Preprints / 2nd IFAC Workshop on Computer Software Structures Integrating AI/KBS Systems in Process Control, August 10-12, 1994, Lund, Sweden

Årzén, Karl-Erik

1994

Document Version:
Publisher's PDF, also known as Version of record

IFAC

Preprints

2nd IFAC Workshop on Computer Software Structures Integrating AI/KBS Systems in Process Control

August 10-12, 1994
Lund, Sweden

IFAC

International Federation
of Automatic Control

Preprints

2nd IFAC Workshop on Computer Software Structures Integrating AI/KBS Systems in Process Control

August 10-12, 1994
Lund, Sweden

Organized by

Department of Automatic Control
Lund Institute of Technology

on behalf of the

Swedish IFAC Committee

Sponsored by

IFAC Technical Committee on Applications

Co-sponsored by

IFAC Technical Committee on Computers
IEEE Control Systems Society
SAIS Swedish Artificial Intelligence Society

Financially supported by

Swedish Research Council for Engineering Sciences (TFR)
Lund Institute of Technology
ABB, Asea-Brown Boveri AB
Gensym Corporation

International Program Committee

Chairman: K.-E. Årzén (S)
K. J. Åström (S)
D. Birdwell (USA)
H. Bersini (B)
A. Van Cauwenberghe (B)
M. Fjeld (N)
R. Fjellheim (N)
C. C. Hang (SGP)
K. Hangos (H)
E. Hollnagel (DK)
Y. Ishida (J)
B. Kuipers (USA)
R. Leitch (UK)
M. Lind (DK)
T. McAvoy (USA)
J. Morris (UK)
L. Mõtus (EST)
G. Pang (CAN)
M. Rodd (UK)
G. Suski (USA)
J. C. Taunton (UK)
T. Terano (J)
A. Titli (F)
V. Venkatasubramanian (USA)
H. B. Verbruggen (NL)
C. Weisang (FRG)
E. Woods (N)
H.-J. Zimmermann (FRG)

National Organizing Committee

K.-E. Årzén, Chairman, Lund Institute of Technology
E. Dagnegård, Lund Institute of Technology
T. Hägglund, Lund Institute of Technology
C. Rytoft, Asea Brown Boveri AB
B. Wittenmark, Lund Institute of Technology

Cover Photo: The Lund University main building designed by Helgo Zettervall and inaugurated in 1882.

Table of Contents

Wednesday, August 10

Plenary Address 1

Software Integration

Knowledge Representation/Modelling

Poster Session 1

Other Techniques

Thursday, August 11

Plenary Address 2

Control

Operator Support

Poster Session 2

Friday, August 12

Applications

* Paper not available at the time for printing

Monitoring and Diagnosis

Author Index

* Paper not available at the time of printing

Towards Integrated Process Supervision: Current Status and Future Directions

Venkat Venkatasubramanian
Laboratory for Intelligent Process Systems, School of Chemical Engineering,
Purdue University, W. Lafayette, IN 47907, USA.

Abstract Process supervision deals with tasks that are executed to operate a process plant safely and economically. These tasks can be classified as data acquisition, regulatory control, monitoring, data reconciliation, fault diagnosis, supervisory control, scheduling and planning. While these operational tasks may be intrinsically different from each other, they are, however, closely related and can not be treated in isolation. Hence, there exists a clear need for an integrated framework so that the operational decision-making can be made more comprehensively and effectively. While such an integrated approach is very compelling and desirable, achieving it is no simple task as there are many challenges in realizing integration. In this paper, we review these challenges and indentify the underlying issues which need to be addressed for achieving an integrated approach to process supervision. We discuss the role of artificial intelligence in this context and how it provides a problem-solving platform for integration. We also survey the current status of automated approaches to operations and conclude with some thoughts on future directions.

Key Words. Artificial intelligence; Failure detection; Integrated plant control; Monitoring

1. INTRODUCTION

Integrated process supervision is the overall, coordinated, management of different operational tasks in a process plant. These operational tasks can be hierarchically categorized as data acquisition, regulatory control, monitoring, fault diagnosis, supervisory control, scheduling and planning. The lower level of the hierarchy involves layers that deal with tasks such as data acquisition and regulatory control. At the intermediate level, one has layers for the tasks of monitoring, data reconciliation, diagnosis, and supervisory control. At a higher level, one has the layers that perform plant-wide optimization, scheduling and planning of process operations. At the lower level, the perspective is local in character, like that of a regulatory controller which is limited to implementing a functional relationship (e.g. the control law) between the manipulated and controlled variables. The intermediate level is concerned with coordination between units, unit optimization and monitoring of production and operating constraints. It also performs fault diagnosis and suggests recovery from these malfunctions. At the higher level, the perspective is more global in character, like that of planning of process operations.

