2 )
/ tank.capacity.@float  + frate_in.real_amt_old.@float  / tank.capacity.@float  ) * timex.sampling.@float
 dummy.k3.@double = ( - tank.outlet_restriction.@float  * SQRT ( level.real_amt_old.@float  + dummy.k2.@double / 2 )
/ tank.capacity.@float  + frate_in.real_amt_old.@float  / tank.capacity.@float  ) * timex.sampling.@float
 dummy.k4.@double = ( - tank.outlet_restriction.@float  * SQRT ( level.real_amt_old.@float  + dummy.k3.@double ) /
tank.capacity.@float  + frate_in.real_amt_old.@float  / tank.capacity.@float  ) * timex.sampling.@float
 level.real_amt.@float  = level.real_amt_old.@float  + ( dummy.k1.@double  + 2 * dummy.k2.@double  + 2 *
dummy.k3.@double + dummy.k4.@double ) / 6
 frate_out.real_amt.@float  = tank.outlet_restriction.@float  * SQRT ( level.real_amt.@float  )
endProcedure

Procedure
@name = tank_simulation_euler
@do =
 dummy.expo.@double = - tank.outlet_restriction.@float  * SQRT ( level.real_amt_old.@float  ) / tank.capacity.@float  +
frate_in.real_amt_old.@float  / tank.capacity.@float
 level.real_amt.@float  = level.real_amt_old.@float  + dummy.expo.@double  * timex.sampling.@float
 frate_out.real_amt.@float  = tank.outlet_restriction.@float  * SQRT ( level.real_amt.@float  )
endProcedure

Procedure
@name = tank_simulation_euler_mod
@do =
 dummy.expo.@double = - tank.outlet_restriction.@float  * SQRT ( level.real_amt_old.@float  ) / tank.capacity.@float  +
frate_in.real_amt_old.@float  / tank.capacity.@float
 dummy.pred_level.@double = level.real_amt_old.@float  + dummy.expo.@double  * timex.sampling.@float
 dummy.eval_pred.@double = - tank.outlet_restriction.@float  * SQRT ( dummy.pred_level.@double ) /
tank.capacity.@float  + frate_in.real_amt_old.@float  / tank.capacity.@float
 level.real_amt.@float  = level.real_amt_old.@float  + ( dummy.expo.@double  + dummy.eval_pred.@double ) *
timex.sampling.@float  / 2
 frate_out.real_amt.@float  = tank.outlet_restriction.@float  * SQRT ( level.real_amt.@float  )
endProcedure
```

```
Procedure
@name = tank_simulation_orig
@do =
 dummy.expo.@double  = EXP ( - tank.outlet_restriction.@float  / 2 / tank.capacity.@float  / SQRT (
level.real_amt_old.@float  ) * timex.sampling.@float  )
 level.real_amt.@float  = level.real_amt_old.@float  * dummy.expo.@double + SQRT ( level.real_amt_old.@float  ) /
tank.outlet_restriction.@float  * ( 1 - dummy.expo.@double  ) * frate_in.real_amt_old.@float
 frate_out.real_amt.@float  = tank.outlet_restriction.@float  * SQRT ( level.real_amt.@float  )
endProcedure

Procedure
@name = update_database
@do =
 PUTPOINTDATA ( "frate_in.real_amt_old.@f", frate_in.real_amt.@float , 100 )
 PUTPOINTDATA ( "level.real_amt_old.@f", level.real_amt.@float , 100 )
 PUTPOINTDATA ( "frate_out.real_amt_old.@f", frate_out.real_amt.@float , 100 )
endProcedure

Procedure
@name = update_simulation_time
@do =
 timex.current.@integer = SECOND ( $Current_time )
 SLEEP ( timex.sleep.@float  )
 timex.elapsed.@float  = timex.elapsed_old.@float  + timex.sampling.@float
 timex.elapsed_old.@float  = timex.elapsed.@float
endProcedure

Restriction
@name = zero
@constraint =  $OAV >= 0.000100
endRestriction

Ifchange
@name = max_certainty
@objectinherit =  ASNCERTAINTY ( $OAV, 100 )
endIfchange

Fuzzy
@name = frate_in_p_amt_high
@source =  frate_in.p_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 0.000000, 50.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_in_p_amt_low
@source =  frate_in.p_amt.@float
@range = 3
@value = 0.000000, 50.000000, 100.000000
@rank = 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_in_p_amt_medium
@source =  frate_in.p_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 100.000000, 50.000000, 0.000000, 0.000000
endFuzzy
```

```
Fuzzy
@name = frate_in_p_amt_pos
@source =  frate_in.p_amt.@float
@range = 2
@value = -100.000000, 100.000000
@rank = 0.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_in_p_dyn_cnt
@source =  frate_in.p_dyn.@float
@range = 5
@value = -100.000000, -10.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_in_p_dyn_dec
@source =  frate_in.p_dyn.@float
@range = 4
@value = -100.000000, -10.000000, 0.000000, 100.000000
@rank = 100.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_in_p_dyn_inc
@source =  frate_in.p_dyn.@float
@range = 4
@value = -100.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_in_real_amt_high
@source =  frate_in.std_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 0.000000, 50.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_in_real_amt_low
@source =  frate_in.std_amt.@float
@range = 3
@value = 0.000000, 50.000000, 100.000000
@rank = 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_in_real_amt_medium
@source =  frate_in.std_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 100.000000, 50.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_in_real_dyn_cnt
@source =  frate_in.real_dyn.@float
@range = 5
@value = -100.000000, -10.000000, 0.000000, 10.000000, 100.000000
```

```
@rank = 0.000000, 0.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_in_real_dyn_dec
@source =  frate_in.real_dyn.@float
@range = 4
@value = -100.000000, -10.000000, 0.000000, 100.000000
@rank = 100.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_in_real_dyn_inc
@source =  frate_in.real_dyn.@float
@range = 4
@value = -100.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_out_diff_amt_pred_large
@source =  frate_out.diff_amt_pred.@float
@range = 4
@value = 0.000000, 5.000000, 15.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_out_diff_amt_pred_medium
@source =  frate_out.diff_amt_pred.@float
@range = 4
@value = 0.000000, 5.000000, 15.000000, 100.000000
@rank = 0.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_out_diff_amt_pred_small
@source =  frate_out.diff_amt_pred.@float
@range = 3
@value = 0.000000, 5.000000, 100.000000
@rank = 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_out_p_amt_high
@source =  frate_out.p_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 0.000000, 50.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_out_p_amt_low
@source =  frate_out.p_amt.@float
@range = 3
@value = 0.000000, 50.000000, 100.000000
@rank = 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_out_p_amt_medium
```

@source = frate_out.p_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 100.000000, 50.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_out_p_amt_pos
@source = frate_out.p_amt.@float
@range = 2
@value = -100.000000, 100.000000
@rank = 0.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_out_p_dyn_cnt
@source = frate_out.p_dyn.@float
@range = 5
@value = -100.000000, -10.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_out_p_dyn_dec
@source = frate_out.p_dyn.@float
@range = 4
@value = -100.000000, -10.000000, 0.000000, 100.000000
@rank = 100.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_out_p_dyn_inc
@source = frate_out.p_dyn.@float
@range = 4
@value = -100.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_out_real_amt_high
@source = frate_out.std_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 0.000000, 50.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = frate_out_real_amt_low
@source = frate_out.std_amt.@float
@range = 3
@value = 0.000000, 50.000000, 100.000000
@rank = 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_out_real_amt_medium
@source = frate_out.std_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 100.000000, 50.000000, 0.000000, 0.000000
endFuzzy

```
Fuzzy
@name = frate_out_real_dyn_cnt
@source =  frate_out.real_dyn.@float
@range = 5
@value = -100.000000, -10.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_out_real_dyn_dec
@source =  frate_out.real_dyn.@float
@range = 4
@value = -100.000000, -10.000000, 0.000000, 100.000000
@rank = 100.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = frate_out_real_dyn_inc
@source =  frate_out.real_dyn.@float
@range = 4
@value = -100.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = level_diff_amt_pred_large
@source =  level.diff_amt_pred.@float
@range = 4
@value = 0.000000, 5.000000, 15.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = level_diff_amt_pred_medium
@source =  level.diff_amt_pred.@float
@range = 4
@value = 0.000000, 5.000000, 15.000000, 100.000000
@rank = 0.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = level_diff_amt_pred_small
@source =  level.diff_amt_pred.@float
@range = 3
@value = 0.000000, 5.000000, 100.000000
@rank = 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = level_p_amt_high
@source =  level.p_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 0.000000, 50.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = level_p_amt_low
@source =  level.p_amt.@float
@range = 3
```

@value = 0.000000, 50.000000, 100.000000
@rank = 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = level_p_amt_medium
@source = level.p_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 100.000000, 50.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = level_p_amt_pos
@source = level.p_amt.@float
@range = 2
@value = -100.000000, 100.000000
@rank = 0.000000, 100.000000
endFuzzy

Fuzzy
@name = level_p_dyn_cnt
@source = level.p_dyn.@float
@range = 5
@value = -100.000000, -10.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = level_p_dyn_dec
@source = level.p_dyn.@float
@range = 4
@value = -100.000000, -10.000000, 0.000000, 100.000000
@rank = 100.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = level_p_dyn_inc
@source = level.p_dyn.@float
@range = 4
@value = -100.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = level_real_amt_high
@source = level.std_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 0.000000, 50.000000, 100.000000, 100.000000
endFuzzy

Fuzzy
@name = level_real_amt_low
@source = level.std_amt.@float
@range = 3
@value = 0.000000, 50.000000, 100.000000
@rank = 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy

```
@name = level_real_amt_medium
@source =  level.std_amt.@float
@range = 5
@value = 0.000000, 50.000000, 75.000000, 90.000000, 100.000000
@rank = 0.000000, 100.000000, 50.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = level_real_amt_pos
@source =  level.std_amt.@float
@range = 2
@value = -100.000000, 100.000000
@rank = 0.000000, 100.000000
endFuzzy

Fuzzy
@name = level_real_dyn_cnt
@source =  level.real_dyn.@float
@range = 5
@value = -100.000000, -10.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = level_real_dyn_dec
@source =  level.real_dyn.@float
@range = 4
@value = -100.000000, -10.000000, 0.000000, 100.000000
@rank = 100.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = level_real_dyn_inc
@source =  level.real_dyn.@float
@range = 4
@value = -100.000000, 0.000000, 10.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 100.000000
endFuzzy

Rule
@name = change_var1
@priority = 0
IF  timex.elapsed.@float  == 500
THEN frate_in.real_amt.@float  = 50
THEN PUTPOINTDATA ( "frate_in.real_amt.@f", frate_in.real_amt.@float , 100 )
endRule

Rule
@name = change_var2
@priority = 0
IF  timex.elapsed.@float  == 1100
THEN frate_in.real_amt.@float  = 75
THEN PUTPOINTDATA ( "frate_in.real_amt.@f", frate_in.real_amt.@float , 100 )
endRule

Rule
@name = continue_1
@priority = 10
IF  TRUE
THEN GETPOINTDATA ( "runx.simulation.again" )
THEN FREERULE ( $Rule, "continue_2" )
```

```
THEN GOTO ( "continue_2" )
endRule

Rule
@name = continue_2
@priority = 10
IF  runx.simulation.again
THEN DONOTHING ( )
ELSE FREERULE ( $Rule, "continue_1" )
ELSE timex.elapsed.@float  = 0
ELSE GOTO ( "continue_1" )
endRule

Rule
@name = global_manager
@priority = 10
IF  TRUE
THEN DISABLEBACKWARDCHAIN ( TRUE )
THEN DISABLEFORWARDCHAIN ( TRUE )
THEN MACRO ( "continue*" )
THEN DISPLAY ( "here", "message-1" )
THEN MACRO ( "initialization*" )
THEN RUN_PROCEDURE ( "read_data_coordinator" )
THEN RUN_PROCEDURE ( "QM_simulator_coordinator" )
THEN DISABLEBACKWARDCHAIN ( FALSE )
THEN DISABLEFORWARDCHAIN ( FALSE )
THEN RUN_PROCEDURE ( "reasoning_coordinator" )
THEN RUN_PROCEDURE ( "update_database" )
THEN RUN_PROCEDURE ( "update_simulation_time" )
THEN RUN_PROCEDURE ( "generate_files" )
THEN MACRO ( "change_var*" )
endRule

Rule
@name = initialization_1
IF  system.already.configured
THEN DONOTHING ( )
ELSE RUN_PROCEDURE ( "system_initialization" )
ELSE system.already.configured is TRUE
endRule

Rule
@name = qm_table3
IF  qm_table3.input.@double  <= qm_table3.input_m.@float
THEN RUN_PROCEDURE ( "qm_table3_low" )
ELSE RUN_PROCEDURE ( "qm_table3_high" )
endRule

Rule
@name = qual_status_p_cnt
@priority = 0
IF  ?variable = ANY ( {p_var}.p_dyn.cnt is TRUE )
THEN ALL ( ?variable.p_dyn_status.@string is "constant" )
endRule

Rule
@name = qual_status_p_dec
@priority = 0
IF  ?variable = ANY ( {p_var}.p_dyn.dec is TRUE )
THEN ALL ( ?variable.p_dyn_status.@string is "decreasing" )
endRule
```

```
Rule
@name = qual_status_p_high
@priority = 0
IF  ?variable = ANY ( {p_var}.p_amt.high is TRUE )
THEN ALL ( ?variable.p_amt_status.@string is "high" )
endRule

Rule
@name = qual_status_p_inc
@priority = 0
IF  ?variable = ANY ( {p_var}.p_dyn.inc is TRUE )
THEN ALL ( ?variable.p_dyn_status.@string is "increasing" )
endRule

Rule
@name = qual_status_p_low
@priority = 0
IF  ?variable = ANY ( {p_var}.p_amt.low is TRUE )
THEN ALL ( ?variable.p_amt_status.@string is "low" )
endRule

Rule
@name = qual_status_p_medium
@priority = 0
IF  ?variable = ANY ( {p_var}.p_amt.medium is TRUE )
THEN ALL ( ?variable.p_amt_status.@string is "medium" )
endRule

Rule
@name = qual_status_real_cnt
@priority = 0
IF  ?variable = ANY ( {p_var}.real_dyn.cnt is TRUE )
THEN ALL ( ?variable.real_dyn_status.@string is "constant" )
endRule

Rule
@name = qual_status_real_dec
@priority = 0
IF  ?variable = ANY ( {p_var}.real_dyn.dec is TRUE )
THEN ALL ( ?variable.real_dyn_status.@string is "decreasing" )
endRule

Rule
@name = qual_status_real_high
@priority = 0
IF  ?variable = ANY ( {p_var}.real_amt.high is TRUE )
THEN ALL ( ?variable.real_amt_status.@string is "high" )
endRule

Rule
@name = qual_status_real_inc
@priority = 0
IF  ?variable = ANY ( {p_var}.real_dyn.inc is TRUE )
THEN ALL ( ?variable.real_dyn_status.@string is "increasing" )
endRule

Rule
@name = qual_status_real_low
@priority = 0
IF  ?variable = ANY ( {p_var}.real_amt.low is TRUE )
```

```
THEN ALL ( ?variable.real_amt_status.@string is "low" )
endRule

Rule
@name = qual_status_real_medium
@priority = 0
IF  ?variable = ANY ( {p_var}.real_amt.medium is TRUE )
THEN ALL ( ?variable.real_amt_status.@string is "medium" )
endRule

Rule
@name = reasoning_0
@priority = 0
IF  analyze.simulation.results
THEN dummy.std_amt.@float  = ( <dummy.reasoning.@string>.p_amt.@float  -
<dummy.reasoning.@string>.real_amt_min.@float  ) / ( <dummy.reasoning.@string>.real_amt_max.@float  -
<dummy.reasoning.@string>.real_amt_min.@float  ) * 100
THEN <dummy.reasoning.@string>.diff_amt_pred.@float  = ABS ( <dummy.reasoning.@string>.std_amt.@float  -
dummy.std_amt.@float  )
THEN FORGET ( " <model.d_out.@s>.diff_amt_mp.medium" )
THEN FORGET ( " <model.d_out.@s>.diff_amt_mp.large" )
THEN FORGET ( "none.of_these.work" )
endRule

Rule
@name = reasoning_1
@priority = 0
IF  analyze.simulation.results
AND  <dummy.reasoning.@string>.diff_amt_pred.small
THEN TEXT ( ".. Measured and predicted  !$dummy.reasoning.@s$!  agree!...
 !$<dummy.reasoning.@s>.diff_amt_pred.@f $!" )
endRule

Rule
@name = reasoning_2
@priority = 0
IF  analyze.simulation.results
AND  <dummy.reasoning.@string>.diff_amt_pred.medium
THEN TEXT ( ".. they are close!... !$  <dummy.reasoning.@s>.diff_amt_pred.@f $!" )
endRule

Rule
@name = reasoning_3
@priority = 0
IF  analyze.simulation.results
AND  <dummy.reasoning.@string>.diff_amt_pred.large
THEN TEXT ( ".. there are some PROBLEMS!!!!... !$ <dummy.reasoning.@s>.diff_amt_pred.@f $!" )
endRule

Rule
@name = simulate_tank
@priority = 0
IF  level.real_amt_old.@float  <= 0
THEN level.real_amt_old.@float  = 0.001000
THEN RUN_PROCEDURE ( "tank_simulation" )
ELSE RUN_PROCEDURE ( "tank_simulation" )
endRule

Rule
@name = tank_simulation_init
IF  tank.simulation.initialized
```

```
THEN DONOTHING ( )
ELSE RUN_PROCEDURE ( "initial_tank_simulation_values" )
ELSE tank.simulation.initialized is TRUE
endRule

Facets
@triplet = {model_ss}.output.@double
@ifchange = max_certainty
endFacets

Facets
@triplet = frate_in.p_amt.high
@fuzzy = frate_in_p_amt_high
endFacets

Facets
@triplet = frate_in.p_amt.low
@fuzzy = frate_in_p_amt_low
endFacets

Facets
@triplet = frate_in.p_amt.medium
@fuzzy = frate_in_p_amt_medium
endFacets

Facets
@triplet = frate_in.p_amt.pos
@fuzzy = frate_in_p_amt_pos
endFacets

Facets
@triplet = frate_in.p_dyn.cnt
@fuzzy = frate_in_p_dyn_cnt
endFacets

Facets
@triplet = frate_in.p_dyn.dec
@fuzzy = frate_in_p_dyn_dec
endFacets

Facets
@triplet = frate_in.p_dyn.inc
@fuzzy = frate_in_p_dyn_inc
endFacets

Facets
@triplet = frate_in.real_amt.high
@fuzzy = frate_in_real_amt_high
endFacets

Facets
@triplet = frate_in.real_amt.low
@fuzzy = frate_in_real_amt_low
endFacets

Facets
@triplet = frate_in.real_amt.medium
@fuzzy = frate_in_real_amt_medium
endFacets