The overall problem of integrated process supervision involves several subproblem areas that are related to each other and can not really be treated as individual problems in isolation. For example, low-level events such as controller failure or some other equipment malfunction, can have a significant impact on the higher-level plans by calling for the revision of planned schedules. Likewise, higher level decisions have a serious impact on lower level activities such as supervisory and regulatory control. In the case of data reconciliation, traditionally one does not consider parameter drifts and structural faults as part of the problem. However, an integrated view is necessary for reconciliation of measured data in the presence of process faults. Thus, while these operational tasks may be intrinsically different from each other, they are, however, closely related to each other and can not be treated as isolated tasks. Hence, we need an approach wherein all these different tasks can be integrated into a single unified framework and so that the operational decision-making can be made more comprehensively and more effectively.

Over the years, a variety of tools and techniques have been developed to address these tasks. They include process modeling and simulation techniques, large scale linear and nonlinear optimization methods, advanced model-based and knowledge-based process control techniques, model-based data reconciliation methods, and statistical, neural net-based and knowledge-based fault detection and diagnosis methods. While no single tool or technique can solve the entire process supervision problem by itself, the proliferation of disparate tools imposes barriers to task integration by fragmenting system

implementation as well as the solution process. Such fragmentation also impedes the understanding of the results and complicates their communication and implementation. This is one of the key challenges towards integration.

While an integrated approach is very compelling and desirable, achieving it is no simple task as there are many challenges in realizing integration. The intent of this survey paper is not so much to provide the answers but to try to identify the key challenges faced, the related fundamental issues, and review the current status and the emerging trends. In this perspective, we will also examine the role of artificial intelligence in integration. Since the scope of this exercise is very broad, we will mainly focus our discussion on the integration of low-level and intermediate-level tasks, namely, the tasks of regulatory control, data reconciliation, monitoring, diagnosis, and supervisory control in this paper.

2. PROBLEM SOLVING PARADIGMS IN PROCESS SUPERVISION

The common problem-solving paradigms that underlie integrated process supervision can be categorized as pattern recognition and classification, symbolic reasoning, and optimization. Many of the taks in process supervision can be handled in different ways. For example, process fault detection can be treated as a statistical classification problem where one tests a measurement against a null hypothesis that the process is normal. Alternatively, by considering a fault to be a parameter disturbance, fault diagnosis can be treated as a parameter estimation problem [Isermann, 1984]. Yet another view is to treat fault diagnosis as a classification of measurement data into fault groups using neural networks. In addition, we also have qualitative methods for fault diagnosis that use causal models of the process to search for the source of abnormality. This is a symbolic reasoning problem. Thus, the use of quite different solution methodologies for the same problem poses serious challenges towards integration. Since this is a central issue in integration, a better understanding of these problem solving paradigms in terms of their domain of application, types of problems that can be solved using these techniques, advantages and disadvantages is essential. To this end, a brief overview of these various problem-solving paradigms is provided in this section.

2.1. Pattern classification approach:

Syntactic pattern recognition is concerned with classifying symbolic information into a given set of classes. The classification task may be guided by a set of rules or grammar that defines the membership relationships or mapping between the patterns and the classes. Alternatively, one could specify this guiding information by a causal model (e.g., in the case of diagnosis) or in general by a set of constraints. Statistical pattern classification, on the other hand, is concerned with classifying numeric information into a given set of classes. Many problems in process operations can be categorized into one of these classification tasks. For example, reasoning about the cause of an abnormality in a process behavior can be considered as a syntactic or statistical pattern classification problem:

• Classifying sensor measurements into one of the fault classes. This is considered as a syntactic classification problem when the reasoning is based on causal models. It is statistical classification, when the numeric values of the measurements are used.

• Classifying temporal trends of sensor measurements into one of the known classes. Time series information of the sensors can be used directly for statistical classification or an abstracted syntactic representation of the measurements for symbolic reasoning purposes.

• Data reconciliation can be posed as a statistical classification problem where one tests a measurement against a null hypothesis to detect any gross sensor faults. In the absence of any gross errors, the data is then rectified to reduce the effect of random noise.

• In modeling for control, composite models can be developed by using classification. For example, choosing the proper model to use can be decided based on the operating regime the process is in and this can be solved as a pattern classification problem.

2.2. Symbolic reasoning approach:

In symbolic reasoning, one often addresses three different kinds of reasoning. They are abductive, inductive and default reasonings. Abduction is the generation of a hypothetical explanation (or cause) for what has been observed. Unlike simple logical deduction, we can get more than one answer in abductive reasoning. Since there is no general way to decide between alternatives, the best one can do is to find a hypothesis that is most probable. Thus, abduction can be thought of as reasoning where we weigh the evidences in the presence of uncertainty. Searching for the cause of an abnormality in a process system is thus an abductive reasoning. In MODEX2 [Venkatasubramanian and Rich, 1988], a model based expert system for fault diagnosis, abductive reasoning is used to generate hypotheses for the sources of faults. In addition, abduction also provides explanations of how the cause could have resulted in the abnormality observed. Such a facility is useful in providing decision support to plant operators. Use of proper knowledge representation technique matters a great deal in determining the computational effort. Model based reasoning allows for efficient bottom-up abduction by suggesting proper rules to try out. Efficiency of such bottom-up search in abduction is considerable [Charniak and McDermott, 1984].

Early work in learning concentrated on systems for pattern classification and game playing. Inductive learning is the classification of a set of experiences into categories or concepts. Inductive learning is performed when one generalizes or specializes a concept definition learned so that it includes all experiences that belong to the concept and exclude those that do not. The clear definition of a concept or category is rarely simple because of the great variety of experiences and uncertainty (noisy data or observations). For this reason, one prefers an adaptive learning scheme. An example of such an adaptive learning scheme is failure-driven learning. Failure-driven learning is refining a concept from failures of expectations as one accumulates related experiences. The failure of heuristic judgement in detecting a source of malfunction in fault diagnosis can trigger a change in the knowledge (or rule) that resulted in the judgement [Rich and Venkatasubramanian, 1989]. Experiences with abnormalities in a plant can be used to generate rules that relate a set of observations with specific causes. One can refine this experiential knowledge over time by generalizing to successful cases not covered and specializing when exceptions are noticed.

One frequently makes default assumptions on the values of various quantities that are manipulated, with the intention of allowing specific reasons for other values to override the current values (e.g. since the outlet is blocked, the flow is now zero), or of rejecting the default if it leads to an inconsistency (e.g. since the outlet of the tank is blocked, there cannot be a decrease in tank level). A fundamental feature of default reasoning is that it is nonmonotonic. In traditional logic, once a fact is deduced, it is considered to remain true for the rest of the reasoning. This is what one means by monotonic. However, as new evidence arises, often one needs to revise the deduced facts to maintain logical consistency. Let us consider our previous argument where we deduced that the tank level cannot decrease (since the outlet of the tank is blocked). After this deduction, if we get new evidence that the tank has a large leak, we will have to retract the conclusion that the tank level cannot decrease. Such a reasoning where retraction of deductions is allowed is nonmonotonic. Default reasoning or nonmonotonic reasoning is an invaluable tool in dealing with situations where all the information is not available at a time or if one has to reason about many, probably inconsistent, cases simultaneously. Reiter [1987] has shown how default logic can be used for reasoning about multiple faults or causes for an abnormality. Reasoning with assumptions explicitly is a related concept [Kavuri and Venkatasubramanian, 1992).

2.3. Optimization approach:

Optimization problems in process operations such as model identification fall under the continuous case, while problems such as allocation of plant resources requiring discrete decisions are combinatorial optimization problems. For example, plant-wide scheduling and optimization in the continuous case and assignment and allocation of plant resources in cases which require the sharing of manufacturing resources between different products are examples of optimization problems in planning. Other examples are:

- Management of inventories and maintenance planning.
- Online estimation of process model, for optimization and model-based process control, data reconciliation, parameter estimation for fault diagnosis.
- Online prediction of the performance of an operating plant.
- Online optimization of control profiles in batch and continuous operations.

Most of the planning problems which are discrete optimization problems are usually solved off-line and hence one can try to solve really large problems. In contrast, most of the continuous optimization problems have to be solved on-line and hence computational effort becomes an important consideration here. Other concerns include convergence problems in multi-dimensional search spaces and local minima problems in continuous nonlinear optimization problems.

3. INTEGRATED PROCESS SUPERVISION: CHALLENGES AND THE ROLE OF AI

Though an integrated framework is very attractive in terms of the benefits it can provide, there are a number of conceptual and implementational challenges that have to be overcome before an industry-wide following of this approach takes place. This section discusses the key requirements and the role of AI in addressing these challenges.

i. *It is necessary to reason about process operations without assuming accurate models.*

In most cases, plant behavior is not accurately known. Even rigorous models are not adequate to predict plant behavior with satisfying accuracy. Furthermore, configuration of plants change during their lifetime. Process operating conditions may vary with the demands for different products produced in the plant. All of these force the operators to make their operating decisions with approximate models of process behavior. AI provides us with techniques for developing qualitative and approximate models, doing inexact reasoning, etc. to cope with situations such as this.