Facets
```

@triplet = frate_in.real_dyn.cnt
@fuzzy = frate_in_real_dyn_cnt
endFacets

Facets
@triplet = frate_in.real_dyn.dec
@fuzzy = frate_in_real_dyn_dec
endFacets

Facets
@triplet = frate_in.real_dyn.inc
@fuzzy = frate_in_real_dyn_inc
endFacets

Facets
@triplet = frate_out.diff_amt_pred.large
@fuzzy = frate_out_diff_amt_pred_large
endFacets

Facets
@triplet = frate_out.diff_amt_pred.medium
@fuzzy = frate_out_diff_amt_pred_medium
endFacets

Facets
@triplet = frate_out.diff_amt_pred.small
@fuzzy = frate_out_diff_amt_pred_small
endFacets

Facets
@triplet = frate_out.diff_amt_pred.@float
@ifchange = max_certainty
endFacets

Facets
@triplet = frate_out.p_amt.high
@fuzzy = frate_out_p_amt_high
endFacets

Facets
@triplet = frate_out.p_amt.low
@fuzzy = frate_out_p_amt_low
endFacets

Facets
@triplet = frate_out.p_amt.medium
@fuzzy = frate_out_p_amt_medium
endFacets

Facets
@triplet = frate_out.p_amt.pos
@fuzzy = frate_out_p_amt_pos
endFacets

Facets
@triplet = frate_out.p_dyn.cnt
@fuzzy = frate_out_p_dyn_cnt
endFacets

Facets
@triplet = frate_out.p_dyn.dec

@fuzzy = frate_out_p_dyn_dec
endFacets

Facets
@triplet = frate_out.p_dyn.inc
@fuzzy = frate_out_p_dyn_inc
endFacets

Facets
@triplet = frate_out.real_amt.high
@fuzzy = frate_out_real_amt_high
endFacets

Facets
@triplet = frate_out.real_amt.low
@fuzzy = frate_out_real_amt_low
endFacets

Facets
@triplet = frate_out.real_amt.medium
@fuzzy = frate_out_real_amt_medium
endFacets

Facets
@triplet = frate_out.real_dyn.cnt
@fuzzy = frate_out_real_dyn_cnt
endFacets

Facets
@triplet = frate_out.real_dyn.dec
@fuzzy = frate_out_real_dyn_dec
endFacets

Facets
@triplet = frate_out.real_dyn.inc
@fuzzy = frate_out_real_dyn_inc
endFacets

Facets
@triplet = level.diff_amt_pred.large
@fuzzy = level_diff_amt_pred_large
endFacets

Facets
@triplet = level.diff_amt_pred.medium
@fuzzy = level_diff_amt_pred_medium
endFacets

Facets
@triplet = level.diff_amt_pred.small
@fuzzy = level_diff_amt_pred_small
endFacets

Facets
@triplet = level.diff_amt_pred.@float
@ifchange = max_certainty
endFacets

Facets
@triplet = level.p_amt.high
@fuzzy = level_p_amt_high

```
endFacets

Facets
@triplet = level.p_amt.low
@fuzzy = level_p_amt_low
endFacets

Facets
@triplet = level.p_amt.medium
@fuzzy = level_p_amt_medium
endFacets

Facets
@triplet = level.p_amt.pos
@fuzzy = level_p_amt_pos
endFacets

Facets
@triplet = level.p_amt.@float
@ifchange = max_certainty
endFacets

Facets
@triplet = level.p_amt_old.@float
@ifchange = max_certainty
endFacets

Facets
@triplet = level.p_amt_ss.@float
@ifchange = max_certainty
endFacets

Facets
@triplet = level.p_dyn.cnt
@fuzzy = level_p_dyn_cnt
endFacets

Facets
@triplet = level.p_dyn.dec
@fuzzy = level_p_dyn_dec
endFacets

Facets
@triplet = level.p_dyn.inc
@fuzzy = level_p_dyn_inc
endFacets

Facets
@triplet = level.p_dyn.@float
@ifchange = max_certainty
endFacets

Facets
@triplet = level.real_amt.high
@fuzzy = level_real_amt_high
endFacets

Facets
@triplet = level.real_amt.low
@fuzzy = level_real_amt_low
endFacets
```

```
Facets
@triplet = level.real_amt.medium
@fuzzy = level_real_amt_medium
endFacets

Facets
@triplet = level.real_amt.pos
@fuzzy = level_real_amt_pos
endFacets

Facets
@triplet = level.real_dyn.cnt
@fuzzy = level_real_dyn_cnt
endFacets

Facets
@triplet = level.real_dyn.dec
@fuzzy = level_real_dyn_dec
endFacets

Facets
@triplet = level.real_dyn.inc
@fuzzy = level_real_dyn_inc
endFacets

Facets
@triplet = level.real_dyn.@float
@ifchange = max_certainty
endFacets

Facets
@triplet = level.std_amt.@float
@ifchange = max_certainty
endFacets

Facets
@triplet = mss_level1.parameter.@float
@default = 1.000000, 100.000000
endFacets

Facets
@triplet = qm_division.output.@double
@ifchange = max_certainty
endFacets

Facets
@triplet = qm_dyn_first.output.@double
@ifchange = max_certainty
endFacets

Facets
@triplet = qm_prop_square.output.@double
@ifchange = max_certainty
endFacets

Facets
@triplet = qm_proportional.output.@double
@ifchange = max_certainty
endFacets
```

```
Facets
@triplet = tank.capacity.@float
@ifchange = max_certainty
endFacets

Facets
@triplet = tank.outlet_restriction.@float
@restriction = zero
endFacets

!*** LoadStrategy must go at the end of the Knowledge Base ***!
LoadStrategy
@name = "tnk1_qnx.stg"
EndLoadStrategy
```

APPENDIX D KNOWLEDGE BASE FOR THE ISCS PROTOTYPE

This appendix presents the source code of the knowledge base of the ISCS, as implemented in ProcessVision. The knowledge incorporated into ISCS, as in the case of every knowledge-based system implementation, is structured by the system at the time the reasoning process takes place during the actual application of the system. Thus, this knowledge is presented using the same format in which ProcessVision stores it.

```
Class
@name = mill
@object = sag_mill
@public = status.running, status.was_running, status.@float
endClass

Class
@name = ball_mill
@object = ball_mill_1, ball_mill_2
@superClass = mill
@public = status.running, status.was_running, status.@float
endClass

Class
@name = feeder
@object = feeder1, feeder2, feeder3, feeder4, feeder5
@public = contribution.@integer, real_speed.@float, speed_ratio_real.@float,
        status.running, status.was_running, status.@float,
        status_chute.plugged
endClass

Class
@name = model_dyn
endClass

Class
@name = process_var
@object = fresh_feed, hi_tons_in, hi_tons_out, sag_amps, sag_hydro, sag_speed
@public = agrees_with.setpoint, controller_name.@string, deviation.@float,
        diff_amt_pred.large, diff_amt_pred.medium, diff_amt_pred.small,
        diff_amt_pred.@float, is_approaching_to.setpoint, is_close_to.setpoint,
        pred_amt.high, pred_amt.low, pred_amt.medium,
        pred_amt.setpoint, pred_amt.@float, pred_amt_old.@float,
        pred_amt_ss.@float, pred_amt_status.@string, pred_dyn.constant,
        pred_dyn.decreasing, pred_dyn.increasing, pred_dyn.@float,
        pred_dyn_status.@string, qmodel_ss.@string, rate_of_change.@float,
        real_amt.high, real_amt.low, real_amt.medium,
        real_amt.positive, real_amt.zero, real_amt.@float,
        real_amt_max.@float, real_amt_min.@float, real_amt_old.@float,
        real_amt_status.@string, real_average.@float, real_dyn.constant,
        real_dyn.decreasing, real_dyn.increasing, real_dyn.@float,
        real_dyn_status.@string, real_setpoint.@float, std_amt.@float,
        std_setpoint.@float, time_constant.@float, time_s_state.@float
endClass

Class
@name = qmss_function
@object = mss_sag_speed
```

```
@public = input1.@double, input2.@double, input3.@double,
        input4.@double, output.@double
endClass

Class
@name = qmss_table3
@public = input1.@double, input_h.@float, input_l.@float,
        input_m.@float, output.@double, output_h.@float,
        output_l.@float, output_m.@float
endClass

Class
@name = restriction_category
@object = ball_mills, bm_pumps, feeders, flotation, other, plugged_chutes, sag_control, sag_shut_down, soft_ore, tailings
@private = restriction.exists
@public = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double
endClass

Class
@name = restriction_id
@object = rr_feeder_down, rr_ffeed_cut, rr_ffeed_limit, rr_ore_supply, rr_sag_stops, rr_soft_ore, rr_speed, rr_water_ratio
@public = cause.ball_mills, cause.bm_pumps, cause.flotation,
        cause.other, cause.plugged_chutes, cause.sag_control,
        cause.sag_shut_down, cause.soft_ore, cause.tailings,
        hour_elapsed.@integer, lost_current_rr.@double, lost_current_shift.@double,
        lost_last_shift.@double, minute_elapsed.@integer, possible_cause.determined,
        real_accumulated_losses.@double, restriction.detected, restriction.detected_past_sampling,
        restriction.existed, restriction.exists, restriction.has_been_identified,
        restriction.ignore, restriction.initialized, restriction.@integer,
        second_elapsed.@integer, time_delta_elapsed.@float, time_elapsed.@float,
        time_final.@time, time_initial.@time, time_total.@float
endClass

Object
@name = a
@attribute = a.@float, a1.@string, aa.@float,
        b.@float, b_1.@float, c.@float,
        c1.@integer, d.@float, param.initialized,
        rr.initialized, time_elapsed.@float, time_init.@time,
        time_now.@time
endObject

Object
@name = analyze
@attribute = simulation.results
endObject

Object
@name = ball_mill_1
@class = ball_mill
@attribute = status.running, status.was_running, status.@float
endObject

Object
@name = ball_mill_2
@class = ball_mill
@attribute = status.running, status.was_running, status.@float
endObject

Object
```

```
@name = ball_mills
@class = restriction_category
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists
endObject

Object
@name = bm_pumps
@class = restriction_category
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists
endObject

Object
@name = category_all
@attribute = current_restriction.@string, percent_accumulated_losses.@float, real_accumulated_losses.@double,
        total_lost_current_shift.@double, total_lost_last_shift.@double, user_chose.new_cause
endObject

Object
@name = category_any
@attribute = current_category.@string
endObject

Object
@name = chute_conveyor
@attribute = status_chute.plugged
endObject

Object
@name = chute_feed
@attribute = status_chute.plugged
endObject

Object
@name = chutes
@attribute = status.plugged
endObject

Object
@name = coarse_ratio
@attribute = pred_amt.@float, real_setpoint.@float
endObject

Object
@name = convert
@attribute = variable.@string
endObject

Object
@name = conveyor_3c
@attribute = status.running
endObject

Object
@name = conveyor_4c
@attribute = status.running
endObject

Object
@name = dummy
```

```
@attribute = a.@string, a.@float, adjustment.@float,
        ball_mill.@string, begin_average.@float, category.@string,
        convert.@string, current_loss.@double, difference.@float,
        double.@double, end_average.@float, equipment.@string,
        eval_pred.@double, expo.@double, feeder.@string,
        final_average.@float, floating.@float, init_average.@float,
        integer.@integer, k1.@double, k2.@double,
        k3.@double, k4.@double, losses.@double,
        macro.@string, model.@string, model_name.@string,
        model_slope.@double, name.@string, new_cause.@string,
        number_of_rr.@integer, pred_level.@double, procedure.@string,
        process_var.@string, reasoning.@string, rectangle.@double,
        remainder.@float, restriction.exists, restriction.@string,
        rotate.@string, rr_analyzed.@integer, rr_being_adjusted.@float,
        rr_time_acc.@integer, std_amt.@float, string.@string,
        time1.@time, time2.@float, time_accumulated.@float,
        time_initial.@time, time_previous.@float, timex.@time,
        triangle.@double, var.@string
endObject

Object
@name = end
@attribute = of.shift
endObject

Object
@name = feeder1
@class = feeder
@attribute = chute.plugged, contribution.@integer, real_speed.@float,
        speed_ratio_real.@float, status.running, status.was_running,
        status.@float, status_chute.plugged
endObject

Object
@name = feeder2
@class = feeder
@attribute = chute.plugged, contribution.@integer, real_speed.@float,
        speed_ratio_real.@float, status.running, status.was_running,
        status.@float, status_chute.plugged
endObject

Object
@name = feeder3
@class = feeder
@attribute = chute.plugged, contribution.@integer, real_speed.@float,
        speed_ratio_real.@float, status.running, status.was_running,
        status.@float, status_chute.plugged
endObject

Object
@name = feeder4
@class = feeder
@attribute = chute.plugged, contribution.@integer, real_speed.@float,
        speed_ratio_real.@float, status.running, status.was_running,
        status.@float, status_chute.plugged
endObject

Object
@name = feeder5
@class = feeder
@attribute = chute.plugged, contribution.@integer, real_speed.@float,
```

```
        speed_ratio_real.@float, status.running, status.was_running,
        status.@float, status_chute.plugged
endObject

Object
@name = feeder_set
@attribute = coarse_contribution.@integer, current_restriction.exists, fine_contribution.@integer,
        status_all.running, total_contribution.@integer
endObject

Object
@name = feeders
@class = restriction_category
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists,
        total_lost_current_shift.@double
endObject

Object
@name = flotation
@class = restriction_category
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists,
        total_lost_current_shift.@double
endObject

Object
@name = fresh_feed
@class = process_var
@attribute = agrees_with.setpoint, controller.meets_demand, controller_mode.cascade,
        controller_name.@string, cut_time.initialized, cut_time_elapsed.@float,
        cut_time_final.@time, cut_time_init.@time, cut_time_total.@double,
        deviation.@float, diff_amt_pred.large, diff_amt_pred.medium,
        diff_amt_pred.small, diff_amt_pred.@float, follows.setpoint,
        has_been.cut, has_been.recovered_from_cut, hi_limit_real.@float,
        hi_limit_target.@float, is_approaching_to.setpoint, is_close_to.setpoint,
        pred_amt.high, pred_amt.low, pred_amt.medium,
        pred_amt.setpoint, pred_amt.@float, pred_amt_old.@float,
        pred_amt_ss.@float, pred_amt_status.@string, pred_dyn.constant,
        pred_dyn.decreasing, pred_dyn.increasing, pred_dyn.@float,
        pred_dyn_status.@string, qmodel_ss.@string, rate_of_change.@float,
        real_amt.high, real_amt.low, real_amt.medium,
        real_amt.positive, real_amt.zero, real_amt.@float,
        real_amt_max.@float, real_amt_min.@float, real_amt_old.@float,
        real_amt_status.@string, real_average.@float, real_dyn.constant,
        real_dyn.decreasing, real_dyn.increasing, real_dyn.@float,
        real_dyn_status.@string, real_setpoint.@float, std_amt.@float,
        std_dyn.@float, std_setpoint.@float, time_constant.@float,
        time_s_state.@float, tonnage_difference.@float, was.cut
endObject

Object
@name = fresh_feed_controller
@attribute = operation.follows_setpoint
endObject

Object
@name = hi_tons
@attribute = target_hi_limit.@float
endObject
```

```
Object
@name = hi_tons_in
@class = process_var
@attribute = agrees_with.setpoint, close_to.setpoint, controller_name.@string,
        deviation.@float, diff_amt_pred.large, diff_amt_pred.medium,
        diff_amt_pred.small, diff_amt_pred.@float, is_approaching_to.setpoint,
        is_close_to.setpoint, pred_amt.high, pred_amt.low,
        pred_amt.medium, pred_amt.setpoint, pred_amt.@float,
        pred_amt_old.@float, pred_amt_ss.@float, pred_amt_status.@string,
        pred_dyn.constant, pred_dyn.decreasing, pred_dyn.increasing,
        pred_dyn.@float, pred_dyn_status.@string, qmodel_ss.@string,
        rate_of_change.@float, real_amt.high, real_amt.low,
        real_amt.medium, real_amt.positive, real_amt.zero,
        real_amt.@float, real_amt_max.@float, real_amt_min.@float,
        real_amt_old.@float, real_amt_status.@string, real_average.@float,
        real_dyn.constant, real_dyn.decreasing, real_dyn.increasing,
        real_dyn.@float, real_dyn_status.@string, real_setpoint.@float,
        setpoint.@float, std_amt.@float, std_setpoint.@float,
        time_constant.@float, time_s_state.@float
endObject

Object
@name = hi_tons_out
@class = process_var
@attribute = agrees_with.setpoint, close_to.setpoint, controller_name.@string,
        deviation.@float, diff_amt_pred.large, diff_amt_pred.medium,
        diff_amt_pred.small, diff_amt_pred.@float, is_approaching_to.setpoint,
        is_close_to.setpoint, pred_amt.high, pred_amt.low,
        pred_amt.medium, pred_amt.setpoint, pred_amt.@float,
        pred_amt_old.@float, pred_amt_ss.@float, pred_amt_status.@string,
        pred_dyn.constant, pred_dyn.decreasing, pred_dyn.increasing,
        pred_dyn.@float, pred_dyn_status.@string, qmodel_ss.@string,
        rate_of_change.@float, real_amt.high, real_amt.low,
        real_amt.medium, real_amt.positive, real_amt.zero,
        real_amt.@float, real_amt_max.@float, real_amt_min.@float,
        real_amt_old.@float, real_amt_status.@string, real_average.@float,
        real_dyn.constant, real_dyn.decreasing, real_dyn.increasing,
        real_dyn.@float, real_dyn_status.@string, real_setpoint.@float,
        setpoint.@float, std_amt.@float, std_setpoint.@float,
        time_constant.@float, time_s_state.@float
endObject

Object
@name = iss_user
@attribute = accumulate_current_rr.give_warning, accumulate_tonnage_lost.give_warning, do_not_correct.give_warning,
        do_not_correct.warning_given
endObject

Object
@name = level_tailing_box
@attribute = real_amt.high, real_amt.@float
endObject

Object
@name = mss_sag_speed
@class = qmss_function
@attribute = input.@double, input1.@double, input2.@double,
        input3.@double, input4.@double, input_h.@float,
        input_l.@float, input_m.@float, output.@double,
        output_h.@float, output_l.@float, output_m.@float
endObject
```

```
Object
@name = other
@class = restriction_category
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists,
        total_lost_current_shift.@double
endObject

Object
@name = plugged_chutes
@class = restriction_category
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists,
        total_lost_current_shift.@double
endObject

Object
@name = power_to_hydro
@attribute = selector.power
endObject

Object
@name = qm_dyn_first
@attribute = dummy.@double, output.@double, output_old.@double,
        output_ss.@double, time_constant.@float, time_s_state.@float
endObject

Object
@name = qmss_table3
@attribute = input.@double, input1.@double, input_h.@float,
        input_l.@float, input_m.@float, output.@double,
        output_h.@float, output_l.@float, output_m.@float,
        slope.@double
endObject

Object
@name = rr_adjustment
@attribute = accumulated_tonnage.@double, accumulated_tonnage_1.@double, accumulated_tonnage_2.@double,
        accumulated_tonnage_3.@double, accumulated_tonnage_4.@double, accumulated_tonnage_5.@double,
        accumulated_tonnage_6.@double, accumulated_tonnage_7.@double, accumulated_tonnage_8.@double,
        accumulated_tonnage_9.@double, accumulated_tonnage_previous.@double, cause_1.@string,
        cause_2.@string, cause_3.@string, cause_4.@string,
        cause_5.@string, cause_6.@string, cause_7.@string,
        cause_8.@string, cause_9.@string, change_in_final_tonnage.@float,
        change_in_initial_tonnage.@float, change_in_tonnage.@float, change_in_tonnage_previous.@float,
        correction_tons_accumulated.@double, did_not_correct.tonnage_lost, do_not_correct.tonnage_lost,
        do_not_correct.warning_given, do_not_correct.warning_sent, final_tonnage.in_progress,
        final_tonnage.was_in_progress, final_tonnage.@float, ignore_tonnage.correction,
        initial_tonnage.@float, maximum_time.@float, mechanism.initialized,
        new_initial_tonnage.@float, number_of_rr.@integer, pred_tonnage_during_rr.@float,
        process.initialized, restriction_1.@string, restriction_2.@string,
        restriction_3.@string, restriction_4.@string, restriction_5.@string,
        restriction_6.@string, restriction_7.@string, restriction_8.@string,
        restriction_9.@string, rr_being_adjusted.@integer, time_elapsed_1.@float,
        time_elapsed_2.@float, time_elapsed_3.@float, time_elapsed_4.@float,
        time_elapsed_5.@float, time_elapsed_6.@float, time_elapsed_7.@float,
        time_elapsed_8.@float, time_elapsed_9.@float, time_elapsed_total.@float
endObject

Object
```

@name = rr_all
@attribute = real_accumulated_losses.@double, time_detected.@integer, time_rr_estimate.@float,
time_rr_exists.@float, time_tons_average.@float, tons_average.@float,
total_lost_current_shift.@double, total_lost_last_shift.@double
endObject

Object
@name = rr_any
@attribute = accumulate_losses.rr_sag_stops, begin_average.@float, current_restriction.@string,
do_not_accumulate.tonnage_current_rr, do_not_accumulate.tonnage_lost, end_average.@float,
first_restriction.appeared, follow.up, last_restriction.disappeared,
minimum_time_to_estimate.@float, pred_tonnage_during_rr.@float, restriction.existed,
restriction.exists, stop_accumulating.tonnage_lost, stops_accumulating.tonnage_lost,
time_first_rr_appeared.@float, time_last_rr_disappeared.@float, time_no_exists_restriction.@float,
time_no_exists_restrictions.@float
endObject

Object
@name = rr_dummy
@attribute = rectangle.@double
endObject

Object
@name = rr_feeder_down
@class = restriction_id
@attribute = cause.ball_mills, cause.bm_pumps, cause.flotation,
cause.other, cause.plugged_chutes, cause.sag_control,
cause.sag_shut_down, cause.soft_ore, cause.tailings,
hour_elapsed.@integer, lost_current_rr.@double, lost_current_shift.@double,
lost_last_shift.@double, minute_elapsed.@integer, possible_cause.determined,
possible_cause.feeders, possible_cause.plugged_chutes, real_accumulated_losses.@double,
restriction.detected, restriction.detected_past_sampling, restriction.existed,
restriction.exists, restriction.has_been_identified, restriction.has_been_initialized,
restriction.identify, restriction.identify_cause, restriction.ignore,
restriction.initialized, restriction.please_identify, restriction.@integer,
second_elapsed.@integer, time_delta_elapsed.@float, time_elapsed.@float,
time_elapsedx.@integer, time_final.@time, time_initial.@time,
time_total.@float
endObject

Object
@name = rr_ffeed_cut
@class = restriction_id
@attribute = cause.ball_mills, cause.bm_pumps, cause.flotation,
cause.other, cause.plugged_chutes, cause.sag_control,
cause.sag_shut_down, cause.soft_ore, cause.tailings,
hour_elapsed.@integer, lost_current_rr.@double, lost_current_shift.@double,
lost_last_shift.@double, minute_elapsed.@integer, possible_cause.ball_mills,
possible_cause.bm_pumps, possible_cause.determined, possible_cause.flotation_operation,
possible_cause.other, possible_cause.sag_control, real_accumulated_losses.@double,
restriction.detected, restriction.detected_past_sampling, restriction.existed,
restriction.exists, restriction.has_been_identified, restriction.has_been_initialized,
restriction.identify, restriction.identify_cause, restriction.ignore,
restriction.initialized, restriction.please_identify, restriction.@integer,
second_elapsed.@integer, time_delta_elapsed.@float, time_elapsed.@float,
time_elapsedx.@integer, time_final.@time, time_initial.@time,
time_total.@float
endObject

Object
@name = rr_ffeed_limit

@class = restriction_id
@attribute = cause.ball_mills, cause.bm_pumps, cause.flotation,
cause.other, cause.plugged_chutes, cause.sag_control,
cause.sag_shut_down, cause.soft_ore, cause.tailings,
hour_elapsed.@integer, lost_current_rr.@double, lost_current_shift.@double,
lost_last_shift.@double, minute_elapsed.@integer, possible_cause.ball_mills,
possible_cause.bm_pumps, possible_cause.determined, possible_cause.flotation_operation,
possible_cause.other, possible_cause.others, possible_cause.tailings,
real_accumulated_losses.@double, restriction.detected, restriction.detected_past_sampling,
restriction.existed, restriction.exists, restriction.has_been_identified,
restriction.has_been_initialized, restriction.identify, restriction.identify_cause,
restriction.ignore, restriction.initialized, restriction.please_identify,
restriction.@integer, second_elapsed.@integer, time_delta_elapsed.@float,
time_elapsed.@float, time_elapsedx.@integer, time_final.@time,
time_initial.@time, time_total.@float
endObject

Object
@name = rr_ore_supply
@class = restriction_id
@attribute = cause.ball_mills, cause.bm_pumps, cause.flotation,
cause.other, cause.plugged_chutes, cause.sag_control,
cause.sag_shut_down, cause.soft_ore, cause.tailings,
hour_elapsed.@integer, lost_current_rr.@double, lost_current_shift.@double,
lost_last_shift.@double, minute_elapsed.@integer, possible_cause.determined,
possible_cause.feeders, possible_cause.other, possible_cause.others,
possible_cause.plugged_chutes, real_accumulated_losses.@double, restriction.detected,
restriction.detected_past_sampling, restriction.existed, restriction.exists,
restriction.has_been_identified, restriction.has_been_initialized, restriction.identify,
restriction.identify_cause, restriction.ignore, restriction.initialized,
restriction.please_identify, restriction.@integer, second_elapsed.@integer,
time_delta_elapsed.@float, time_elapsed.@float, time_elapsedx.@integer,
time_final.@time, time_initial.@time, time_total.@float
endObject

Object
@name = rr_sag_stops@class = restriction_id
@attribute = cause.ball_mills, cause.bm_pumps, cause.flotation,
cause.other, cause.plugged_chutes, cause.sag_control,
cause.sag_shut_down, cause.soft_ore, cause.tailings,
hour_elapsed.@integer, lost_current_rr.@double, lost_current_shift.@double,
lost_last_shift.@double, minute_elapsed.@integer, possible_cause.ball_mills,
possible_cause.bm_pumps, possible_cause.determined, possible_cause.sag_control,
possible_cause.sag_liners, possible_cause.sag_mill, possible_cause.sag_pump,
possible_cause.sag_shut_down, real_accumulated_losses.@double, restriction.detect,
restriction.detected, restriction.detected_past_sampling, restriction.existed,
restriction.exists, restriction.has_been_identified, restriction.has_been_initialized,
restriction.identify, restriction.identify_cause, restriction.ignore,
restriction.initialized, restriction.please_identify, restriction.@integer,
restrictioon_cause.determined, second_elapsed.@integer, time_delta_elapsed.@float,
time_elapsed.@float, time_elapsedx.@integer, time_final.@time,
time_initial.@time, time_total.@float
endObject

Object
@name = rr_soft_ore
@class = restriction_id
@attribute = cause.ball_mills, cause.bm_pumps, cause.flotation,
cause.other, cause.plugged_chutes, cause.sag_control,
cause.sag_shut_down, cause.soft_ore, cause.tailings,
hour_elapsed.@integer, lost_current_rr.@double, lost_current_shift.@double,

```
        lost_last_shift.@double, minute_elapsed.@integer, possible_cause.determined,
        possible_cause.soft_ore, possible_cause.tailings, real_accumulated_losses.@double,
        restriction.detected, restriction.detected_past_sampling, restriction.existed,
        restriction.exists, restriction.has_been_identified, restriction.has_been_initialized,
        restriction.identify, restriction.identify_cause, restriction.ignore,
        restriction.initialized, restriction.please_identify, restriction.@integer,
        second_elapsed.@integer, time_delta_elapsed.@float, time_elapsed.@float,
        time_elapsedx.@integer, time_final.@time, time_initial.@time,
        time_total.@float
endObject

Object
@name = rr_speed
@class = restriction_id
@attribute = cause.ball_mills, cause.bm_pumps, cause.flotation,
        cause.other, cause.plugged_chutes, cause.sag_control,
        cause.sag_liners, cause.sag_shut_down, cause.soft_ore,
        cause.tailings, hour_elapsed.@integer, lost_current_rr.@double,
        lost_current_shift.@double, lost_last_shift.@double, minute_elapsed.@integer,
        possible_cause.determined, real_accumulated_losses.@double, restriction.detected,
        restriction.detected_past_sampling, restriction.existed, restriction.exists,
        restriction.has_been_identified, restriction.has_been_initialized, restriction.identify,
        restriction.identify_cause, restriction.ignore, restriction.initialized,
        restriction.@integer, second_elapsed.@integer, time_delta_elapsed.@float,
        time_elapsed.@float, time_elapsedx.@integer, time_final.@time,
        time_initial.@time, time_total.@float
endObject

Object
@name = rr_water_ratio
@class = restriction_id
@attribute = cause.ball_mills, cause.bm_pumps, cause.flotation,
        cause.other, cause.plugged_chutes, cause.sag_control,
        cause.sag_liners, cause.sag_shut_down, cause.soft_ore,
        cause.tailings, hour_elapsed.@integer, lost_current_rr.@double,
        lost_current_shift.@double, lost_last_shift.@double, minute_elapsed.@integer,
        possible_cause.determined, possible_cause.sag_control, real_accumulated_losses.@double,
        restriction.detected, restriction.detected_past_sampling, restriction.existed,
        restriction.exists, restriction.has_been_identified, restriction.has_been_initialized,
        restriction.identify, restriction.identify_cause, restriction.ignore,
        restriction.initialized, restriction.please_identify, restriction.@integer,
        second_elapsed.@integer, time_delta_elapsed.@float, time_elapsed.@float,
        time_elapsedx.@integer, time_final.@time, time_initial.@time,
        time_total.@float
endObject

Object
@name = runx
@attribute = simulation.again
endObject

Object
@name = sag_amps
@class = process_var
@attribute = agrees_with.setpoint, controller_name.@string, deviation.@float,
        diff_amt_pred.large, diff_amt_pred.medium, diff_amt_pred.small,
        diff_amt_pred.@float, is_approaching_to.setpoint, is_close_to.setpoint,
        pred_amt.high, pred_amt.low, pred_amt.medium,
        pred_amt.setpoint, pred_amt.@float, pred_amt_old.@float,
        pred_amt_ss.@float, pred_amt_status.@string, pred_dyn.constant,
        pred_dyn.decreasing, pred_dyn.increasing, pred_dyn.@float,
```

```
        pred_dyn_status.@string, qm_param1.@float, qm_param_original.@float,
        qmodel_ss.@string, rate_of_change.@float, real_amt.high,
        real_amt.low, real_amt.medium, real_amt.positive,
        real_amt.zero, real_amt.@float, real_amt_max.@float,
        real_amt_min.@float, real_amt_old.@float, real_amt_status.@string,
        real_average.@float, real_dyn.constant, real_dyn.decreasing,
        real_dyn.increasing, real_dyn.@float, real_dyn_status.@string,
        real_setpoint.@float, std_amt.@float, std_setpoint.@float,
        time_constant.@float, time_s_state.@float
endObject

Object
@name = sag_ball_load
@attribute = real_amt.@float
endObject

Object
@name = sag_control
@class = restriction_category
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists,
        total_lost_current_shift.@double
endObject

Object
@name = sag_hydro
@class = process_var
@attribute = agrees_with.setpoint, controller_name.@string, deviation.@float,
        diff_amt_pred.large, diff_amt_pred.medium, diff_amt_pred.small,
        diff_amt_pred.@float, hi_limit_real.@float, hi_limit_target.@float,
        is_approaching_to.setpoint, is_close_to.setpoint, pred_amt.high,
        pred_amt.low, pred_amt.medium, pred_amt.setpoint,
        pred_amt.@float, pred_amt_old.@float, pred_amt_ss.@float,
        pred_amt_status.@string, pred_dyn.constant, pred_dyn.decreasing,
        pred_dyn.increasing, pred_dyn.@float, pred_dyn_status.@string,
        qmodel_ss.@string, rate_of_change.@float, real_amt.high,
        real_amt.low, real_amt.medium, real_amt.positive,
        real_amt.zero, real_amt.@float, real_amt_max.@float,
        real_amt_min.@float, real_amt_old.@float, real_amt_status.@string,
        real_average.@float, real_dyn.constant, real_dyn.decreasing,
        real_dyn.increasing, real_dyn.@float, real_dyn_status.@string,
        real_setpoint.@float, std_amt.@float, std_setpoint.@float,
        time_constant.@float, time_s_state.@float
endObject

Object
@name = sag_mill
@class = mill
@attribute = model_name.@string, status.running, status.was_running,
        status.@float
endObject

Object
@name = sag_power
@attribute = amps_amt.@float, model_parameter.@float, real_amt.@float,
        real_setpoint.@float
endObject

Object
@name = sag_shut_down
@class = restriction_category
```

```
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists
endObject

Object
@name = sag_speed
@class = process_var
@attribute = agrees_with.setpoint, controller_name.@string, deviation.@float,
        diff_amt_pred.large, diff_amt_pred.medium, diff_amt_pred.small,
        diff_amt_pred.@float, is_approaching_to.setpoint, is_close_to.setpoint,
        pred_amt.high, pred_amt.low, pred_amt.medium,
        pred_amt.setpoint, pred_amt.@float, pred_amt_old.@float,
        pred_amt_ss.@float, pred_amt_status.@string, pred_dyn.constant,
        pred_dyn.decreasing, pred_dyn.increasing, pred_dyn.@float,
        pred_dyn_status.@string, qmodel_ss.@string, rate_of_change.@float,
        real_amt.high, real_amt.low, real_amt.medium,
        real_amt.positive, real_amt.zero, real_amt.@float,
        real_amt_max.@float, real_amt_min.@float, real_amt_old.@float,
        real_amt_status.@string, real_average.@float, real_dyn.constant,
        real_dyn.decreasing, real_dyn.increasing, real_dyn.@float,
        real_dyn_status.@string, real_setpoint.@float, std_amt.@float,
        std_setpoint.@float, time_constant.@float, time_s_state.@float
endObject

Object
@name = sag_water
@attribute = real_amt.@float
endObject

Object
@name = sag_water_ratio
@attribute = real_amt.@float
endObject

Object
@name = soft_ore
@class = restriction_category
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists,
        total_lost_current_shift.@double
endObject

Object
@name = start
@attribute = simulation.yes
endObject

Object
@name = stop
@attribute = simulation.yes
endObject

Object
@name = system
@attribute = already.configured
endObject

Object
@name = SystemClock
@attribute = style_5.@string
endObject
```

```
Object
@name = tail_box1
@attribute = level.high, real_amt.@float
endObject

Object
@name = tail_box2
@attribute = level.high, real_amt.@float
endObject

Object
@name = tailings
@class = restriction_category
@attribute = current_restriction.exists, lost_current_shift.@double, lost_last_shift.@double,
        percent_accumulated_losses.@float, real_accumulated_losses.@double, restriction.exists,
        total_lost_current_shift.@double
endObject

Object
@name = timex
@attribute = current_timex.@time, elapsed.@float, elapsed_old.@float,
        elapsed_time.@float, last_sampling.instant, sampling.@float,
        sampling_time.@float, since_last_sampling.@integer, sleep.@float,
        timex.@string
endObject

Object
@name = water_ratio
@attribute = pred_amt.@float, real_amt.@float
endObject

Procedure
@name = QM_simulator_coordinator
@do =
 RUN_PROCEDURE ( "convert_all_measurements" )
 RUN_PROCEDURE ( "run_model_ss" )
 RUN_PROCEDURE ( "run_model_dyn" )
 RUN_PROCEDURE ( "obtain_predicted_dynamics" )
 RUN_PROCEDURE ( "rotate_all_predictions" )
 MACRO ( "qual_status*" )
endProcedure

Procedure
@name = amps_deviation
@do =
 sag_amps.deviation.@float  = sag_amps.real_setpoint.@float  - sag_amps.real_amt.@float
endProcedure

Procedure
@name = check_equipment_status_coordinator
@do =
 dummy.equipment.@string is "ball_mill_1"
 FREERULE ( $Rule, "check_equip_status*" )
 MACRO ( "check_equip_status*" )
 dummy.equipment.@string is "ball_mill_2"
 FREERULE ( $Rule, "check_equip_status*" )
 MACRO ( "check_equip_status*" )
 dummy.equipment.@string is "feeder1"
 FREERULE ( $Rule, "check_equip_status*" )
 MACRO ( "check_equip_status*" )
```

```
 dummy.equipment.@string is "feeder5"
 FREERULE ( $Rule, "check_equip_status*" )
 MACRO ( "check_equip_status*" )
endProcedure

Procedure
@name = check_ffeed_controller
@do =
 STRCOPY ( dummy.process_var.@string, "fresh_feed" )
 FREERULE ( $Rule, "check_controllers*" )
 MACRO ( "check_controllers_*" )
 MACRO ( "check_controller_ffeed" )
endProcedure

Procedure
@name = choose_ball_mills
@do =
 dummy.new_cause.@string is "ball_mills"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure

Procedure
@name = choose_bm_pumps
@do =
 dummy.new_cause.@string is "bm_pumps"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure

Procedure
@name = choose_feeders
@do =
 dummy.new_cause.@string is "feeders"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure

Procedure
@name = choose_flotation
@do =
 dummy.new_cause.@string is "flotation"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure

Procedure
@name = choose_other
@do =
 dummy.new_cause.@string is "other"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure

Procedure
@name = choose_plugged_chutes
@do =
 dummy.new_cause.@string is "plugged_chutes"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure
```

```
Procedure
@name = choose_sag_control
@do =
 dummy.new_cause.@string is "sag_control"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure

Procedure
@name = choose_sag_liners
@do =
 dummy.new_cause.@string is "sag_liners"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure

Procedure
@name = choose_soft_ore
@do =
 dummy.new_cause.@string is "soft_ore"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure

Procedure
@name = choose_tailings
@do =
 dummy.new_cause.@string is "tailings"
 FREERULE ( $Rule, "choose_one_cause" )
 MACRO ( "choose_one_cause" )
endProcedure

Procedure
@name = coarse_ratio_prediction
@do =
 feeder_set.coarse_contribution.@integer = feeder1.contribution.@integer + feeder5.contribution.@integer
 feeder_set.fine_contribution.@integer = feeder2.contribution.@integer + feeder3.contribution.@integer +
feeder4.contribution.@integer
 coarse_ratio.pred_amt.@float  = 100.000000 * feeder_set.coarse_contribution.@integer / (
feeder_set.coarse_contribution.@integer + feeder_set.fine_contribution.@integer )
endProcedure

Procedure
@name = controller_name
@do =
 ?all_vars = CLASSTOLIST ( "process_var" )
 ALL ( ?all_vars.controller_name.@string is "" )
 fresh_feed.controller_name.@string is "ffeed_controller"
endProcedure

Procedure
@name = convert_all_measurements
@do =
 STRCOPY ( dummy.convert.@string, "fresh_feed" )
 RUN_PROCEDURE ( "convert_real_var" )
 STRCOPY ( dummy.convert.@string, "sag_amps" )
 RUN_PROCEDURE ( "convert_real_var" )
endProcedure

Procedure
```

```
@name = convert_all_setpoints
@do =
 STRCOPY ( dummy.convert.@string, "fresh_feed" )
 RUN_PROCEDURE ( "convert_setpoints" )
endProcedure

Procedure
@name = convert_real_var
@do =
 <dummy.convert.@string>.std_amt.@float  = ( <dummy.convert.@string>.real_amt.@float  -
<dummy.convert.@string>.real_amt_min.@float  ) / ( <dummy.convert.@string>.real_amt_max.@float  -
<dummy.convert.@string>.real_amt_min.@float  ) * 100
endProcedure

Procedure
@name = convert_setpoints
@do =
 <dummy.convert.@string>.std_setpoint.@float  = ( <dummy.convert.@string>.real_setpoint.@float  -
<dummy.convert.@string>.real_amt_min.@float  ) / ( <dummy.convert.@string>.real_amt_max.@float  -
<dummy.convert.@string>.real_amt_min.@float  ) * 100
endProcedure

Procedure
@name = determine_rr_causes
@do =
 ?current_rr = ANY ( {restriction_id}.restriction.exists is TRUE )
 ALL ( MACRO ( ?current_rr.restriction_name.@string ) )
endProcedure

Procedure
@name = disable_accumulate_current_rr
@do =
 rr_any.do_not_accumulate.tonnage_current_rr is TRUE
 rr_adjustment.ignore_tonnage.correction is TRUE
 iss_user.accumulate_current_rr.give_warning is TRUE
 TEXT ( "USER:  DISABLE tonnage accumulation for current restriction.", "rr-user" )
 TEXT ( "USER:  DISABLE tonnage accumulation for current restriction.", "restrictions" )
endProcedure

Procedure
@name = disable_accumulation_tons_lost
@do =
 rr_any.stop_accumulating.tonnage_lost is TRUE
 iss_user.accumulate_tonnage_lost.give_warning is TRUE
 TEXT ( "USER:  DISABLE mechanism for accumulation of tonnage lost", "rr-user" )
 TEXT ( "USER:  DISABLE mechanism for accumulation of tonnage lost", "restrictions" )
endProcedure

Procedure
@name = disable_correct_tonnage_lost
@do =
 rr_adjustment.do_not_correct.tonnage_lost is TRUE
 iss_user.do_not_correct.give_warning is TRUE
 TEXT ( "USER:  DISABLE correction of tonnage lost estimated.", "rr-user" )
 TEXT ( "USER:  DISABLE correction of tonnage lost estimated.", "restrictions" )
endProcedure