ii. *It is necessary to reason with incomplete and/or uncertain information about the process.*

Operators often face situations where they receive conflicting information about the status of the process or the various process units. This could be due to faulty sensors, for example. Also, they often deal with situations when all the information needed

about the process may not be available. Thus, operators are forced to reason and make assessments about the process with incomplete and/or uncertain data. Realizing these operational constraints in practice and having a means to handle incompleteness and uncertainty is essential to the decision making process. Again, artificial intelligence techniques play a useful role in handling this requirement of an automated system.

iii. *It is necessary to understand, and hence represent, process behavior at different levels of detail depending on the nature of the task.*

The amount of information that is available to the operator is often sufficient to understand the essentials of the behavior of a process. However, the voluminous data results in an information clutter and the operator is now faced with the task of gleaning the important features he needs from this vast amount of data. Information from process measurements, perhaps over an entire month, needs to be organized so that he can get a more global picture of a section or the overall plant easily. Given the large size of plants and different information requirements of tasks, it is necessary to reason with knowledge at different levels of detail. Reasoning with knowledge at different levels of detail is a difficult task as one has to carefully ensure the consistency of the information at different levels of detail. Given the information clutter, it is inevitable that the operator have some way to look at the required information in a compact way. For example in a process plant, there may be as many as 1500 process variables observed every two seconds for behavior during a selected period [Bailey, 1984]. The trends are displayed on monitors and there can be two, four or eight process variables displayed per screen at any one time. This dictates the need for a hierarchic organization from process subsystems to loop clusters down to single loops. This also emphasizes the need for an automated framework for extracting important qualitative features of process behavior from raw sensor data. Powerful knowledge representation and pattern classification techniques of AI are indispensable for this problem.

iv. *It is necessary to make assumptions about a process when modeling or describing it. One has to ensure the validity and consistency of these assumptions.*

When a process unit is described by a model, the model is constructed based on some assumptions, mostly assumptions of normal behavior. However, in diagnostic applications, these assumptions may be violated. In order to avoid inconsistencies, it is necessary to explicitly consider and change the model and the assumptions during the reasoning process. What is needed is a representation of the process model that can represent the process behavior for a given set of assumptions. It is necessary to explicitly define the underlying assumptions, have a scheme to verify the consistency of these assumptions and choose the process model based on these assumptions.

As an example, consider the problem of controlling a process. The controller configuration, parameters and the control law are determined by the mathematical models of the process and the controller. The success of the control scheme crucially depends upon whether the assumptions that underlie these models are still valid. For example, models assume that the sensors provide accurate information. In the case of a gross fault in the sensor, the controller action not only becomes ineffective but may even cause adverse process behavior. Similarly in a hierarchical model for process operations, the decisions made at a higher level can have significant impact on the lower level implementations and thus their assumptions are crucial. Failing to detect the violation of an assumption can result in a gross disruption of the operations. AI provides us with the framework for treating assumptions explicitly, thereby making the automated system readily alert to assumptions violation.

v. *It is necessary to integrate tasks and solution approaches. This requires integrating different problem-solving paradigms, knowledge representation schemes, and search techniques.*

To effectively provide an integrated framework, one needs to carefully address the knowledge representation and search issues. It is necessary to represent structural, functional and behavioral information about the process. We can think of these as three complementary sources of information each organized hierarchically. One needs to address how the three hierarchies are built and how they interact. One of the key functions of such a knowledge representation is to let one examine the process at any preferred level of detail in any desired hierarchy. For example, for the task of process fault diagnosis, one is concerned with structural information within the individual units and the overall connectivity of the process. For planning tasks, one may take a higher-level perspective on the process plant, lumping many units together as a larger, abstract, input-output module. The different tasks may employ different problem-solving paradigms which, in turn, would call for different representation and search strategies. All of these need to integrated to offer a complete perspective of process operations. Due to the character of the issues involved in here, artificial intelligence plays a crucial role.

vi. *It is necessary to keep the role of an operator primary and active, not secondary and passive, in the operating environment that is managed with the assistance of on-line intelligent systems.*

While it might be acceptable to delegate all the control to computers when we are dealing with regulatory control problems, it might be more risky to do so when it comes to supervisory and higher-level decision-making. This is due to the character of the problems and issues involved as well as due to the limitations of current intelligent systems. In addition, one has other concerns such as the liability and legal aspects of this problem. Thus, it is important to have the operators actively involved in the decision-making process and make the on-line intelligent systems play an advisory role. This is also necessary to keep the operators' skills sharp, as otherwise their skills could deteriorate over time due to their increased dependence on the advisory systems as a crutch. There is a delicate balance that has to be achieved here. Since the operator's role would be primary, this creates special demands on the design of the advisory systems, such as:

- simple, operator-friendly user-interfaces
- emphasis on visual, graphical display of information for ease of understanding
- structured, guided access to data and knowledge about process status and behavior
- explanatory capabilities to offer insights into the systems reasoning and recommendations

Thus, the design of such systems should be operator-centered, with his or her needs and capabilities in mind. Such a perspective places considerable emphasis on man-machine interaction issues and the nature of the user-interface, which are important requirements that will benefit from artificial intelligence techniques.