Procedure
@name = disable_current_rr
@do =
 rr_any.do_not_accumulate.tonnage_current_rr is TRUE
```

```
 iss_user.accumulate_current_rr.give_warning is TRUE
endProcedure

Procedure
@name = dynamics_predicted
@do =
 <dummy.var.@string>.pred_dyn.@float  = RATEOFCHANGE ( <dummy.var.@string>.pred_amt.@float , 0,
<dummy.var.@string>.rate_of_change.@float  ) * 60
endProcedure

Procedure
@name = dynamics_real
@do =
 <dummy.var.@string>.real_dyn.@float  = RATEOFCHANGE ( <dummy.var.@string>.real_amt.@float , 0,
<dummy.var.@string>.rate_of_change.@float  ) * 60
endProcedure

Procedure
@name = enable_accumulate_current_rr
@do =
 rr_any.do_not_accumulate.tonnage_current_rr is FALSE
 rr_adjustment.ignore_tonnage.correction is FALSE
 iss_user.accumulate_current_rr.give_warning is TRUE
 TEXT ( "USER:  ENABLE tonnage accumulation for current restriction.", "rr-user" )
 TEXT ( "USER:  ENABLE tonnage accumulation for current restriction.", "restrictions" )
endProcedure

Procedure
@name = enable_accumulation_tons_lost
@do =
 rr_any.stop_accumulating.tonnage_lost is FALSE
 iss_user.accumulate_tonnage_lost.give_warning is TRUE
 TEXT ( "USER:  ENABLE mechanisms for accumulation of tonnage lost.", "rr-user" )
 TEXT ( "USER:  ENABLE mechanisms for accumulation of tonnage lost.", "restricitons" )
endProcedure

Procedure
@name = enable_correct_tonnage_lost
@do =
 rr_adjustment.do_not_correct.tonnage_lost is FALSE
 iss_user.do_not_correct.give_warning is TRUE
 TEXT ( "USER:  ENABLE correction of tonnage lost estimated.", "rr-user" )
 TEXT ( "USER:  ENABLE correction of tonnage lost estimated.", "restrictions" )
endProcedure

Procedure
@name = end_of_shift
@do =
 end.of.shift is TRUE
endProcedure

Procedure
@name = initial_equipment_status
@do =
 ball_mill_1.status.@float  = 1
 ball_mill_2.status.@float  = 1
 sag_mill.status.@float  = 1
 chutes.status.plugged is FALSE
 PUTPOINTDATA ( "ball_mill_1.status.@float", ball_mill_1.status.@float , 100 )
 PUTPOINTDATA ( "ball_mill_2.status.@float", ball_mill_2.status.@float , 100 )
 PUTPOINTDATA ( "sag_mill.status.@float", sag_mill.status.@float , 100 )
```

```
 TEXT ( " initializing equipment again .... ", "developer" )
endProcedure

Procedure
@name = initial_feeder_status
@do =
 feeder1.status.@float  = 1
 feeder2.status.@float  = 1
 feeder3.status.@float  = 1
 feeder4.status.@float  = 1
 feeder5.status.@float  = 1
 coarse_ratio.real_setpoint.@float  = 2.000000 / 5.000000 * 100
 PUTPOINTDATA ( "feeder1.status.@float", feeder1.status.@float , 100 )
 PUTPOINTDATA ( "feeder2.status.@float", feeder2.status.@float , 100 )
 PUTPOINTDATA ( "feeder3.status.@float", feeder3.status.@float , 100 )
 PUTPOINTDATA ( "feeder4.status.@float", feeder4.status.@float , 100 )
 PUTPOINTDATA ( "feeder5.status.@float", feeder5.status.@float , 100 )
endProcedure

Procedure
@name = initial_values
@do =
 end.of.shift is FALSE
 timex.sampling_time.@float  = 30
 timex.last_sampling.instant is TRUE
 RUN_PROCEDURE ( "initialize_variables" )
 RUN_PROCEDURE ( "initialize_models" )
 RUN_PROCEDURE ( "initialize_rr_categories" )
 RUN_PROCEDURE ( "initialize_restrictions" )
 RUN_PROCEDURE ( "initialize_tonnage_adjustment" )
 RUN_PROCEDURE ( "initialize_user_settings" )
 RUN_PROCEDURE ( "x_initial_equipment_status" )
 RUN_PROCEDURE ( "x_initial_feeder_status" )
 RUN_PROCEDURE ( "mss_sag_speed_definition" )
endProcedure

Procedure
@name = initialize_models
@do =
 sag_power.model_parameter.@float  = 1750.000000 / 6300.000000
 sag_mill.model_name.@string is "mss_sag_power"
 sag_amps.qm_param_original.@float  = 6300.000000 / 1750.000000
 sag_amps.time_constant.@float  = 30.000000
endProcedure

Procedure
@name = initialize_restrictions
@do =
 ?restriction = CLASSTOLIST ( "restriction_id" )
 ALL ( ?restriction.restriction.initialized is FALSE )
 ALL ( ?restriction.restriction.existed is FALSE )
 ALL ( ?restriction.possible_cause.determined is FALSE )
 ALL ( ?restriction.restriction.detected is FALSE )
 ALL ( ?restriction.restriction.exists is FALSE )
 ALL ( ?restriction.time_elapsed.@float  = 0.000000 )
 ALL ( ?restriction.lost_current_rr.@double  = 0.000000 )
 ALL ( ?restriction.lost_current_shift.@double  = 0.000000 )
 ALL ( ?restriction.lost_last_shift.@double  = 0.000000 )
 ALL ( ?restriction.real_accumulated_losses.@double  = 0.000000 )
 rr_adjustment.maximum_time.@float  = 2400.000000
 rr_adjustment.final_tonnage.in_progress is FALSE
```

```
 rr_adjustment.final_tonnage.was_in_progress is FALSE
 rr_all.time_tons_average.@float  = 2400.000000
 rr_all.time_rr_exists.@float  = 600.000000
 rr_all.real_accumulated_losses.@double  = 0.000000
 rr_all.total_lost_last_shift.@double  = 0.000000
 rr_any.restriction.existed is FALSE
 rr_any.restriction.exists is FALSE
 rr_any.first_restriction.appeared is FALSE
 rr_any.last_restriction.disappeared is TRUE
 rr_any.do_not_accumulate.tonnage_current_rr is FALSE
 rr_any.stop_accumulating.tonnage_lost is FALSE
 rr_any.minimum_time_to_estimate.@float  = 600.000000
 rr_any.time_no_exists_restriction.@float  = 600.000000
 rr_any.accumulate_losses.rr_sag_stops is FALSE
 rr_speed.restriction.ignore is FALSE
 rr_speed.restriction.detected_past_sampling is FALSE
 rr_speed.restriction.has_been_identified is FALSE
 rr_speed.time_elapsed.@float  = 0.000000
endProcedure

Procedure
@name = initialize_rr_categories
@do =
 ?category = CLASSTOLIST ( "restriction_category" )
 ALL ( ?category.current_restriction.exists is FALSE )
 ALL ( ?category.lost_current_shift.@double  = 0.000000 )
 ALL ( ?category.lost_last_shift.@double  = 0.000000 )
 ALL ( ?category.real_accumulated_losses.@double  = 0.000000 )
 category_all.user_chose.new_cause is FALSE
endProcedure

Procedure
@name = initialize_tonnage_adjustment
@do =
 rr_adjustment.restriction_1.@string is ""
 rr_adjustment.cause_1.@string is ""
 rr_adjustment.time_elapsed_1.@float  = 0.000000
 rr_adjustment.do_not_correct.tonnage_lost is FALSE
 rr_adjustment.did_not_correct.tonnage_lost is FALSE
 rr_adjustment.ignore_tonnage.correction is FALSE
endProcedure

Procedure
@name = initialize_user_settings
@do =
 iss_user.do_not_correct.give_warning is FALSE
 iss_user.accumulate_current_rr.give_warning is FALSE
 iss_user.accumulate_tonnage_lost.give_warning is FALSE
endProcedure

Procedure
@name = initialize_variables
@do =
 RUN_PROCEDURE ( "max_min_values" )
 sag_speed.real_amt.@float  = 75
 fresh_feed.real_amt.@float  = 1500
 fresh_feed.real_setpoint.@float  = 1500
 fresh_feed.rate_of_change.@float  = 3600
 fresh_feed.hi_limit_target.@float  = 2000
 fresh_feed.hi_limit_real.@float  = 2000
 fresh_feed.controller_mode.cascade is TRUE
```

```
 sag_power.real_amt.@float  = 6300
 sag_power.real_setpoint.@float  = 6600
 water_ratio.real_amt.@float  = 0.650000
 hi_tons.target_hi_limit.@float  = 2000
 hi_tons_in.real_amt.@float  = 1500
 hi_tons_out.real_amt.@float  = 1500
 level_tailing_box.real_amt.@float  = 2
endProcedure

Procedure
@name = max_min_values
@do =
 fresh_feed.real_amt_min.@float  = 0
 fresh_feed.real_amt_max.@float  = 2000
 sag_amps.real_amt_min.@float  = 0
 sag_amps.real_amt_max.@float  = 7000
endProcedure

Procedure
@name = mss_sag_power
@do =
 fresh_feed.pred_amt.@float  = fresh_feed.real_amt.@float  + ( sag_amps.real_setpoint.@float  -
sag_amps.real_amt.@float  ) * sag_power.model_parameter.@float
endProcedure

Procedure
@name = mss_sag_speed_definition
@do =
 mss_sag_speed.input_l.@float  = 65.000000
 mss_sag_speed.output_l.@float  = 900.000000
 mss_sag_speed.input_m.@float  = 77.000000
 mss_sag_speed.output_m.@float  = 100.000000
 mss_sag_speed.input_h.@float  = 79.000000
 mss_sag_speed.output_h.@float  = 0.000000
endProcedure

Procedure
@name = mss_sag_speed_run
@do =
 dummy.model_name.@string is "mss_sag_speed"
 mss_sag_speed.input.@double  = sag_speed.real_amt.@float
 MACRO ( "qmss_table3" )
 fresh_feed.pred_amt.@float  = fresh_feed.real_amt.@float  + mss_sag_speed.output.@double
 TEXT ( "I'm here in MSS_SAG_SPEED_RUN ", "developer" )
endProcedure

Procedure
@name = obtain_predicted_dynamics
@do =
 STRCOPY ( dummy.var.@string, "" )
 RUN_PROCEDURE ( "dynamics_predicted" )
endProcedure

Procedure
@name = obtain_real_dynamics
@do =
 STRCOPY ( dummy.var.@string, "fresh_feed" )
 RUN_PROCEDURE ( "dynamics_real" )
endProcedure

Procedure
```

```
@name = qm_amps
@do =
 dummy.expo.@double = EXP ( - timex.sampling.@float / sag_amps.time_constant.@float )
 sag_amps.pred_amt_ss.@float = sag_amps.qm_param1.@float * fresh_feed.real_amt.@float
 sag_amps.pred_amt.@float = sag_amps.pred_amt_old.@float * dummy.expo.@double + ( 1 - dummy.expo.@double )
* sag_amps.pred_amt_ss.@float
 sag_amps.pred_amt_old.@float = sag_amps.pred_amt.@float
endProcedure

Procedure
@name = qm_dyn_first
@do =
 qm_dyn_first.dummy.@double = EXP ( - timex.sampling.@float / qm_dyn_first.time_constant.@float )
 qm_dyn_first.output.@double = qm_dyn_first.output_old.@double * qm_dyn_first.dummy.@double + ( 1 -
qm_dyn_first.dummy.@double ) * qm_dyn_first.output_ss.@double
endProcedure

Procedure
@name = qmss_table3_high
@do =
 dummy.model_slope.@double = ( <dummy.model_name.@string>.output_h.@float -
<dummy.model_name.@string>.output_m.@float ) / ( <dummy.model_name.@string>.input_h.@float -
<dummy.model_name.@string>.input_m.@float )
 <dummy.model_name.@string>.output.@double = ( <dummy.model_name.@string>.input.@double -
<dummy.model_name.@string>.input_m.@float ) * dummy.model_slope.@double +
<dummy.model_name.@string>.output_m.@float
endProcedure

Procedure
@name = qmss_table3_low
@do =
 dummy.model_slope.@double = ( <dummy.model_name.@string>.output_m.@float -
<dummy.model_name.@string>.output_l.@float ) / ( <dummy.model_name.@string>.input_m.@float -
<dummy.model_name.@string>.input_l.@float )
 <dummy.model_name.@string>.output.@double = ( <dummy.model_name.@string>.input.@double -
<dummy.model_name.@string>.input_l.@float ) * dummy.model_slope.@double +
<dummy.model_name.@string>.output_l.@float
endProcedure

Procedure
@name = re_configuration
@do =
 runx.simulation.again is FALSE
 timex.elapsed.@float = 0
endProcedure

Procedure
@name = read_data
@do =
 RUN_PROCEDURE ( "x_read_variables" )
 RUN_PROCEDURE ( "x_read_feeder_status" )
 RUN_PROCEDURE ( "x_read_equipment_status" )
 RUN_PROCEDURE ( "amps_deviation" )
 sag_amps.real_average.@float = TIMEAVERAGE ( sag_amps.real_amt.@float , 0, 3600 )
 fresh_feed.real_average.@float = TIMEAVERAGE ( fresh_feed.real_amt.@float , 0, 3600 )
 MACRO ( "feeders_all_running*" )
 MACRO ( "check_chutes*" )
 MACRO ( "check_equip_status*" )
endProcedure

Procedure
```

```
@name = read_data_coordinator
@do =
 timex.current_timex.@time is $Current_time
 RUN_PROCEDURE ( "read_data" )
 RUN_PROCEDURE ( "convert_all_measurements" )
 RUN_PROCEDURE ( "convert_all_setpoints" )
 RUN_PROCEDURE ( "obtain_real_dynamics" )
 MACRO ( "controller_deviation" )
 MACRO ( "variables_high*" )
endProcedure

Procedure
@name = read_feeder_status
@do =
 GETPOINTDATA ( "feeder1.status.running" )
 GETPOINTDATA ( "feeder2.status.running" )
 GETPOINTDATA ( "feeder3.status.running" )
 GETPOINTDATA ( "feeder4.status.running" )
 GETPOINTDATA ( "feeder5.status.running" )
endProcedure

Procedure
@name = read_variables
@do =
 GETPOINTDATA ( "fresh_feed.real_amt.@f" )
 GETPOINTDATA ( "fresh_feed.real_setpoint.@f" )
 GETPOINTDATA ( "water_ratio.real_amt.@float" )
endProcedure

Procedure
@name = reasoning_coordinator
@do =
 analyze.simulation.results is TRUE
 analyze.simulation.results is FALSE
endProcedure

Procedure
@name = rotate_all_predictions
@do =
 STRCOPY ( dummy.rotate.@string, "" )
 RUN_PROCEDURE ( "rotate_predicted_var" )
endProcedure

Procedure
@name = rotate_predicted_var
@do =
 <dummy.rotate.@string>.pred_amt_old.@float  = <dummy.rotate.@string>.pred_amt.@float
endProcedure

Procedure
@name = run_model_dyn
@do =
 RUN_PROCEDURE ( "" )
endProcedure

Procedure
@name = run_model_ss
@do =
 RUN_PROCEDURE ( "" )
endProcedure
```

```
Procedure
@name = simulation_parameters
@do =
 timex.sampling.@float  = 5
 timex.sleep.@float  = 1.000000
endProcedure

Procedure
@name = start_simulation
@do =
 runx.simulation.again is TRUE
 PUTPOINTDATA ( "runx.simulation.again", 100 )
endProcedure

Procedure
@name = stop_simulation
@do =
 runx.simulation.again is FALSE
 PUTPOINTDATA ( "runx.simulation.again", 0 )
endProcedure

Procedure
@name = system_initialization
@do =
 timex.sampling.@float  = 60
 RUN_PROCEDURE ( "controller_name" )
 RUN_PROCEDURE ( "max_min_values" )
 RUN_PROCEDURE ( "initial_values" )
endProcedure

Procedure
@name = update_database
@do =
 DONOTHING ( )
endProcedure

Procedure
@name = update_time
@do =
 timex.current_timex.@time is $Current_time
endProcedure

Procedure
@name = write_db_coordinator
@do =
 RUN_PROCEDURE ( "write_variables" )
 RUN_PROCEDURE ( "write_feeder_status" )
 RUN_PROCEDURE ( "write_mill_status" )
endProcedure

Procedure
@name = write_feeder_status
@do =
 PUTPOINTDATA ( "feeder1.status.@float", feeder1.status.@float  , 100 )
 PUTPOINTDATA ( "feeder2.status.@float", feeder2.status.@float  , 100 )
 PUTPOINTDATA ( "feeder3.status.@float", feeder3.status.@float  , 100 )
 PUTPOINTDATA ( "feeder4.status.@float", feeder4.status.@float  , 100 )
 PUTPOINTDATA ( "feeder5.status.@float", feeder5.status.@float  , 100 )
endProcedure

Procedure
```

```
@name = write_mill_status
@do =
 PUTPOINTDATA ( "ball_mill_1.status.@float", ball_mill_1.status.@float , 100 )
 PUTPOINTDATA ( "ball_mill_2.status.@float", ball_mill_2.status.@float , 100 )
 PUTPOINTDATA ( "sag_mill.status.@float", sag_mill.status.@float , 100 )
endProcedure

Procedure
@name = write_variables
@do =
 PUTPOINTDATA ( "fresh_feed.real_amt.@float", fresh_feed.real_amt.@float , 100 )
 PUTPOINTDATA ( "fresh_feed.real_setpoint.@float", fresh_feed.real_setpoint.@float , 100 )
 PUTPOINTDATA ( "water_ratio.real_amt.@float", water_ratio.real_amt.@float , 100 )
endProcedure

Restriction
@name = zero
@constraint = $OAV >= 0.000100
endRestriction

Ifchange
@name = equipment_status
@objectinherit = dummy.equipment.@string is OBJ ( $OAV )
 FREERULE ( $Rule, "equipment_running_1" )
 MACRO ( "equipment_running_1" )
endIfchange

Ifchange
@name = max_certainty
@objectinherit = ASNCERTAINTY ( $OAV, 100 )
endIfchange

Ifchange
@name = rr_dep_timing
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, OBJ ( $OAV ) )
 FREERULE ( $Rule, "timing_rr_*" )
 MACRO ( "timing_rr_dep" )
 MACRO ( "timing_rr_dep_elapsed" )
 MACRO ( "timing_rr_final" )
endIfchange

Ifchange
@name = rr_feeder_down_exists
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, "rr_feeder_down" )
 MACRO ( "timing_rr_indep" )
 MACRO ( "timing_rr_elapsed" )
 MACRO ( "timing_rr_final" )
endIfchange

Ifchange
@name = rr_ffeed_cut_exists
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, "rr_ffeed_cut" )
 MACRO ( "timing_rr_indep" )
 MACRO ( "timing_rr_elapsed" )
 MACRO ( "timing_rr_final" )
endIfchange

Ifchange
@name = rr_ffeed_limit_exists
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, "rr_ffeed_limit" )
 FREERULE ( $Rule, "timing_rr_*" )
```

```
 MACRO ( "timing_rr_dep" )
 MACRO ( "timing_rr_elapsed" )
 MACRO ( "timing_rr_final" )
endIfchange

Ifchange
@name = rr_indep_timing
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, OBJ ( $OAV ) )
 FREERULE ( $Rule, "timing_rr_*" )
 MACRO ( "timing_rr_indep" )
 MACRO ( "timing_rr_indep_elapsed" )
 MACRO ( "timing_rr_final" )
endIfchange

Ifchange
@name = rr_ore_supply_exists
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, "rr_ore_supply" )
 MACRO ( "timing_rr_indep" )
 MACRO ( "timing_rr_elapsed" )
 MACRO ( "timing_rr_final" )
endIfchange

Ifchange
@name = rr_sag_stops_exists
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, "rr_sag_stops" )
 MACRO ( "timing_rr_indep" )
 MACRO ( "timing_rr_elapsed" )
 MACRO ( "timing_rr_final" )
endIfchange

Ifchange
@name = rr_soft_ore_exists
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, "rr_soft_ore" )
 MACRO ( "timing_rr_dep" )
 MACRO ( "timing_rr_elapsed" )
 MACRO ( "timing_rr_final" )
endIfchange

Ifchange
@name = rr_speed_detected
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, OBJ ( $OAV ) )
 MACRO ( "timing_rr_speed" )
 MACRO ( "timing_rr_elapsed" )
 MACRO ( "timing_rr_speed_final" )
endIfchange

Ifchange
@name = rr_speed_timing
@objectinherit = rr_any.current_restriction.@string is OBJ ( $OAV )
 FREERULE ( $Rule, "timing_rr_speed_*" )
 MACRO ( "timing_rr_speed" )
 MACRO ( "timing_rr_speed_elapsed" )
 MACRO ( "timing_rr_speed_final" )
endIfchange

Ifchange
@name = rr_water_ratio_exists
@objectinherit = STRCOPY ( rr_any.current_restriction.@string, "rr_water_ratio" )
 MACRO ( "timing_rr_indep" )
 MACRO ( "timing_rr_elapsed" )
 MACRO ( "timing_rr_final" )
```

```
endIfchange

Ifchange
@name = set_time_constant
@objectinherit =  dummy.name.@string = OBJ ( $OAV )
 <dummy.name.@string>.time_constant.@float  = $OAV / 4
endIfchange

Ifchange
@name = set_time_s_state
@objectinherit =  dummy.name.@string = OBJ ( $OAV )
 <dummy.name.@string>.time_s_state.@float  = $OAV * 4
endIfchange

Fuzzy
@name = agrees_with_ffeed
@source =  fresh_feed.deviation.@float
@range = 3
@value = 0.000000, 4.000000, 100.000000
@rank = 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = close_to_ffeed
@source =  fresh_feed.deviation.@float
@range = 5
@value = 0.000000, 2.000000, 5.000000, 11.000000, 100.000000
@rank = 0.000000, 50.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = constant_ffeed
@source =  fresh_feed.real_dyn.@float
@range = 5
@value = -100.000000, -5.000000, 0.000000, 5.000000, 100.000000
@rank = 0.000000, 0.000000, 100.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = decreasing_ffeed
@source =  fresh_feed.real_dyn.@float
@range = 4
@value = -100.000000, -5.000000, 5.000000, 100.000000
@rank = 100.000000, 50.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = increasing_ffeed
@source =  fresh_feed.real_dyn.@float
@range = 4
@value = -100.000000, -5.000000, 5.000000, 100.000000
@rank = 0.000000, 0.000000, 50.000000, 100.000000
endFuzzy

Fuzzy
@name = level_tailings_high
@source =  level_tailing_box.real_amt.@float
@range = 3
@value = 0.000000, 1.500000, 2.000000
@rank = 0.000000, 100.000000, 100.000000
endFuzzy
```

```
Fuzzy
@name = positive_fresh_feed
@source =  fresh_feed.std_amt.@float
@range = 3
@value = 0.000000, 2.500000, 100.000000
@rank = 0.000000, 50.000000, 100.000000
endFuzzy

Fuzzy
@name = positive_sag_speed
@source =  sag_speed.real_amt.@float
@range = 3
@value = 0.000000, 2.500000, 100.000000
@rank = 0.000000, 50.000000, 100.000000
endFuzzy

Fuzzy
@name = tail_box1_high
@source =  tail_box1.real_amt.@float
@range = 3
@value = 0.000000, 14.000000, 15.000000
@rank = 0.000000, 50.000000, 100.000000
endFuzzy

Fuzzy
@name = tail_box2_high
@source =  tail_box2.real_amt.@float
@range = 3
@value = 0.000000, 14.000000, 15.000000
@rank = 0.000000, 50.000000, 100.000000
endFuzzy

Fuzzy
@name = zero_fresh_feed
@source =  fresh_feed.std_amt.@float
@range = 4
@value = 0.000000, 2.500000, 5.000000, 100.000000
@rank = 100.000000, 50.000000, 0.000000, 0.000000
endFuzzy

Fuzzy
@name = zero_sag_speed
@source =  sag_speed.real_amt.@float
@range = 4
@value = 0.000000, 2.500000, 5.000000, 100.000000
@rank = 100.000000, 50.000000, 0.000000, 0.000000
endFuzzy

Rule
@name = accumulate_adjusted_1
IF  rr_any.stop_accumulating.tonnage_lost is FALSE
THEN dummy.losses.@double  = <rr_any.current_restriction.@string>.lost_current_shift.@double
THEN <rr_any.current_restriction.@string>.lost_current_shift.@double  = dummy.losses.@double  +
rr_adjustment.correction_tons_accumulated.@double
THEN dummy.losses.@double  = <rr_any.current_restriction.@string>.real_accumulated_losses.@double
THEN <rr_any.current_restriction.@string>.real_accumulated_losses.@double  = dummy.losses.@double  +
rr_adjustment.correction_tons_accumulated.@double
THEN MACRO ( "accumulate_losses_11" )
THEN TEXT ( "here in accum adjusted 1", "developer" )
THEN TEXT ( " current rr: !$ rr_any.current_restriction.@s $!", "developer" )
```

```
THEN TEXT ( " lost_curr_shift: !$ <rr_any.current_restriction.@s>.lost_current_shift.@db $!", "developer" )
THEN TEXT ( " real_accum_losses: !$ <rr_any.current_restriction.@s>.real_accumulated_losses.@db $! ", "developer" )
THEN TEXT ( "adjustment: !$ rr_adjustment.correction_tons_accumulated.@db $! ", "developer" )
ELSE IGNORE ( $Rule, "accumulate_adjusted*" )
endRule

Rule
@name = accumulate_adjusted_2
IF  TRUE
THEN dummy.losses.@double = <category_any.current_category.@string>.lost_current_shift.@double
THEN <category_any.current_category.@string>.lost_current_shift.@double = dummy.losses.@double +
rr_adjustment.correction_tons_accumulated.@double
THEN dummy.losses.@double = <category_any.current_category.@string>.real_accumulated_losses.@double
THEN <category_any.current_category.@string>.real_accumulated_losses.@double = dummy.losses.@double +
rr_adjustment.correction_tons_accumulated.@double
THEN FREERULE ( $Rule, "accumulate_losses*" )
THEN MACRO ( "accumulate_losses_21" )
THEN MACRO ( "accumulate_losses_3*" )
THEN TEXT ( "accum adjusted 2...", "developer" )
endRule

Rule
@name = accumulate_losses_0
@comment = "called from RR TIMING FINAL"
IF  rr_any.stop_accumulating.tonnage_lost
OR  rr_any.do_not_accumulate.tonnage_current_rr
THEN <rr_any.current_restriction.@string>.lost_current_rr.@double = 0.000000
THEN MACRO ( "accumulate_losses_4" )
THEN IGNORE ( $Rule, "accumulate_losses*" )
endRule

Rule
@name = accumulate_losses_1
@comment = "called from RR TIMING FINAL"
IF  <rr_any.current_restriction.@string>.time_elapsed.@float  >= rr_any.minimum_time_to_estimate.@float
THEN dummy.losses.@double = <rr_any.current_restriction.@string>.lost_current_shift.@double
THEN <rr_any.current_restriction.@string>.lost_current_shift.@double = dummy.losses.@double +
<rr_any.current_restriction.@string>.lost_current_rr.@double
THEN dummy.losses.@double = <rr_any.current_restriction.@string>.real_accumulated_losses.@double
THEN <rr_any.current_restriction.@string>.real_accumulated_losses.@double = dummy.losses.@double +
<rr_any.current_restriction.@string>.lost_current_rr.@double
ELSE <rr_any.current_restriction.@string>.lost_current_rr.@double = 0.000000
ELSE TEXT ( " the restriction '!$ rr_any.current_restriction.@s$!' has been ignored:  Not long enough! ", "restrictions" )
ELSE rr_adjustment.ignore_tonnage.correction is TRUE
ELSE IGNORE ( $Rule, "accumulate_losses*" )
endRule

Rule
@name = accumulate_losses_11
@comment = "called from RR TIMING FINAL"
IF  rr_any.accumulate_losses.rr_sag_stops
THEN rr_all.total_lost_current_shift.@double = CLASSTOTAL ( {restriction_id}.lost_current_shift.@double )
THEN rr_all.real_accumulated_losses.@double = CLASSTOTAL ( {restriction_id}.real_accumulated_losses.@double )
ELSE rr_all.total_lost_current_shift.@double = CLASSTOTAL ( {restriction_id}.lost_current_shift.@double ) -
rr_sag_stops.lost_current_shift.@double
ELSE rr_all.real_accumulated_losses.@double = CLASSTOTAL ( {restriction_id}.real_accumulated_losses.@double ) -
rr_sag_stops.real_accumulated_losses.@double
endRule

Rule
@name = accumulate_losses_2
```

```
IF  ?causex = ANY ( {restriction_category}.current_restriction.exists is TRUE )
THEN category_any.current_category.@string is LISTFIRSTOBJECT ( ?causex )
THEN dummy.string.@string is LISTALLOBJECT ( ?causex, "", "" )
THEN TEXT ( "End of restriction  !$ rr_any.current_restriction.@s $! -- causes .. !$ dummy.string.@s $! ...", "developer" )
THEN dummy.losses.@double  = <category_any.current_category.@string>.lost_current_shift.@double
THEN <category_any.current_category.@string>.lost_current_shift.@double  = dummy.losses.@double +
<rr_any.current_restriction.@string>.lost_current_rr.@double
THEN dummy.losses.@double  = <category_any.current_category.@string>.real_accumulated_losses.@double
THEN <category_any.current_category.@string>.real_accumulated_losses.@double  = dummy.losses.@double +
<rr_any.current_restriction.@string>.lost_current_rr.@double
THEN <rr_any.current_restriction.@string>.lost_current_rr.@double  = 0.000000
endRule

Rule
@name = accumulate_losses_21
IF  rr_any.accumulate_losses.rr_sag_stops
THEN category_all.total_lost_current_shift.@double  = CLASSTOTAL (
{restriction_category}.lost_current_shift.@double  )
THEN category_all.real_accumulated_losses.@double  = CLASSTOTAL (
{restriction_category}.real_accumulated_losses.@double  )
ELSE category_all.total_lost_current_shift.@double  = CLASSTOTAL ( {restriction_category}.lost_current_shift.@double
) - sag_shut_down.lost_current_shift.@double
ELSE category_all.real_accumulated_losses.@double  = CLASSTOTAL (
{restriction_category}.real_accumulated_losses.@double  ) - sag_shut_down.real_accumulated_losses.@double
endRule

Rule
@name = accumulate_losses_3
IF  ?category_x = CLASSTOLIST ( "restriction_category" )
THEN ALL ( ?category_x.percent_accumulated_losses.@float  = ?category_x.real_accumulated_losses.@double /
category_all.real_accumulated_losses.@double  * 100.000000 )
endRule

Rule
@name = accumulate_losses_31
IF  rr_any.accumulate_losses.rr_sag_stops
THEN category_all.percent_accumulated_losses.@float  = CLASSTOTAL (
{restriction_category}.percent_accumulated_losses.@float  )
ELSE category_all.percent_accumulated_losses.@float  = CLASSTOTAL (
{restriction_category}.percent_accumulated_losses.@float  ) - sag_shut_down.percent_accumulated_losses.@float
endRule

Rule
@name = accumulate_losses_4
@comment = "needed when no restriction exists (the cuases has not been set to false)"
IF  ?restrictionx = ANY ( {restriction_id}.restriction.exists is TRUE )
THEN DONOTHING ( )
ELSE <category_any.current_category.@string>.current_restriction.exists is FALSE
endRule

Rule
@name = amps_model_0
IF  fresh_feed.real_average.@float  >= 1200
THEN sag_amps.qm_param1.@float  = sag_amps.real_average.@float  / fresh_feed.real_average.@float
ELSE sag_amps.qm_param1.@float  = sag_amps.qm_param_original.@float
endRule

Rule
@name = amps_model_1
IF  TRUE
THEN RUN_PROCEDURE ( "qm_amps" )
```

```
endRule

Rule
@name = cause_rr_feeder_down_0
IF  rr_feeder_down.possible_cause.determined is FALSE
THEN MACRO ( "initialize_causes*" )
THEN rr_feeder_down.possible_cause.feeders is FALSE
THEN rr_feeder_down.possible_cause.plugged_chutes is FALSE
ELSE IGNORE ( $Rule, "cause_rr_feeder_down*" )
endRule

Rule
@name = cause_rr_feeder_down_1
IF  chutes.status.plugged
THEN rr_feeder_down.possible_cause.plugged_chutes is TRUE
THEN plugged_chutes.current_restriction.exists is TRUE
THEN rr_feeder_down.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_feeder_down' is: Plugged chutes", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_feeder_down' is: Plugged chutes", "restrictions" )
THEN IGNORE ( $Rule, "cause_rr_feeder_down*" )
endRule

Rule
@name = cause_rr_feeder_down_2
IF  TRUE
THEN rr_feeder_down.possible_cause.feeders is TRUE
THEN feeders.current_restriction.exists is TRUE
THEN rr_feeder_down.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_feeder_down' is: Feeder problems", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_feeder_down' is: Feeder problems", "restrictions" )
endRule

Rule
@name = cause_rr_ffeed_cut_0
IF  rr_ffeed_cut.possible_cause.determined is FALSE
THEN MACRO ( "initialize_causes*" )
THEN rr_ffeed_cut.possible_cause.ball_mills is FALSE
THEN rr_ffeed_cut.possible_cause.bm_pumps is FALSE
THEN rr_ffeed_cut.possible_cause.sag_control is FALSE
THEN rr_ffeed_cut.possible_cause.other is FALSE
ELSE IGNORE ( $Rule, "cause_rr_ffeed_cut*" )
endRule

Rule
@name = cause_rr_ffeed_cut_1
IF  ball_mill_1.status.running is FALSE
OR  ball_mill_2.status.running is FALSE
THEN rr_ffeed_cut.possible_cause.ball_mills is TRUE
THEN rr_ffeed_cut.possible_cause.bm_pumps is TRUE
THEN ball_mills.current_restriction.exists is TRUE
THEN rr_ffeed_cut.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_cut' is: Ball Mills or BM pumps", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_cut' is: Ball Mills or BM pumps", "restrictions" )
THEN IGNORE ( $Rule, "cause_rr_ffeed_cut*" )
endRule

Rule
@name = cause_rr_ffeed_cut_2
IF  ball_mill_1.status.running is TRUE
AND ball_mill_2.status.running is TRUE
THEN rr_ffeed_cut.possible_cause.sag_control is TRUE
```

```
THEN sag_control.current_restriction.exists is TRUE
THEN rr_ffeed_cut.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_cut' is: SAG Mill controls", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_cut' is: SAG Mill controls", "restrictions" )
THEN IGNORE ( $Rule, "cause_rr_ffeed_cut*" )
endRule

Rule
@name = cause_rr_ffeed_cut_3
IF  rr_ffeed_cut.possible_cause.determined is FALSE
THEN rr_ffeed_cut.possible_cause.other is TRUE
THEN other.current_restriction.exists is TRUE
THEN rr_ffeed_cut.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_cut' is: Another problem", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_cut' is: Another problem", "restrictions" )
endRule

Rule
@name = cause_rr_ffeed_limit_0
IF  rr_ffeed_limit.possible_cause.determined is FALSE
THEN MACRO ( "initialize_causes*" )
THEN rr_ffeed_limit.possible_cause.ball_mills is FALSE
THEN rr_ffeed_limit.possible_cause.bm_pumps is FALSE
THEN rr_ffeed_limit.possible_cause.tailings is FALSE
THEN rr_ffeed_limit.possible_cause.other is FALSE
ELSE IGNORE ( $Rule, "cause_rr_ffeed_limit*" )
endRule

Rule
@name = cause_rr_ffeed_limit_1
IF  ball_mill_1.status.running is FALSE
OR  ball_mill_2.status.running is FALSE
THEN rr_ffeed_limit.possible_cause.ball_mills is TRUE
THEN rr_ffeed_limit.possible_cause.bm_pumps is TRUE
THEN ball_mills.current_restriction.exists is TRUE
THEN rr_ffeed_limit.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_limit' is: Ball Mills or BM pumps", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_limit' is: Ball Mills or BM pumps", "restrictions" )
THEN IGNORE ( $Rule, "cause_rr_ffeed_limit*" )
endRule

Rule
@name = cause_rr_ffeed_limit_2
IF  ball_mill_1.status.running
AND ball_mill_2.status.running
AND ( tail_box1.level.high | tail_box2.level.high )
THEN rr_ffeed_limit.possible_cause.tailings is TRUE
THEN tailings.current_restriction.exists is TRUE
THEN rr_ffeed_limit.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_limit' is: Level of tailings box", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_limit' is: Level of tailings box", "restrictions" )
THEN IGNORE ( $Rule, "cause_rr_ffeed_limit*" )
endRule

Rule
@name = cause_rr_ffeed_limit_3
IF  TRUE
THEN rr_ffeed_limit.possible_cause.other is TRUE
THEN other.current_restriction.exists is TRUE
THEN rr_ffeed_limit.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_limit' is: Another problem", "rr-cause" )
```

```
THEN TEXT ( "Possible cause of restriction 'rr_ffeed_limit' is: Another problem", "restrictions" )
endRule

Rule
@name = cause_rr_ore_supply_0
IF  rr_ore_supply.possible_cause.determined is FALSE
THEN MACRO ( "initialize_causes*" )
THEN rr_ore_supply.possible_cause.plugged_chutes is FALSE
THEN rr_ore_supply.possible_cause.feeders is FALSE
THEN rr_ore_supply.possible_cause.other is FALSE
ELSE IGNORE ( $Rule, "cause_rr_ore_supply*" )
endRule

Rule
@name = cause_rr_ore_supply_1
IF  chutes.status.plugged
THEN rr_ore_supply.possible_cause.plugged_chutes is TRUE
THEN plugged_chutes.current_restriction.exists is TRUE
THEN rr_ore_supply.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_ore_supply' is: Chutes are plugged", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_ore_supply' is: Chutes are plugged", "restrictions" )
THEN IGNORE ( $Rule, "cause_rr_ore_supply*" )
endRule

Rule
@name = cause_rr_ore_supply_2
IF  feeder_set.status_all.running is FALSE
AND  chutes.status.plugged is FALSE
THEN rr_ore_supply.possible_cause.feeders is TRUE
THEN feeders.current_restriction.exists is TRUE
THEN rr_ore_supply.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_ore_supply' is: Feeder problems", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_ore_supply' is: Feeder problems", "restrictions" )
THEN IGNORE ( $Rule, "cause_rr_ore_supply*" )
endRule

Rule
@name = cause_rr_ore_supply_3
IF  TRUE
THEN rr_ore_supply.possible_cause.other is TRUE
THEN other.current_restriction.exists is TRUE
THEN rr_ore_supply.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_ore_supply' is: Another problem", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_ore_supply' is: Another problem", "restrictions" )
endRule

Rule
@name = cause_rr_sag_stops_1
IF  rr_sag_stops.possible_cause.determined is FALSE
THEN MACRO ( "initialize_causes*" )
THEN rr_sag_stops.possible_cause.sag_shut_down is TRUE
THEN sag_shut_down.current_restriction.exists is TRUE
THEN rr_sag_stops.possible_cause.determined is TRUE
THEN TEXT ( " SAG Mill SHUT DOWN", "restrictions" )
endRule

Rule
@name = cause_rr_soft_ore_1
IF  rr_soft_ore.possible_cause.determined is FALSE
THEN MACRO ( "initialize_causes*" )
THEN rr_soft_ore.possible_cause.soft_ore is TRUE
```

```
THEN soft_ore.current_restriction.exists is TRUE
THEN rr_soft_ore.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_soft_ore' is: Soft Ore", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_soft_ore' is: Soft Ore", "restrictions" )
endRule

Rule
@name = cause_rr_water_ratio_1
IF  rr_water_ratio.possible_cause.determined is FALSE
THEN rr_water_ratio.possible_cause.sag_control is TRUE
THEN rr_water_ratio.possible_cause.determined is TRUE
THEN TEXT ( "Possible cause of restriction 'rr_water_ratio' is: SAG Mill controls", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_water_ratio' is: SAG Mill controls", "restrictions" )
endRule

Rule
@name = check_chutes
IF  ?chutex = ANY ( {feeder}.status_chute.plugged )
OR  chute_feed.status_chute.plugged
OR  chute_conveyor.status_chute.plugged
THEN chutes.status.plugged is TRUE
ELSE chutes.status.plugged is FALSE
endRule