One can see from this discussion that the use of AI techniques to face these challenges is not only desirable, but also necessary.

4. CURRENT STATUS OF AUTOMATION IN PROCESS SUPERVISION: A BRIEF REVIEW

The main focus of this paper is to address issues in integrated process supervision for the low-level and intermediate-level tasks. As mentioned before, these tasks are: regulatory control, process monitoring, fault diagnosis, data reconciliation and supervisory control. In this section, the current status of automation of these tasks are briefly reviewed.

4.1. Process Monitoring

Process monitoring refers to the task of identifying the state of the system from sensor data. Process trend analysis and prediction are important components of process monitoring. Knowing the current process trends, the causes that drive them, and the possible future evolution of these trends are essential for supervisory decision making. The central issues here are representing and reasoning with temporal evolution of process trends, multi-scale data, sensor noise and data uncertainties, and cause and effect models of process trends. Recent research in this area has shown some promise for integrated supervision applications. Stephanopoulos recognized the importance of process trend representation for higher-level process integration early on and developed a formal framework. This framework handles temporal data, reasonable discontinuous and continuous functions, and the abstraction of semi-quantitative and qualitative trends (Cheung and Stephanopoulos, 1989).

Venkatasubramanian and co-workers (Janusz and Venkatasubramanian, 1991; Rengaswamy and Venkatasubramanian, 1992) developed a similar approach in their qualitative representation scheme. The fundamental element in their representation scheme is the primitive. They use a finite difference method to calculate the first and second derivatives of the process trend changes and based on these values the primitives are identified. For noisy data, neural networks are used to extract the primitives as they are noise-tolerant. The ability to handle noise is incorporated in two stages. At a lower level, the neural net-based pattern classification approach is used to identify the fundamental features of the trends. At a higher level, the syntactic information is abstracted and represented in a hierarchical fashion with an error correcting code smoothing out the errors made at the lower level. Such syntactic approaches are suitable for hierarchical representation of the trend information and for developing error correcting codes, which eliminate the effects of high noise and outliers.

Multilevel abstraction of important events in a process trend is possible through scale-space filtering through the use of a bank of filters each sensitive to certain localized region in the time-frequency domain (Marr and Hidreth, 1980). An example of such a filter that has been extensively used in image processing is $\nabla^2 G$, where G is a Gaussian distribution (Marr and Hidreth, 1980). Another important recent development in this area is the emergence of wavelet-based frameworks. The recent work of Bakshi and Stephanopoulos (1992) using wavelet networks for representing trends shows considerable promise.

4.2. Fault detection and diagnosis

Fault detection and diagnosis is concerned with the detection of abnormal behavior and the identification of their causal origins. Over the recent years, there has been considerable progress towards the automation of fault detection and diagnosis. A general description of fault diagnosis would include the following kinds of abnormalities:

Gross parameter changes in a model
In any modeling, there are processes occurring below the selected level of detail. These unmodeled processes are lumped as parameters. "Parameter faults" arise when there is a disturbance entering the process from the environment through one or more parameters. An example of such a fault is a change in the concentration of the reactant in a reactor feed or the change in the activity of a catalyst.

Structural changes
Structural changes refer to changes in the model itself. They occur due to hard failures in equipment. Structural faults result in a change in the information-flow between various variables. This corresponds to dropping the appropriate model equations and restructuring the other equations to describe the current situation in the process. An example of a structural fault would be a controller failure which would imply that the manipulated variable is no longer functionally dependent on the controlled variable.

Malfunctioning sensors and actuators
Gross errors usually occur in actuators and sensors. There could be a fixed failure, a constant bias (positive or negative) or an out-of-range failure. These are also important faults that need to be identified quickly in view of the fact that some of the instruments might provide feedback signals which are essential for the control of the plant .

The solution strategies for fault diagnosis range from purely qualitative to purely quantitative, with various combinations in between. There are different perspectives from which one can view the problem of fault diagnosis. One can look at the fault diagnosis problem from the perspective of the transformations the measurements go through before arriving at the final solution. Figure 1 shows these transformations.