Rule
@name = check_controller_ffeed
IF  fresh_feed.real_amt.@float  < fresh_feed.real_setpoint.@float
AND fresh_feed.follows.setpoint is FALSE
THEN fresh_feed.controller.meets_demand is FALSE
ELSE fresh_feed.controller.meets_demand is TRUE
endRule

Rule
@name = check_controllers_1
IF  <dummy.process_var.@string>.real_setpoint.@float  > <dummy.process_var.@string>.real_amt.@float
AND <dummy.process_var.@string>.real_dyn.increasing
THEN <dummy.process_var.@string>.is_approaching_to.setpoint is TRUE
ELSE <dummy.process_var.@string>.is_approaching_to.setpoint is FALSE
endRule

Rule
@name = check_controllers_2
IF  <dummy.process_var.@string>.real_setpoint.@float  < <dummy.process_var.@string>.real_amt.@float
AND <dummy.process_var.@string>.real_dyn.decreasing
THEN <dummy.process_var.@string>.is_approaching_to.setpoint is TRUE
ELSE <dummy.process_var.@string>.is_approaching_to.setpoint is FALSE
endRule

Rule
@name = check_controllers_3
IF  <dummy.process_var.@string>.agrees_with.setpoint
OR  <dummy.process_var.@string>.is_close_to.setpoint
OR  <dummy.process_var.@string>.is_approaching_to.setpoint
THEN <dummy.process_var.@string>.follows.setpoint is TRUE
ELSE <dummy.process_var.@string>.follows.setpoint is FALSE
endRule

Rule
@name = check_equip_status_1
IF  feeder1.status.running is FALSE
AND feeder1.status.was_running
```

```
THEN TEXT ( " ... Feeder-1 stopped ...", "warning" )
THEN TEXT ( " ... Feeder-1 STOPPED ...", "restrictions" )
endRule

Rule
@name = check_equip_status_10
IF feeder5.status.running
AND feeder5.status.was_running is FALSE
THEN TEXT ( " ... Feeder-5 STARTED again ...", "warning" )
THEN TEXT ( " ... Feeder-5 STARTED again ...", "restrictions" )
endRule

Rule
@name = check_equip_status_11
IF sag_mill.status.running is FALSE
AND sag_mill.status.was_running
THEN TEXT ( " ... SAG MILL SHUT-DOWN ...", "warning" )
THEN TEXT ( " ... SAG MILL SHUT-DOWN ...", "restrictions" )
endRule

Rule
@name = check_equip_status_12
IF sag_mill.status.running
AND sag_mill.status.was_running is FALSE
THEN TEXT ( " ... SAG MILL STARTED AGAIN ...", "warning" )
THEN TEXT ( " ... SAG MILL STARTED AGAIN ...", "restrictions" )
endRule

Rule
@name = check_equip_status_13
IF ball_mill_1.status.running is FALSE
AND ball_mill_1.status.was_running
THEN TEXT ( " ... BALL MILL 1 SHUT-DOWN ...", "warning" )
THEN TEXT ( " ... BALL MILL 1 SHUT-DOWN ...", "restrictions" )
endRule

Rule
@name = check_equip_status_14
IF ball_mill_2.status.running is FALSE
AND ball_mill_2.status.was_running
THEN TEXT ( " ... BALL MILL 2 SHUT-DOWN ...", "warning" )
THEN TEXT ( " ... BALL MILL 2 SHUT-DOWN ...", "restrictions" )
endRule

Rule
@name = check_equip_status_15
IF ball_mill_1.status.running
AND ball_mill_1.status.was_running is FALSE
THEN TEXT ( " ... BALL MILL 1 STARTED AGAIN ...", "warning" )
THEN TEXT ( " ... BALL MILL 1 STARTED AGAIN ...", "restrictions" )
endRule

Rule
@name = check_equip_status_16
IF ball_mill_2.status.running
AND ball_mill_2.status.was_running is FALSE
THEN TEXT ( " ... BALL MILL 2 STARTED AGAIN ...", "warning" )
THEN TEXT ( " ... BALL MILL 2 STARTED AGAIN ...", "restrictions" )
endRule

Rule
```

```
@name = check_equip_status_2
IF  feeder2.status.running is FALSE
AND feeder2.status.was_running
THEN TEXT ( " ... Feeder-2 stopped ...", "warning" )
THEN TEXT ( " ... Feeder-2 STOPPED ...", "restrictions" )
endRule

Rule
@name = check_equip_status_3
IF  feeder3.status.running is FALSE
AND feeder3.status.was_running
THEN TEXT ( " ... Feeder-3 stopped ...", "warning" )
THEN TEXT ( " ... Feeder-3 STOPPED ...", "restrictions" )
endRule

Rule
@name = check_equip_status_4
IF  feeder4.status.running is FALSE
AND feeder4.status.was_running
THEN TEXT ( " ... Feeder-4 stopped ...", "warning" )
THEN TEXT ( " ... Feeder-4 STOPPED ...", "restrictions" )
endRule

Rule
@name = check_equip_status_5
IF  feeder5.status.running is FALSE
AND feeder5.status.was_running
THEN TEXT ( " ... Feeder-5 stopped ...", "warning" )
THEN TEXT ( " ... Feeder-5 STOPPED ...", "restrictions" )
endRule

Rule
@name = check_equip_status_6
IF  feeder1.status.running
AND feeder1.status.was_running is FALSE
THEN TEXT ( " ... Feeder-1 STARTED again ...", "warning" )
THEN TEXT ( " ... Feeder-1 STARTED again ...", "restrictions" )
endRule

Rule
@name = check_equip_status_7
IF  feeder2.status.running
AND feeder2.status.was_running is FALSE
THEN TEXT ( " ... Feeder-2 STARTED again ...", "warning" )
THEN TEXT ( " ... Feeder-2 STARTED again ...", "restrictions" )
endRule

Rule
@name = check_equip_status_8
IF  feeder3.status.running
AND feeder3.status.was_running is FALSE
THEN TEXT ( " ... Feeder-3 STARTED again ...", "warning" )
THEN TEXT ( " ... Feeder-3 STARTED again ...", "restrictions" )
endRule

Rule
@name = check_equip_status_9
IF  feeder4.status.running
AND feeder4.status.was_running is FALSE
THEN TEXT ( " ... Feeder-4 STARTED again ...", "warning" )
THEN TEXT ( " ... Feeder-4 STARTED again ...", "restrictions" )
```

```
endRule

Rule
@name = choose_one_cause
IF  rr_any.restriction.exists
AND  rr_any.do_not_accumulate.tonnage_current_rr is FALSE
AND  rr_any.stop_accumulating.tonnage_lost is FALSE
THEN ?causex = CLASSTOLIST ( "restriction_category" )
THEN ALL ( ?causex.current_restriction.exists is FALSE )
THEN <dummy.new_cause.@string>.current_restriction.exists is TRUE
THEN TEXT ( "USER:  New cause for current restriction - !$ dummy.new_cause.@s $!", "rr-user" )
THEN TEXT ( "USER:  New cause for current restriction - !$ dummy.new_cause.@s $!", "restrictions" )
THEN category_all.user_chose.new_cause is TRUE
endRule

Rule
@name = choose_one_cause_2
@comment = "I'm not sure about the usefulness of this rule....."
IF  category_all.user_chose.new_cause
THEN ?causex = CLASSTOLIST ( "restriction_category" )
THEN dummy.new_cause.@string is LISTALLOBJECT ( ?causex, "", "" )
THEN TEXT ( "Possible cause of restriction 'rr_water_ratio' is: SAG Mill controls", "rr-cause" )
THEN TEXT ( "Possible cause of restriction 'rr_water_ratio' is: SAG Mill controls", "restrictions" )
THEN TEXT ( "User has chosen a new cause for the current restriction - NEW  CAUSE:  !$ dummy.new_cause.@s $!", "rr-
cause" )
THEN TEXT ( "User has chosen a new cause for the current restriction - NEW  CAUSE:  !$ dummy.new_cause.@s $!",
"restrictions" )
THEN category_all.user_chose.new_cause is FALSE
endRule

Rule
@name = controller_deviation
IF  ?p_var = ANY ( STRLEN ( {process_var}.controller_name.@string ) > 0 )
THEN ALL ( {process_var}.deviation.@float  = ABS ( {process_var}.std_setpoint.@float  -
{process_var}.std_amt.@float  ) )
endRule

Rule
@name = detect_rr_feeder_down_1
IF  feeder2.status.running is FALSE
OR  feeder3.status.running is FALSE
OR  feeder4.status.running is FALSE
THEN rr_feeder_down.restriction.detected is TRUE
ELSE rr_feeder_down.restriction.detected is FALSE
endRule

Rule
@name = detect_rr_ffeed_cut_1
IF  fresh_feed.has_been.cut
THEN rr_ffeed_cut.restriction.detected is TRUE
ELSE rr_ffeed_cut.restriction.detected is FALSE
endRule

Rule
@name = detect_rr_ffeed_limit_0
IF  fresh_feed.controller_mode.cascade is FALSE
THEN rr_ffeed_limit.restriction.detected is TRUE
ELSE rr_ffeed_limit.restriction.detected is FALSE
endRule

Rule
```

```
@name = detect_rr_ffeed_limit_1
IF  TRUE
THEN dummy.difference.@float  = fresh_feed.hi_limit_real.@float  - fresh_feed.real_setpoint.@float
endRule

Rule
@name = detect_rr_ffeed_limit_2
IF  dummy.difference.@float  <= 1.000000
AND  fresh_feed.hi_limit_real.@float  < fresh_feed.hi_limit_target.@float
THEN rr_ffeed_limit.restriction.detected is TRUE
ELSE rr_ffeed_limit.restriction.detected is FALSE
endRule

Rule
@name = detect_rr_ore_supply_1
IF  fresh_feed.controller.meets_demand is FALSE
AND  sag_mill.status.running
THEN rr_ore_supply.restriction.detected is TRUE
ELSE rr_ore_supply.restriction.detected is FALSE
endRule

Rule
@name = detect_rr_sag_stops_1
IF  sag_mill.status.running is FALSE
THEN rr_sag_stops.restriction.detected is TRUE
ELSE rr_sag_stops.restriction.detected is FALSE
endRule

Rule
@name = detect_rr_soft_ore
IF  fresh_feed.real_setpoint.@float  >= fresh_feed.hi_limit_real.@float
AND  fresh_feed.hi_limit_real.@float  >= fresh_feed.hi_limit_target.@float
THEN rr_soft_ore.restriction.detected is TRUE
ELSE rr_soft_ore.restriction.detected is FALSE
endRule

Rule
@name = detect_rr_speed_1
IF  hi_tons_in.real_amt.@float  > hi_tons_out.real_amt.@float
THEN rr_speed.restriction.detected is TRUE
ELSE rr_speed.restriction.detected is FALSE
endRule

Rule
@name = detect_rr_water_ratio_1
IF  water_ratio.real_amt.@float  < water_ratio.pred_amt.@float
THEN rr_water_ratio.restriction.detected is TRUE
ELSE rr_water_ratio.restriction.detected is FALSE
endRule

Rule
@name = determine_rr_causes_1
IF  ?current_rr = ANY ( {restriction_id}.restriction.exists is TRUE )
THEN ALL ( MACRO ( STRCONCAT ( "cause_", STRCONCAT ( OBJ ( ?current_rr.restriction.exists ), "_*" ) ) ) )
endRule

Rule
@name = determine_rr_causes_2
IF  ?categoryx = ANY ( {restriction_category}.current_restriction.exists is TRUE )
THEN category_all.current_restriction.@string is LISTFIRSTOBJECT ( ?categoryx )
ELSE category_all.current_restriction.@string is "none"
```

```
endRule

Rule
@name = determine_rr_exists_1
IF  TRUE
THEN MACRO ( "exists_rr_sag_stops*" )
THEN MACRO ( "exists_rr_ffeed_cut*" )
THEN MACRO ( "exists_rr_ore_supply*" )
THEN MACRO ( "exists_rr_ffeed_limit*" )
THEN MACRO ( "exists_rr_soft_ore*" )
THEN MACRO ( "exists_rr_feeder_down*" )
THEN MACRO ( "exists_any_restriction" )
THEN MACRO ( "exists_rr_any_2*" )
THEN MACRO ( "exists_rr_timing*" )
THEN MACRO ( "follow_up_begin_rr*" )
THEN MACRO ( "exists_rr_any_3*" )
endRule

Rule
@name = determine_tonnage_difference
IF  fresh_feed.pred_amt.@float  > fresh_feed.real_amt.@float
THEN fresh_feed.tonnage_difference.@float  = fresh_feed.pred_amt.@float  - fresh_feed.real_amt.@float
ELSE fresh_feed.tonnage_difference.@float  = 0
endRule

Rule
@name = do_not_accumulate_current_rr
IF  rr_any.do_not_accumulate.tonnage_current_rr
AND  iss_user.accumulate_current_rr.give_warning
THEN TEXT ( "USER:  Do not accumulate tonnage lost in current restriction", "rr-user" )
THEN TEXT ( "USER:  Do not accumulate tonnage lost in current restriction", "restrictions" )
THEN iss_user.accumulate_current_rr.give_warning is FALSE
endRule

Rule
@name = do_not_accumulate_tons_lost_1
IF  rr_any.stop_accumulating.tonnage_lost
AND  iss_user.accumulate_tonnage_lost.give_warning
THEN TEXT ( "USER:  Do not accumulate tonnage lost.", "rr-user" )
THEN TEXT ( "USER:  Do not accumulate tonnage lost.", "restrictions" )
THEN iss_user.accumulate_tonnage_lost.give_warning is FALSE
endRule

Rule
@name = do_not_accumulate_tons_lost_2
IF  rr_any.stop_accumulating.tonnage_lost is FALSE
AND  iss_user.accumulate_tonnage_lost.give_warning
THEN TEXT ( "USER:  Continue accumulating tonnage lost.", "rr-user" )
THEN TEXT ( "USER:  Continue accumulating tonnage lost.", "restrictions" )
THEN iss_user.accumulate_tonnage_lost.give_warning is FALSE
endRule

Rule
@name = do_not_correct_tons_1
IF  rr_adjustment.do_not_correct.tonnage_lost
AND  iss_user.do_not_correct.give_warning
THEN TEXT ( "USER: DISABLE algorithm to adjust tonnage estimated in each restriction.", "rr-user" )
THEN TEXT ( "USER: DISABLE algorithm to adjust tonnage estimated in each restriction.", "restrictions" )
THEN iss_user.do_not_correct.give_warning is FALSE
endRule
```

```
Rule
@name = do_not_correct_tons_2
IF  rr_adjustment.do_not_correct.tonnage_lost is FALSE
AND iss_user.do_not_correct.give_warning
THEN TEXT ( "USER: ENABLE algorithm to adjust tonnage estimated in each restriction.", "rr-user" )
THEN TEXT ( "USER: ENABLE algorithm to adjust tonnage estimated in each restriction.", "restrictions" )
THEN iss_user.do_not_correct.give_warning is FALSE
endRule

Rule
@name = end_of_shift_0
IF  end.of.shift is FALSE
THEN IGNORE ( $Rule, "end_of_shift*" )
ELSE end.of.shift is FALSE
endRule

Rule
@name = end_of_shift_1
IF  ?restrictionx = ANY ( {restriction_id}.restriction.exists is TRUE )
THEN rr_any.current_restriction.@string is LISTFIRSTOBJECT ( ?restrictionx )
THEN dummy.string.@string is LISTALLOBJECT ( ?restrictionx, "", "" )
THEN FREERULE ( $Rule, "accumulate_losses*" )
THEN MACRO ( "accumulate_losses*" )
THEN <rr_any.current_restriction.@string>.lost_last_shift.@double =
<rr_any.current_restriction.@string>.lost_current_shift.@double
THEN rr_all.total_lost_last_shift.@double = rr_all.total_lost_current_shift.@double
endRule

Rule
@name = end_of_shift_2
IF  ?restrictionx = CLASSTOLIST ( "restriction_id" )
THEN ALL ( ?restrictionx.lost_last_shift.@double = ?restrictionx.lost_current_shift.@double )
THEN rr_all.total_lost_last_shift.@double = rr_all.total_lost_current_shift.@double
THEN ALL ( ?restrictionx.lost_current_shift.@double = 0 )
THEN rr_all.total_lost_current_shift.@double = 0
endRule

Rule
@name = end_of_shift_3
IF  ?categoryx = CLASSTOLIST ( "restriction_category" )
THEN ALL ( ?categoryx.lost_last_shift.@double = ?categoryx.lost_current_shift.@double )
THEN category_all.total_lost_last_shift.@double = category_all.total_lost_current_shift.@double
THEN ALL ( ?categoryx.lost_current_shift.@double = 0 )
THEN category_all.total_lost_current_shift.@double = 0
endRule

Rule
@name = end_of_shift_4
IF  TRUE
THEN TEXT ( " ", "restrictions" )
THEN TEXT ( "****************** E N D  O F  S H I F T  **********", "restrictions" )
THEN TEXT ( "* ", "restrictions" )
THEN TEXT ( "* TOTAL TONNAGE LOST ACCUMULATED TO-DATE: !$ rr_all.real_accumulated_losses.@db $!
Tons. ", "restrictions" )
THEN TEXT ( "* ", "restrictions" )
THEN TEXT ( "* TOTAL TONNAGE LOST LAST DURING PAST SHIFT: !$ rr_all.total_lost_last_shift.@db $! Tons. ",
"restrictions" )
THEN TEXT ( "* ", "restrictions" )
THEN TEXT ( "*     R E S T R I C T I O N ", "restrictions" )
THEN TEXT ( "*   SAG not running : !$ rr_sag_stops.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*     Feed was cut : !$ rr_ffeed_cut.lost_last_shift.@db $! Tons. ", "restrictions" )
```

```
THEN TEXT ( "*       Feeder down : !$ rr_feeder_down.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*        Ore Supply : !$ rr_ore_supply.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*   Restricted feed : !$ rr_ffeed_limit.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*         Soft Ore : !$ rr_soft_ore.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "* ", "restrictions" )
THEN TEXT ( "*      C A T E G O R Y ", "restrictions" )
THEN TEXT ( "*        Soft Ore : !$ soft_ore.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*  Plugged chutes : !$ plugged_chutes.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "* SAG mill control : !$ sag_control.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*      Ball mills : !$ ball_mills.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*       BM pumps : !$ bm_pumps.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*       Flotation : !$ flotation.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*       Tailings : !$ tailings.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*        Feeders : !$ feeders.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "*          Other : !$ other.lost_last_shift.@db $! Tons. ", "restrictions" )
THEN TEXT ( "* ", "restrictions" )
THEN TEXT ( "****************** E N D   O F   S H I F T   **********", "restrictions" )
THEN TEXT ( " ", "restrictions" )
endRule

Rule
@name = estimate_init_tonnage_0
IF   rr_any.restriction.existed
THEN IGNORE ( $Rule, "estimate_init_tonnage*" )
endRule

Rule
@name = estimate_init_tonnage_1
IF   rr_any.current_restriction.@string is_not "soft_ore"
THEN rr_any.time_last_rr_disappeared.@float   = AGE ( rr_any.last_restriction.disappeared )
THEN rr_any.time_first_rr_appeared.@float   = AGE ( rr_any.first_restriction.appeared )
THEN rr_any.begin_average.@float   = rr_all.time_tons_average.@float
THEN rr_any.end_average.@float   = timex.sampling_time.@float
ELSE GOTO ( "estimate_init_tonnage_4" )
endRule

Rule
@name = estimate_init_tonnage_2
IF   rr_any.time_last_rr_disappeared.@float   < rr_any.begin_average.@float
THEN rr_any.begin_average.@float   = rr_any.time_last_rr_disappeared.@float
endRule

Rule
@name = estimate_init_tonnage_3
IF   rr_any.begin_average.@float   > rr_any.end_average.@float
THEN fresh_feed.pred_amt.@float   = TIMEAVERAGE ( fresh_feed.real_amt.@float , rr_any.end_average.@float ,
rr_any.begin_average.@float   )
endRule

Rule
@name = estimate_init_tonnage_4
IF   TRUE
THEN rr_adjustment.initial_tonnage.@float   = fresh_feed.pred_amt.@float
THEN rr_adjustment.pred_tonnage_during_rr.@float   = fresh_feed.pred_amt.@float
endRule

Rule
@name = exists_any_restriction
IF   ?any_rr = ANY ( {restriction_id}.restriction.exists is TRUE )
THEN rr_any.restriction.exists is TRUE
ELSE rr_any.restriction.exists is FALSE
```

```
ELSE fresh_feed.pred_amt.@float  = fresh_feed.real_amt.@float
endRule

Rule
@name = exists_rr_any_2
IF  rr_any.restriction.existed is FALSE
AND  rr_any.restriction.exists
AND  rr_any.first_restriction.appeared is FALSE
THEN rr_any.first_restriction.appeared is TRUE
THEN IGNORE ( $Rule, "exists_rr_any_21" )
endRule

Rule
@name = exists_rr_any_21
IF  rr_any.restriction.existed is FALSE
AND  rr_any.restriction.exists
AND  rr_any.first_restriction.appeared is TRUE
THEN rr_any.first_restriction.appeared is FALSE
endRule

Rule
@name = exists_rr_any_3
IF  rr_any.restriction.existed
AND  rr_any.restriction.exists is FALSE
AND  rr_any.last_restriction.disappeared is FALSE
THEN rr_any.last_restriction.disappeared is TRUE
THEN IGNORE ( $Rule, "exists_rr_any_31" )
endRule

Rule
@name = exists_rr_any_31
IF  rr_any.restriction.existed
AND  rr_any.restriction.exists is FALSE
AND  rr_any.last_restriction.disappeared
THEN rr_any.last_restriction.disappeared is FALSE
endRule

Rule
@name = exists_rr_feeder_down_1
@priority = 7
IF  rr_feeder_down.restriction.detected
AND  rr_sag_stops.restriction.detected is FALSE
AND  rr_ffeed_cut.restriction.detected is FALSE
AND  rr_ore_supply.restriction.detected is FALSE
AND  rr_soft_ore.restriction.detected is FALSE
AND  rr_ffeed_limit.restriction.detected is FALSE
THEN rr_feeder_down.restriction.exists is TRUE
ELSE rr_feeder_down.restriction.exists is FALSE
endRule

Rule
@name = exists_rr_feeder_down_2
@priority = 7
IF  rr_feeder_down.restriction.exists is FALSE
AND  rr_feeder_down.restriction.existed
THEN rr_any.current_restriction.@string is "rr_feeder_down"
THEN FREERULE ( $Rule, "timing_rr_final" )
THEN MACRO ( "timing_rr_final" )
endRule

Rule
```

```
@name = exists_rr_ffeed_cut_1
@priority = 8
IF  rr_ffeed_cut.restriction.detected
AND  rr_sag_stops.restriction.detected is FALSE
THEN rr_ffeed_cut.restriction.exists is TRUE
ELSE rr_ffeed_cut.restriction.exists is FALSE
endRule

Rule
@name = exists_rr_ffeed_cut_2
@priority = 8
IF  rr_ffeed_cut.restriction.exists is FALSE
AND  rr_ffeed_cut.restriction.existed
THEN rr_any.current_restriction.@string is "rr_ffeed_cut"
THEN FREERULE ( $Rule, "timing_rr_final" )
THEN MACRO ( "timing_rr_final" )
endRule

Rule
@name = exists_rr_ffeed_limit_1
IF  rr_ffeed_limit.restriction.detected
AND  rr_sag_stops.restriction.exists is FALSE
AND  rr_ffeed_cut.restriction.exists is FALSE
AND  rr_ore_supply.restriction.exists is FALSE
THEN rr_ffeed_limit.restriction.exists is TRUE
ELSE rr_ffeed_limit.restriction.exists is FALSE
endRule

Rule
@name = exists_rr_ffeed_limit_3
IF  rr_ffeed_limit.restriction.exists is FALSE
AND  rr_ffeed_limit.restriction.existed
THEN rr_any.current_restriction.@string is "rr_ffeed_limit"
THEN FREERULE ( $Rule, "timing_rr_final" )
THEN MACRO ( "timing_rr_final" )
endRule

Rule
@name = exists_rr_ore_supply_1
@priority = 6
IF  rr_ore_supply.restriction.detected
AND  rr_sag_stops.restriction.detected is FALSE
AND  rr_ffeed_cut.restriction.exists is FALSE
THEN rr_ore_supply.restriction.exists is TRUE
ELSE rr_ore_supply.restriction.exists is FALSE
endRule

Rule
@name = exists_rr_ore_supply_2
@priority = 6
IF  rr_ore_supply.restriction.exists is FALSE
AND  rr_ore_supply.restriction.existed
THEN rr_any.current_restriction.@string is "rr_ore_supply"
THEN FREERULE ( $Rule, "timing_rr_final" )
THEN MACRO ( "timing_rr_final" )
endRule

Rule
@name = exists_rr_sag_stops_1
@priority = 9
IF  rr_sag_stops.restriction.detected
```

```
THEN rr_sag_stops.restriction.exists is TRUE
ELSE rr_sag_stops.restriction.exists is FALSE
endRule

Rule
@name = exists_rr_sag_stops_2
@priority = 9
IF  rr_sag_stops.restriction.detected is FALSE
AND  rr_sag_stops.restriction.existed
THEN rr_any.current_restriction.@string is "rr_sag_stops"
THEN FREERULE ( $Rule, "timing_rr_final" )
THEN MACRO ( "timing_rr_final" )
endRule

Rule
@name = exists_rr_soft_ore_1
IF  rr_soft_ore.restriction.detected
AND  rr_sag_stops.restriction.detected is FALSE
AND  rr_ffeed_cut.restriction.exists is FALSE
AND  rr_ore_supply.restriction.exists is FALSE
AND  sag_amps.deviation.@float  > 30.000000
THEN rr_soft_ore.restriction.exists is TRUE
ELSE rr_soft_ore.restriction.exists is FALSE
endRule

Rule
@name = exists_rr_soft_ore_3
IF  rr_soft_ore.restriction.exists is FALSE
AND  rr_soft_ore.restriction.existed
THEN rr_any.current_restriction.@string is "rr_soft_ore"
THEN FREERULE ( $Rule, "timing_rr_final" )
THEN MACRO ( "timing_rr_final" )
endRule

Rule
@name = exists_rr_timing_1
@comment = "called from determine_exists_rr"
IF  ?restrictionx = ANY ( {restriction_id}.restriction.exists is TRUE )
THEN MACRO ( "tonnage_final_aborted*" )
THEN rr_any.current_restriction.@string is LISTFIRSTOBJECT ( ?restrictionx )
ELSE IGNORE ( $Rule, "exists_rr_timing*" )
endRule

Rule
@name = exists_rr_timing_2
IF  rr_any.current_restriction.@string is "rr_soft_ore"
THEN FREERULE ( $Rule, "timing_rr_*" )
THEN MACRO ( "timing_rr_indep" )
THEN RUN_PROCEDURE ( sag_mill.model_name.@string )
THEN MACRO ( "timing_rr_indep_elapsed" )
ELSE FREERULE ( $Rule, "timing_rr_*" )
ELSE MACRO ( "timing_rr_indep*" )
endRule

Rule
@name = exists_rr_water_ratio_1
@priority = 10
IF  rr_water_ratio.restriction.detected
THEN rr_water_ratio.restriction.exists is TRUE
THEN rr_any.current_restriction.@string is "rr_water_ratio"
THEN FREERULE ( $Rule, "timing_rr_*" )
```

```
THEN MACRO ( "timing_rr_indep*" )
ELSE rr_water_ratio.restriction.exists is FALSE
endRule

Rule
@name = exists_rr_water_ratio_2
@priority = 10
IF  rr_water_ratio.restriction.detected is FALSE
AND  rr_water_ratio.restriction.existed
THEN rr_any.current_restriction.@string is "rr_water_ratio"
THEN FREERULE ( $Rule, "timing_rr_final" )
THEN MACRO ( "timing_rr_final" )
endRule

Rule
@name = feeders_all_running
IF  ?feeder = ANY ( {feeder}.status.running is FALSE )
THEN feeder_set.status_all.running is FALSE
ELSE feeder_set.status_all.running is TRUE
endRule

Rule
@name = follow_up_begin_rr_0
IF  rr_any.restriction.exists
AND  rr_any.restriction.existed is FALSE
THEN DONOTHING ( )
ELSE IGNORE ( $Rule, "follow_up_begin_rr*" )
endRule

Rule
@name = follow_up_begin_rr_1
@comment = "Instantiated in rule DETERMINE EXISTS RR"
IF  TRUE
THEN rr_adjustment.final_tonnage.in_progress is FALSE
THEN rr_adjustment.number_of_rr.@integer = 0
THEN rr_adjustment.initial_tonnage.@float  = fresh_feed.pred_amt.@float
THEN rr_adjustment.pred_tonnage_during_rr.@float  = fresh_feed.pred_amt.@float
THEN rr_adjustment.final_tonnage.@float  = fresh_feed.pred_amt.@float
THEN rr_adjustment.time_elapsed_total.@float  = 0.000000
THEN rr_adjustment.final_tonnage.in_progress is FALSE
THEN rr_adjustment.ignore_tonnage.correction is FALSE
THEN rr_any.do_not_accumulate.tonnage_current_rr is FALSE
THEN TEXT ( " follow up begin ....", "developer" )
endRule

Rule
@name = follow_up_begin_rr_2
@comment = "Instantiated in rule DETERMINE EXISTS RR"
IF  rr_any.time_last_rr_disappeared.@float  < rr_any.time_no_exists_restriction.@float
THEN rr_adjustment.ignore_tonnage.correction is TRUE
THEN TEXT ( " ADJUSTMENT current restriction IGNORED....", "developer" )
THEN TEXT ( " ADJUSTMENT current restriction IGNORED....", "restrictions" )
endRule

Rule
@name = follow_up_end_rr_1
@comment = "Instantiated from TIME_FINAL"
IF  rr_any.current_restriction.@string is "rr_soft_ore"
OR  rr_any.current_restriction.@string is "rr_sag_stops"
THEN rr_adjustment.ignore_tonnage.correction is TRUE
THEN IGNORE ( $Rule, "follow_up_end_rr*" )
```

```
endRule

Rule
@name = follow_up_end_rr_2
@comment = "Instantiated from TIME_FINAL"
IF  rr_adjustment.ignore_tonnage.correction is FALSE
THEN rr_adjustment.number_of_rr.@integer = rr_adjustment.number_of_rr.@integer + 1
THEN rr_adjustment.\restriction_\<rr_adjustment.number_of_rr.@integer>.@string is rr_any.current_restriction.@string
THEN rr_adjustment.\cause_\<rr_adjustment.number_of_rr.@integer>.@string is category_any.current_category.@string
THEN rr_adjustment.\time_elapsed_\<rr_adjustment.number_of_rr.@integer>.@float  =
<rr_any.current_restriction.@string>.time_total.@float
THEN rr_adjustment.time_elapsed_total.@float  = rr_adjustment.time_elapsed_total.@float  +
<rr_any.current_restriction.@string>.time_total.@float
THEN TEXT ( "  here in follow up end 2 .... #: !$ rr_adjustment.number_of_rr.@i $! restr: !$ rr_any.current_restriction.@s
$!  cause: !$ category_any.current_category.@s $!", "developer" )
THEN TEXT ( " rr: !$ rr_adjustment.restriction_1.@s $!  cause: !$ rr_adjustment.cause_1.@s $!  time: !$
rr_adjustment.time_elapsed_1.@f $!", "developer" )
ELSE IGNORE ( $Rule, "follow_up_end_rr*" )
endRule

Rule
@name = follow_up_end_rr_3
@comment = "Instantiated as soon as a restriction is found"
IF  rr_adjustment.number_of_rr.@integer > 10
THEN TEXT ( " The mechanism to adjust tonnage estimation detected more than 10  restriction ...!", "restrictions" )
endRule

Rule
@name = fresh_feed_cut_1
IF  sag_mill.status.running
AND  fresh_feed.real_amt.zero
THEN fresh_feed.has_been.cut is TRUE
ELSE fresh_feed.has_been.cut is FALSE
endRule

Rule
@name = fresh_feed_cut_2
IF  fresh_feed.has_been.cut
AND  fresh_feed.cut_time.initialized is FALSE
THEN fresh_feed.cut_time_init.@time = timex.current_timex.@time
THEN fresh_feed.cut_time.initialized is TRUE
endRule

Rule
@name = fresh_feed_cut_3
IF  fresh_feed.has_been.cut
AND  fresh_feed.cut_time.initialized
THEN fresh_feed.cut_time_elapsed.@float  = ELAPSEDTIME ( timex.current_timex.@time,
fresh_feed.cut_time_init.@time )
endRule

Rule
@name = fresh_feed_recovered_1
IF  fresh_feed.was.cut
AND  fresh_feed.real_amt.positive
THEN fresh_feed.has_been.recovered_from_cut is TRUE
THEN fresh_feed.cut_time_final.@time is timex.current_timex.@time
THEN fresh_feed.cut_time_total.@double  = ELAPSEDTIME ( fresh_feed.cut_time_final.@time,
fresh_feed.cut_time_init.@time )
THEN FORGET ( "fresh_feed.cut_time_init.@t" )
THEN FORGET ( "fresh_feed.cut_time.initialized" )
```

```
endRule

Rule
@name = global_manager_1
@priority = 50
IF  TRUE
THEN MACRO ( "run_simulation_again*" )
THEN MACRO ( "initialization_1" )
THEN MACRO ( "sampling_time*" )
THEN RUN_PROCEDURE ( "update_time" )
THEN DISABLEBACKWARDCHAIN ( F )
THEN RUN_PROCEDURE ( "read_data_coordinator" )
THEN RUN_PROCEDURE ( "check_ffeed_controller" )
THEN MACRO ( "amps_model*" )
THEN MACRO ( "detect_rr*" )
THEN DISABLEBACKWARDCHAIN ( T )
THEN MACRO ( "determine_rr_exists*" )
THEN MACRO ( "determine_tonnage_difference" )
THEN MACRO ( "determine_rr_causes*" )
THEN MACRO ( "tonnage_final_average*" )
THEN MACRO ( "end_of_shift*" )
THEN MACRO ( "update_db*" )
THEN RUN_PROCEDURE ( "write_db_coordinator_xxxxx" )
THEN IGNORE ( $Rule, "a*" )
THEN IGNORE ( $Rule, "c*" )
THEN IGNORE ( $Rule, "d*" )
THEN IGNORE ( $Rule, "e*" )
THEN IGNORE ( $Rule, "f*" )
THEN IGNORE ( $Rule, "i*" )
THEN IGNORE ( $Rule, "l*" )
THEN IGNORE ( $Rule, "q*" )
THEN IGNORE ( $Rule, "r*" )
THEN IGNORE ( $Rule, "t*" )
THEN IGNORE ( $Rule, "u*" )
endRule

Rule
@name = initialization_1
IF  system.already.configured
THEN DONOTHING ( )
ELSE RUN_PROCEDURE ( "system_initialization" )
ELSE system.already.configured is TRUE
endRule

Rule
@name = initialize_causes
IF  ?causex = CLASSTOLIST ( "restriction_category" )
THEN ALL ( ?causex.current_restriction.exists is FALSE )
endRule

Rule
@name = qmss_table3
IF  <dummy.model_name.@string>.input.@double  <= <dummy.model_name.@string>.input_m.@float
THEN RUN_PROCEDURE ( "qmss_table3_low" )
ELSE RUN_PROCEDURE ( "qmss_table3_high" )
endRule

Rule
@name = qual_status_p_cnt
@priority = 0
IF  ?variable = ANY ( {process_var}.pred_dyn.constant is TRUE )
```

```
THEN ALL ( ?variable.p_dyn_status.@string is "constant" )
endRule

Rule
@name = qual_status_p_dec
@priority = 0
IF  ?variable = ANY ( {process_var}.pred_dyn.decreasing is TRUE )
THEN ALL ( ?variable.p_dyn_status.@string is "decreasing" )
endRule

Rule
@name = qual_status_p_high
@priority = 0
IF  ?variable = ANY ( {process_var}.pred_amt.high is TRUE )
THEN ALL ( ?variable.pred_amt_status.@string is "high" )
endRule

Rule
@name = qual_status_p_inc
@priority = 0
IF  ?variable = ANY ( {process_var}.pred_dyn.increasing is TRUE )
THEN ALL ( ?variable.p_dyn_status.@string is "increasing" )
endRule

Rule
@name = qual_status_p_low
@priority = 0
IF  ?variable = ANY ( {process_var}.pred_amt.low is TRUE )
THEN ALL ( ?variable.pred_amt_status.@string is "low" )
endRule

Rule
@name = qual_status_p_medium
@priority = 0
IF  ?variable = ANY ( {process_var}.pred_amt.medium is TRUE )
THEN ALL ( ?variable.pred_amt_status.@string is "medium" )
endRule

Rule
@name = qual_status_real_cnt
@priority = 0
IF  ?variable = ANY ( {process_var}.real_dyn.constant is TRUE )
THEN ALL ( ?variable.real_dyn_status.@string is "constant" )
endRule

Rule
@name = qual_status_real_dec
@priority = 0
IF  ?variable = ANY ( {process_var}.real_dyn.decreasing is TRUE )
THEN ALL ( ?variable.real_dyn_status.@string is "decreasing" )
endRule

Rule
@name = qual_status_real_high
@priority = 0
IF  ?variable = ANY ( {process_var}.real_amt.high is TRUE )
THEN ALL ( ?variable.real_amt_status.@string is "high" )
endRule

Rule
@name = qual_status_real_inc
```

```
@priority = 0
IF  ?variable = ANY ( {process_var}.real_dyn.increasing is TRUE )
THEN ALL ( ?variable.real_dyn_status.@string is "increasing" )
endRule

Rule
@name = qual_status_real_low
@priority = 0
IF  ?variable = ANY ( {process_var}.real_amt.low is TRUE )
THEN ALL ( ?variable.real_amt_status.@string is "low" )
endRule

Rule
@name = qual_status_real_medium
@priority = 0
IF  ?variable = ANY ( {process_var}.real_amt.medium is TRUE )
THEN ALL ( ?variable.real_amt_status.@string is "medium" )
endRule

Rule
@name = rr_just_finished
IF  rr_any.restriction.exists is FALSE
AND  rr_any.restriction.existed
THEN rr_any.do_not_accumulate.tonnage_current_rr is FALSE
endRule

Rule
@name = run_simulation_again_1
@priority = 10
IF  TRUE
THEN GETPOINTDATA ( "runx.simulation.again" )
THEN FREERULE ( $Rule, "run_simulation_again_2" )
THEN GOTO ( "run_simulation_again_2" )
endRule

Rule
@name = run_simulation_again_2
@priority = 10
IF  runx.simulation.again
THEN DONOTHING ( )
ELSE FREERULE ( $Rule, "run_simulation_again_1" )
ELSE GOTO ( "run_simulation_again_1" )
endRule

Rule
@name = sampling_time_1
IF  TRUE
THEN timex.since_last_sampling.@integer = AGE ( timex.last_sampling.instant )
endRule

Rule
@name = sampling_time_2
IF  timex.since_last_sampling.@integer < timex.sampling_time.@float
THEN dummy.integer.@integer = timex.sampling_time.@float  - timex.since_last_sampling.@integer
THEN SLEEP ( dummy.integer.@integer )
ELSE TEXT ( "WARNING! ... It took longer than sampling time ( !$ timex.since_last_sampling.@i $! secs.) ", "developer"
)
endRule

Rule
@name = sampling_time_3
```

```
IF  timex.last_sampling.instant
THEN timex.last_sampling.instant is FALSE
ELSE timex.last_sampling.instant is TRUE
endRule

Rule
@name = test
IF  GETPOINTDATA ( "feeder1.status.@float" )
THEN TEXT ( " feeder1: !$ feeder1.status.@float $! successful", "rr_indep" )
ELSE TEXT ( " feeder1: !$ feeder1.status.@float $! .. it could not read feeder1", "rr_indep" )
endRule

Rule
@name = test2
IF  GETPOINTDATA ( "fresh_feed.real_amt.@float" )
THEN TEXT ( " fresh-feed: !$ fresh_feed.real_amt.@f $! successful", "rr_indep" )
ELSE TEXT ( " else fresh - feed: !$ feeder2.status.@float $! .. it could not read fresh feed", "rr_indep" )
endRule