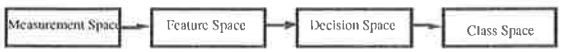


Fig. 1. Transformations in a Fault Diagnostic System

Measurement space is the space of sensor measurements that is available to perform fault diagnosis. Feature space is the space of reduced set of features representative of the measurement space that is developed by transforming the measurement space using *a priori* knowledge about the process. A set of decision variables are developed from the feature space, and class space is the set of integers indexing each individual fault and an additional integer to represent the normal operation of the process and is the final interpretation delivered to the user by the diagnostic system. The transformation from feature space to decision space is performed by a search technique that tries to minimize the mismatch between the actual observations and the observations for different faulty modes, either in a qualitative or quantitative manner. For example, symbolic reasoning is done qualitatively where one tries to minimize the mismatch between the observation (sensor i is low, sensor j is high and so on) and the template for various faulty modes. In contrast, in parameter estimation methods the mismatch is a least squares norm and is minimized by searching in a parameter space that models the various faults. The transformation from the decision space to class space is effected using either thresholding, template matching or symbolic reasoning as the case may be. Hence, the two important components in a diagnostic system are the *a priori* knowledge and the search technique used.

One can view diagnostic systems from either of these two perspectives as well. A general classification of fault diagnostic systems can be done based on the three different solution methods used. They are: Knowledge-based, Analytical model-based, and Pattern recognition-based methods. Each of these diagnostic methods is a combination of a particular type of *a priori* knowledge and a search technique. Knowledge-based systems predominantly use qualitative models of the process in tandem with different search techniques. The quantitative model-based approaches rely on mathematically representing the inconsistencies between the actual and expected behavior as residuals. Pattern recognition is the task of assigning a pattern to one of k predetermined classes. The knowledge about these classes is usually obtained from process history data. Under each of these general solution philosophies there are many combinations of *a priori* knowledge and search techniques and is not possible to enumerate all these combinations. Hence, the attempt here is to provide a flavor for some of these methods.

Knowledge-based methods
In this subsection, knowledge-based techniques are illustrated with the aid of some typical approaches to fault diagnosis.

Hypothesize-and-test Using Causal Models:
There are many approaches to the design of model-based expert systems for chemical process fault diagnosis. One approach is the hypothesis/test strategy. A rigorous approach to hypothesis formulation is the method of O'Shima and coworkers (Iri et. al., 1979; Shiozaki et. al., 1985).

The basic premise of this approach is that, for a fault hypothesis to be considered viable, causal pathways must link the proposed fault origin with all observed abnormal measurements. A signed digraph is used as the representation of the influences between the process variables.

Another class of model-based reasoning methods used for diagnosis use agenda-based search techniques. An example of this approach is MODEX (Rich and Venkatasubramanian,1987), which is a system developed to reason from first-principles model-based knowledge. It extension, MODEX2 (Venkatasubramanian and Rich, 1988), is a two-tiered approach using a compiled knowledge-base at the top tier and a deep-level causal model at the second tier.

Finite State-Space Search of Fault Trees:

Fault trees provide a computational means for combining logic to analyze system faults. To perform consistent diagnosis from fault trees, the trees must comprehensively represent the process causal relationships. To ensure this consistency, causal models in the form of signed digraphs are developed. Causal fault trees are developed from these digraphs (Lapp and Powers, 1977). Fault trees determine causal pathways through which primal events (faults) can propagate through the system to cause the top event (some significant malfunction).

Once a fault-tree is synthesized, the information from it is stored in the form of cut sets. Cut sets exhaustively specify all possible paths in the digraph resulting from the fault. Kramer and Palowitch (1989) developed a method of deriving rules for diagnosing faults from signed digraphs. Ulerich and Powers (1987) used digraph models to include human operator action and failure models due to operator action. They also illustrate how real-time data can be used to infer events in a control loop.

Search in malfunction hypotheses hierarchy:

Knowledge within these systems is organized as a hierarchy of malfunction hypotheses, representing different levels of process abstraction (Ramesh, et. al., 1992). Each level of hierarchy represents an increasing level of process detail. Under the establish-refine strategy, a hypothesis under consideration is evaluated by examining the knowledge-groups associated with it. The search mechanism consists of exploring the hierarchy in a top-down fashion and a malfunction hypotheses is completely validated.

Analytical Model-based Approaches:

The analytical model-based approaches require knowledge about the process in terms of either input-output models or first principles quantitative models based on mass and energy balance equations. Here, the following techniques are used.

Observer-based Fault Detection and Isolation:

The main focus in observer-based fault detection and isolation is the generation of a set of residuals which detect and uniquely identify different faults. These residuals should be robust in that the decisions should not be corrupted in the face of unknown inputs. Unknown inputs here include unstructured uncertainties such as process and measurement noise and modeling uncertainties. The aim of observers is to come up with an error innovation sequence like

$$e(k+1) = Fe(k) - TKf(k) \quad (1)$$

where F is the observer system matrix and T is the input transformation matrix, and K is the fault distribution matrix.

If no faults occur in the process, f(k) = 0 then

$$e(k+1) = Fe(k) \quad (2)$$

If the absolute value of the eigenvalues of F are less than 1, then e(k) -> 0 as k ->∞.

In the absence of any faults, this observer will track the process independent of the unknown inputs d(k) to the process. Hence these are known as an unknown input observers. The necessary and sufficiency condition for the existence of these kinds of observers are described in Frank and Wunnenberg (1989).

Parity-Space Approach:

Parity equations are suitably arranged forms of the input-output model of the plant. The basic idea is to check the parity (consistency) of these input-output models of the plant by using the sensor outputs (measurements) and known process inputs. The idea of the approach is to rearrange the model structure so as to get the best isolation of the failures. Chow and Willsky (1984) proposed a procedure to generate parity equations from the state-space representation of a dynamic system. Gertler and Singer (1990) extended this to statistical isolability under noisy conditions by defining marginal sizes for fault alarms. All these methods are limited to failures that do not include gross process parameter drifts. However, they are an attractive alternative owing to their ability to determine, *a priori,* isolability of different faults. A general scheme for considering both direct and temporal redundancy in parity equation generation is provided by Chow and Wilsky (1986). In contrast, voting techniques are often used in systems that possess high degree of parallel hardware redundancy (Willsky, 1976).

Parameter Estimation:

Diagnosis of parameter drifts which are not measurable directly requires on-line parameter estimation methods. Accurate parametric models of the process are needed, usually in the continuous domain in the form of ordinary and partial differential

equations. Young (1981) and Isermann (1984) surveyed different parameter estimation techniques such as least squares, instrumental variables and estimation via discrete-time models.

Pattern Cassification Methods:
Pattern classification using process history data is usually performed using either statistical and non-statistical techniques. Bayes classifier is one of the more popular classifiers using the statistical properties of the input data. Neural networks have proved to be a popular non-statistical approach to pattern recognition.

Parametric and Non-parametric Classifiers:
Statistical pattern classification methods again can be roughly compartmentalized into two components: (i) a priori knowledge; assumption about the form of Probability Density Function (PDF) available, and (ii) search technique; the classifier design. Estimation of PDF (Fukunaga, 1972) can be classified as parametric and non-parametric estimation and the classifier can be also designed either in a parametric or non-parametric fashion.

Neural Networks as Classifiers:
A lot of interest has been shown in the recent literature in the application of neural networks for the task of pattern classification in fault diagnosis (Hoskins and Himmelblau, 1988; Kavuri and Venkatasubramanian, 1994). To understand neural networks better it helps to view them from a statistical pattern recognition perspective. Let us consider a standard two layer neural network. The first layer connecting the input to the hidden nodes tries to estimate the PDF for each class and the second layer connecting the hidden nodes and the output acts as a classifier. It is not surprising then that the network based classifiers are inferior to parametric classifiers when the density information for the class is available. When the assumption of a functional form for the density function could be made, parametric classifiers are a better choice. However, for a general classification problem, an *a priori* choice cannot be made for the functional form of the density. Moreover, the data available for the classes may not be enough to develop approximations to the density function. In such cases, non-parametric classifiers such as the network based classifiers are to be preferred.

As a general comparison of these different approaches, one can state that knowledge-based systems can be used where fundamental principles based approaches are more difficult or lacking, where there is an abundance of experience but not enough detail available to develop accurate quantitative models. However, they suffer from the resolution problems resulting from the ambiguity in qualitative reasoning. Parity space and observer-based approaches (analytical model-based methods) are shown to be equivalent in that they can be developed to generate the same residuals. Merits and demerits of one group carry to the other. These methods provide design schemes in which the effects of unknown disturbances can be minimized, isolability conditions ascertained, and sensitivity analysis performed in a consistent manner. The cost for obtaining these advantages are mainly modeling effort, computational effort, and the restrictions that one places on the class of acceptable models. Pattern classifiers are constructed solely based on process history data. The main advantages of classifiers are: their real-time performance, ease of knowledge acquisition, and applicability to a wide variety of systems. There are some limitations to methods which are based solely on process history data. It is the limitation of their generalization capability outside of the training data. This problem is alleviated by radial and ellipsoidal units by avoiding a decision in case there are no similar training patterns in that region. This allows the network to detect unfamiliar situations arising from novel faults. Besides its lack of ability to generalize to unfamiliar regions of measurement space, classifiers based solely on process history data also have difficulties in identifying multiple faults.