Rule
@name = test3
IF  ?feed = ANY ( {feeder}.status.@float  == 1 )
THEN ALL ( ?feed.status.running is TRUE )
endRule

Rule
@name = test4
IF  ?feed = ANY ( {feeder}.status.@float  == 0 )
THEN ALL ( ?feed.status.running is FALSE )
endRule

Rule
@name = test5
IF  TRUE
THEN dummy.a.@float  = 1
endRule

Rule
@name = test6
IF  TRUE
THEN a.c.@float  = TIMEAVERAGE ( fresh_feed.real_amt.@float , rr_speed.time_elapsed.@float , (
rr_speed.time_elapsed.@float  + 30 ) )
THEN a.d.@float  = TIMEAVERAGE ( fresh_feed.real_amt.@float , 0, rr_speed.time_elapsed.@float  )
THEN TEXT ( " ffeed prior= !$ a.c.@f $! .... now: !$ a.d.@f $! ....", "rr_dep" )
endRule

Rule
@name = test7
IF  ?listx = CLASSTOLIST ( "restriction_category" )
THEN ALL ( ?listx.current_restriction.exists is FALSE )
THEN soft_ore.current_restriction.exists is TRUE
endRule

Rule
@name = test70
IF  ?list1 = ANY ( {restriction_category}.current_restriction.exists is TRUE )
THEN ?list2 = LISTADDOBJECT ( ?list1, "ball_miiills" )
THEN dummy.integer.@integer = 1
THEN a.b_1.@float  = 0.000000
THEN a.\b_\<dummy.integer.@integer>.@float  = 12.000000
THEN TEXT ( "result:  !$ a.b_1.@f $!" )
```

```
THEN dummy.string.@string is LISTFIRSTOBJECT ( ?list2 )
THEN TEXT ( " !$ dummy.string.@s $! " )
endRule

Rule
@name = test71
IF  ?listx = ANY ( {restriction_category}.current_restriction.exists is TRUE )
THEN dummy.string.@string is LISTALLOBJECT ( ?list1, "", "" )
THEN TEXT ( "trying" )
THEN TEXT ( "init  !$ dummy.string.@s $! " )
THEN ?listx = LISTADDOBJECT ( ?listx, "ball_mills" )
THEN ?listy = LISTMERGE ( ?list1, ?listx )
THEN a.a1.@string is LISTALLOBJECT ( ?listy, "", "" )
THEN TEXT ( "result: !$ a.a1.@s $! " )
THEN a.a1.@string is LISTFIRSTOBJECT ( ?listy )
THEN TEXT ( " !$ a.a1.@s $! " )
endRule

Rule
@name = test80
IF  TRUE
THEN a.c1.@integer = 1
endRule

Rule
@name = test81
IF  TRUE
THEN a.\b_\<a.c1.@integer>.@float  = 15.400000
THEN TEXT ( " the result is !$ a.b_1.@f$!" )
endRule

Rule
@name = test90
IF  ?listx = CLASSTOLIST ( "restriction_id" )
THEN a.c1.@integer = LISTNUMOBJECT ( ?listx )
THEN dummy.integer.@integer = 1
endRule

Rule
@name = test91
IF  dummy.integer.@integer <= a.c1.@integer
THEN dummy.string.@string is LISTNTHOBJECT ( ?listx, dummy.integer.@integer )
THEN FREERULE ( $Rule, "test92" )
THEN GOTO ( "test92" )
endRule

Rule
@name = test92
IF  TRUE
THEN dummy.integer.@integer = dummy.integer.@integer + 1
THEN FREERULE ( $Rule, "test91" )
THEN GOTO ( "test91" )
endRule

Rule
@name = timing_rr_dep
IF  <rr_any.current_restriction.@string>.restriction.initialized is FALSE
THEN <rr_any.current_restriction.@string>.time_initial.@time is rr_speed.time_initial.@time
THEN <rr_any.current_restriction.@string>.restriction.initialized is TRUE
THEN TEXT ( "restriction !$ rr_any.current_restriction.@s $! detected at !$ timex.current_timex.@t $! ", "restrictions" )
THEN <rr_any.current_restriction.@string>.possible_cause.determined is FALSE
```

```
THEN FREERULE ( $Rule, "estimate_init_tonnage*" )
THEN MACRO ( "estimate_init_tonnage*" )
THEN <rr_any.current_restriction.@string>.lost_current_rr.@double = rr_speed.lost_current_rr.@double
THEN MACRO ( "follow_up_rr*" )
endRule

Rule
@name = timing_rr_dep_elapsed
IF  <rr_any.current_restriction.@string>.restriction.initialized
THEN <rr_any.current_restriction.@string>.time_elapsed.@float  = rr_speed.time_elapsed.@float
THEN <rr_any.current_restriction.@string>.hour_elapsed.@integer = rr_speed.hour_elapsed.@integer
THEN <rr_any.current_restriction.@string>.minute_elapsed.@integer = rr_speed.minute_elapsed.@integer
THEN <rr_any.current_restriction.@string>.second_elapsed.@integer = rr_speed.second_elapsed.@integer
THEN <rr_any.current_restriction.@string>.lost_current_rr.@double = rr_speed.lost_current_rr.@double
endRule

Rule
@name = timing_rr_final
IF  TRUE
THEN <rr_any.current_restriction.@string>.time_final.@time is timex.current_timex.@time
THEN <rr_any.current_restriction.@string>.time_total.@float  = <rr_any.current_restriction.@string>.time_elapsed.@float
THEN <rr_any.current_restriction.@string>.restriction.initialized is FALSE
THEN TEXT ( "restriction !$ rr_any.current_restriction.@s $! is over at !$
<rr_any.current_restriction.@string>.time_final.@time $!", "restrictions" )
THEN TEXT ( " - Total time: !$ <rr_any.current_restriction.@string>.hour_elapsed.@i
$!:!$<rr_any.current_restriction.@string>.minute_elapsed.@i$!:!$<rr_any.current_restriction.@string>.second_elapsed.@i
$!", "restrictions" )
THEN TEXT ( "Tonnage lost: !$ <rr_any.current_restriction.@string>.lost_current_rr.@db $! Tons.", "restrictions" )
THEN FREERULE ( $Rule, "accumulate_losses*" )
THEN MACRO ( "accumulate_losses*" )
THEN FREERULE ( $Rule, "follow_up_end_rr*" )
THEN MACRO ( "follow_up_end_rr*" )
endRule

Rule
@name = timing_rr_indep
IF  <rr_any.current_restriction.@string>.restriction.initialized is FALSE
THEN <rr_any.current_restriction.@string>.time_initial.@time is timex.current_timex.@time
THEN <rr_any.current_restriction.@string>.restriction.initialized is TRUE
THEN <rr_any.current_restriction.@string>.possible_cause.determined is FALSE
THEN <rr_any.current_restriction.@string>.time_elapsed.@float  = 0.000000
THEN <rr_any.current_restriction.@string>.lost_current_rr.@double = 0.000000
THEN TEXT ( " ", "restrictions" )
THEN TEXT ( "restriction !$ rr_any.current_restriction.@string $! detected at !$
<rr_any.current_restriction.@string>.time_initial.@time $! ", "restrictions" )
THEN FREERULE ( $Rule, "estimate_init_tonnage*" )
THEN MACRO ( "estimate_init_tonnage*" )
endRule

Rule
@name = timing_rr_indep_elapsed
IF  <rr_any.current_restriction.@string>.restriction.initialized
THEN dummy.time_previous.@float  = <rr_any.current_restriction.@string>.time_elapsed.@float
THEN <rr_any.current_restriction.@string>.time_elapsed.@float  = AGE (
<rr_any.current_restriction.@string>.restriction.exists )
THEN <rr_any.current_restriction.@string>.time_delta_elapsed.@float  =
<rr_any.current_restriction.@string>.time_elapsed.@float  - dummy.time_previous.@float
THEN <rr_any.current_restriction.@string>.hour_elapsed.@integer =
<rr_any.current_restriction.@string>.time_elapsed.@float  / 3600.000000
THEN dummy.remainder.@float  = <rr_any.current_restriction.@string>.time_elapsed.@float  -
<rr_any.current_restriction.@string>.hour_elapsed.@integer * 3600
```

```
THEN <rr_any.current_restriction.@string>.minute_elapsed.@integer = dummy.remainder.@float / 60.000000
THEN <rr_any.current_restriction.@string>.second_elapsed.@integer = dummy.remainder.@float -
<rr_any.current_restriction.@string>.minute_elapsed.@integer * 60
THEN FREERULE ( $Rule, "tonnage_lost_lower" )
THEN MACRO ( "tonnage_lost_lower" )
endRule

Rule
@name = tonnage_adjustment_0
IF TRUE
THEN rr_adjustment.change_in_final_tonnage.@float = rr_adjustment.final_tonnage.@float -
rr_adjustment.pred_tonnage_during_rr.@float
THEN rr_adjustment.change_in_initial_tonnage.@float = rr_adjustment.initial_tonnage.@float -
rr_adjustment.pred_tonnage_during_rr.@float
THEN rr_adjustment.rr_being_adjusted.@integer = 1
THEN ASNCERTAINTY ( rr_adjustment.rr_being_adjusted.@integer, 100 )
THEN ASNCERTAINTY ( rr_adjustment.number_of_rr.@integer, 100 )
THEN rr_adjustment.change_in_tonnage_previous.@float = 0.000000
THEN dummy.time_accumulated.@float = 0.000000
endRule

Rule
@name = tonnage_adjustment_1
IF rr_adjustment.rr_being_adjusted.@integer <= rr_adjustment.number_of_rr.@integer
THEN dummy.time_accumulated.@float = dummy.time_accumulated.@float +
rr_adjustment.\time_elapsed_\<rr_adjustment.rr_being_adjusted.@integer>.@float
THEN rr_adjustment.change_in_tonnage.@float = dummy.time_accumulated.@float /
rr_adjustment.time_elapsed_total.@float * rr_adjustment.change_in_final_tonnage.@float
THEN dummy.rectangle.@double = rr_adjustment.change_in_initial_tonnage.@float *
rr_adjustment.\time_elapsed_\<rr_adjustment.rr_being_adjusted.@integer>.@float
THEN dummy.triangle.@double = rr_adjustment.\time_elapsed_\<rr_adjustment.rr_being_adjusted.@integer>.@float * (
rr_adjustment.change_in_tonnage.@float + rr_adjustment.change_in_tonnage_previous.@float ) / 2.000000
THEN rr_adjustment.correction_tons_accumulated.@double = ( dummy.rectangle.@double + dummy.triangle.@double )
/ 3600.000000
THEN rr_any.current_restriction.@string is
rr_adjustment.\restriction_\<rr_adjustment.rr_being_adjusted.@integer>.@string
THEN category_any.current_category.@string is
rr_adjustment.\cause_\<rr_adjustment.rr_being_adjusted.@integer>.@string
THEN FREERULE ( $Rule, "accumulate_adjusted*" )
THEN MACRO ( "accumulate_adjusted*" )
THEN FREERULE ( $Rule, "tonnage_adjustment_2" )
THEN GOTO ( "tonnage_adjustment_2" )
ELSE IGNORE ( $Rule, "tonnage_adjustment*" )
endRule

Rule
@name = tonnage_adjustment_2
IF TRUE
THEN rr_adjustment.rr_being_adjusted.@integer = rr_adjustment.rr_being_adjusted.@integer + 1
THEN rr_adjustment.change_in_tonnage_previous.@float = rr_adjustment.change_in_tonnage.@float
THEN FREERULE ( $Rule, "tonnage_adjustment_1" )
THEN GOTO ( "tonnage_adjustment_1" )
endRule

Rule
@name = tonnage_final_aborted_1
@comment = "called from ???"
IF rr_adjustment.final_tonnage.was_in_progress
AND rr_any.time_last_rr_disappeared.@float >= rr_any.time_no_exists_restriction.@float
THEN MACRO ( "tonnage_adjustment*" )
THEN TEXT ( "final aborted ..... an rr BEFORE time is up!", "developer" )
```

```
endRule

Rule
@name = tonnage_final_aborted_2
@comment = "called from ???"
IF  rr_adjustment.final_tonnage.was_in_progress
AND  rr_any.time_last_rr_disappeared.@float  < rr_any.time_no_exists_restriction.@float
THEN dummy.integer.@integer = rr_any.time_no_exists_restrictions.@float  / 60
THEN TEXT ( "A restriction appeared BEFORE !$ dummy.integer.@i $! min; Lost tonnage adjustment IGNORED!",
"restrictions" )
endRule

Rule
@name = tonnage_final_average_0
IF  rr_adjustment.ignore_tonnage.correction
OR  rr_adjustment.do_not_correct.tonnage_lost
OR  rr_any.do_not_accumulate.tonnage_current_rr
OR  rr_any.stop_accumulating.tonnage_lost
THEN IGNORE ( $Rule, "tonnage_final_average*" )
endRule

Rule
@name = tonnage_final_average_1
IF  rr_any.restriction.exists is FALSE
AND  rr_any.restriction.existed
THEN rr_adjustment.final_tonnage.in_progress is TRUE
endRule

Rule
@name = tonnage_final_average_2
IF  rr_adjustment.final_tonnage.in_progress
AND  AGE ( rr_any.last_restriction.disappeared ) <= rr_adjustment.maximum_time.@float
THEN rr_adjustment.final_tonnage.@float  = TIMEAVERAGE ( fresh_feed.real_amt.@float  , 0, AGE (
rr_any.last_restriction.disappeared ) )
endRule

Rule
@name = tonnage_final_average_3
IF  rr_adjustment.final_tonnage.in_progress
AND  AGE ( rr_any.last_restriction.disappeared ) >= rr_adjustment.maximum_time.@float
THEN rr_adjustment.final_tonnage.in_progress is FALSE
THEN IGNORE ( $Rule, "tonnage_final_average*" )
THEN MACRO ( "tonnage_adjustment*" )
endRule

Rule
@name = tonnage_lost_lower
IF  fresh_feed.pred_amt.@float  > fresh_feed.real_amt.@float
THEN dummy.current_loss.@double  = <rr_any.current_restriction.@string>.lost_current_rr.@double
THEN <rr_any.current_restriction.@string>.lost_current_rr.@double  = dummy.current_loss.@double  + (
fresh_feed.pred_amt.@float  - fresh_feed.real_amt.@float  ) *
<rr_any.current_restriction.@string>.time_delta_elapsed.@float  / 3600.000000
endRule

Rule
@name = update_db_correct_tonnage
IF  rr_adjustment.do_not_correct.tonnage_lost
THEN rr_adjustment.did_not_correct.tonnage_lost is TRUE
ELSE rr_adjustment.did_not_correct.tonnage_lost is FALSE
endRule
```

```
Rule
@name = update_db_equipment_1
IF  TRUE
THEN ?bmill = ANY ( {ball_mill}.status.running )
THEN ALL ( ?bmill.status.was_running is TRUE )
THEN ?feedx = ANY ( {feeder}.status.running )
THEN ALL ( ?feedx.status.was_running is TRUE )
endRule

Rule
@name = update_db_equipment_2
IF  ?bmill = ANY ( {ball_mill}.status.running is FALSE )
THEN ALL ( ?bmill.status.was_running is FALSE )
endRule

Rule
@name = update_db_equipment_3
IF  ?feedx = ANY ( {feeder}.status.running is FALSE )
THEN ALL ( ?feedx.status.was_running is FALSE )
endRule

Rule
@name = update_db_fresh_feed_cut
IF  fresh_feed.has_been.cut
THEN fresh_feed.was.cut is TRUE
ELSE fresh_feed.was.cut is FALSE
endRule

Rule
@name = update_db_restrictions_1
IF  ?restriction = ANY ( {restriction_id}.restriction.exists is FALSE )
THEN ALL ( ?restriction.restriction.existed is FALSE )
THEN ALL ( ?restriction.restriction.initialized is FALSE )
THEN ALL ( ?restriction.time_elapsed.@float  = 0.000000 )
THEN ALL ( ?restriction.hour_elapsed.@integer = 0 )
THEN ALL ( ?restriction.minute_elapsed.@integer = 0 )
THEN ALL ( ?restriction.second_elapsed.@integer = 0 )
THEN ALL ( ?restriction.restriction.has_been_identified is FALSE )
endRule

Rule
@name = update_db_restrictions_11
IF  ?restrictionx = ANY ( {restriction_id}.restriction.exists is TRUE )
THEN ALL ( ?restrictionx.restriction.existed is TRUE )
endRule

Rule
@name = update_db_restrictions_2
IF  rr_any.restriction.exists
THEN rr_any.restriction.existed is TRUE
ELSE rr_any.restriction.existed is FALSE
endRule

Rule
@name = update_db_restrictions_3
IF  rr_speed.restriction.detected is TRUE
THEN rr_speed.restriction.detected_past_sampling is TRUE
ELSE rr_speed.restriction.detected_past_sampling is FALSE
endRule

Rule
```

```
@name = update_db_restrictions_4
IF  rr_adjustment.final_tonnage.in_progress
THEN rr_adjustment.final_tonnage.was_in_progress is TRUE
ELSE rr_adjustment.final_tonnage.was_in_progress is FALSE
endRule

Rule
@name = variables_high
IF  ?v_high = ANY ( {process_var}.real_amt.high )
THEN DONOTHING ( )
endRule

Rule
@name = variables_high_2
IF  tail_box1.level.high
OR  tail_box2.level.high
THEN DONOTHING ( )
endRule

Rule
@name = water_ratio_1
IF  fresh_feed.real_amt.@float  < 1700
THEN water_ratio.pred_amt.@float  = 0.650000
ELSE water_ratio.pred_amt.@float  = 0.650000 - 0.150000 / 300.000000 * ( fresh_feed.real_amt.@float  - 1700.000000 )
endRule

Facets
@triplet = {restriction_id}.restriction.has_been_identified
@default = 0.000000
endFacets

Facets
@triplet = ball_mill_1.status.@float
@ifchange = equipment_status
endFacets

Facets
@triplet = ball_mill_2.status.@float
@ifchange = equipment_status
endFacets

Facets
@triplet = dummy.number_of_rr.@integer
@ifchange = max_certainty
endFacets

Facets
@triplet = dummy.rr_analyzed.@integer
@ifchange = max_certainty
endFacets

Facets
@triplet = fresh_feed.agrees_with.setpoint
@fuzzy = agrees_with_ffeed
endFacets

Facets
@triplet = fresh_feed.is_approaching_to.setpoint
@default = 0.000000
endFacets
```

Facets
@triplet = fresh_feed.is_close_to.setpoint
@fuzzy = close_to_ffeed
endFacets

Facets
@triplet = fresh_feed.real_amt.positive
@fuzzy = positive_fresh_feed
endFacets

Facets
@triplet = fresh_feed.real_amt.zero
@fuzzy = zero_fresh_feed
endFacets

Facets
@triplet = fresh_feed.real_dyn.constant
@fuzzy = constant_ffeed
endFacets

Facets
@triplet = fresh_feed.real_dyn.decreasing
@fuzzy = decreasing_ffeed
endFacets

Facets
@triplet = fresh_feed.real_dyn.increasing
@fuzzy = increasing_ffeed
endFacets

Facets
@triplet = level_tailing_box.real_amt.high
@fuzzy = level_tailings_high
endFacets

Facets
@triplet = rr_adjustment.number_of_rr.@integer
@ifchange = max_certainty
endFacets

Facets
@triplet = rr_adjustment.rr_being_adjusted.@integer
@ifchange = max_certainty
endFacets

Facets
@triplet = rr_ffeed_limit.restriction.exists
@default = 0.000000
endFacets

Facets
@triplet = rr_ffeed_limit.restriction.has_been_identified
@default = 0.000000
endFacets

Facets
@triplet = sag_mill.status.@float
@ifchange = equipment_status
endFacets

Facets

```
@triplet = sag_speed.real_amt.positive
@fuzzy = positive_sag_speed
endFacets

Facets
@triplet = sag_speed.real_amt.zero
@fuzzy = zero_sag_speed
endFacets

Facets
@triplet = tail_box1.level.high
@fuzzy = tail_box1_high
endFacets

Facets
@triplet = tail_box2.level.high
@fuzzy = tail_box2_high
endFacets

!*** LoadStrategy must go at the end of the Knowledge Base ***!
LoadStrategy
@name = "rr-hvc.stg"
EndLoadStrategy
```