The review of the fault diagnosis approaches presented here does not adequately cover the considerable body of work that is available in the literature. This review is necessarily brief due to spatial constraints. For a more thorough review the reader is referred to Venkatasubramanian et. al. [1994].

4.3. Data Reconciliation

Data reconciliation can be viewed as a quantitative fault diagnosis problem with the focus on detecting sensor faults and sensor biases. Another important goal is to remove the measurement noise from process data to enhance the control performance. Data reconciliation usually consists of three parts: (i) identification of the biased parameter, (ii) estimation of the bias, and (iii) rectification of the sensor measurements.

Romagnoli and Stephanopoulos (1981) proposed a systematic method for identifying the source and location of gross errors in linear systems under steady-state conditions. There are three levels to their proposed strategy. (i) A structured search of the balance equations for measurements with gross errors. (ii) Sequential search of the balance equations that reduces the search further. (iii) Another level of sequential search that identifies the gross error. Mah

and Tamhane (1982) proposed a statistical test on the residuals to identify gross error.

Crowe, et. al. (1983) proposed a matrix projection method for data reconciliation problems for the linear case. Valko and Vajda (1987) introduced the idea of decoupling the parameter estimation problem from the state variable estimation problem. Though the original problem is not naturally decomposable in this manner, the rationale for doing this is that one always has good initial values for the state variables, whereas, it is hard to provide good initial values for the parameters. Recently, Liebman et. al. (1992) proposed a nonlinear programming methodology for data reconciliation in nonlinear processes under transient conditions.

Most practical processes are nonlinear in nature and hence linear reconciliation methods might not be adequate for practical problems. Also, steady-state reconciliation methods might prove ineffective in handling transients. In this context, the nonlinear programming approach proposed by Liebman, et al. (1992) is quite promising. The issues that have to be addressed in this approach are, computational complexity in the case of large-scale nonlinear problems and nonconvexity problems.

4.4. Supervisory Control

Supervisory control, typically, has a variety of functions. It includes model updating, controller tuning, reacting to equipment failures, "gain scheduling" to reflect changes in the disturbance variables, changes in the process, and so on. It might also include, for example, in batch plant automation, dealing with sequential controls and exception handling. It could also potentially include automatically making major changes in controller configurations or control algorithms to reflect process changes, online optimization, automated startups and shutdowns of continuous plants and scheduling. These are knowledge intensive tasks and knowledge-based methods have been proposed previously in the literature.

Kraus and Myron [1984] presented a self-tuning controller that uses pattern recognition techniques. Automatic controllers are tuned manually in usual practice. The control engineer perturbs the closed loop, observes the pattern of response, and compares this response to the desired pattern. Then, using his experience, he adjusts the control parameters. The pattern recognition based self-tuning PID algorithm monitors the closed-loop recovery following a set point or load disturbance and automatically calculates the P, I and D so as to minimize the process recovery time, subject to user-specifiable damping and overshoot constraints. Cooper and Lalonde (1990) have presented the idea of detecting naturally-occurring input excitation events based on the recent history of manipulated process input and also the calculation of model gains to develop a continuous gain schedule function for better control of nonlinear systems. The utility of knowledge-based expert systems in performing diagnostics and tuning control systems in real-time has been discussed by Arzen (1991). The formal integration of pattern recognition techniques in control systems to design "Intelligent Controllers" has been proposed by Stephanopoulos (1991).

5. BRINGING IT ALL TOGETHER: FUTURE DIRECTIONS IN INTEGRATED PROCESS SUPERVISION

As we reviewed in the last section, there has been considerable progress made in the last decade in developing efficient solution strategies for the various operational tasks. In this section, a perspective on how these paradigms, tools, and techniques in process operations might evolve and come together in the near future is presented. To this effect, first, one possible integration framework is discussed. The intent here is to discuss the nature and the extent of interaction that might occur between the various tasks in such a framework. Then, some perspectives on how the conceptualization and solution techniques for these various tasks themselves might develop is provided.

5.1. A Framework for Integrated Process Supervision:

One can approach the formulation of an integration framework from different viewpoints such as: (i) information flow, (ii) flow and management of data (iii) functional blocks view, and (iv) knowledge management. From the perspective of operational tasks, the most important facet of the integration framework is the functional blocks view and one such interpretation of the integration framework is provided in Fig. 2. The figure shows the structure and the interfacing of various process operational tasks and the information-flow dependence between modules that perform these tasks. The main functions of the monitoring system are to provide concise executive summaries to be presented to the operator, extract and abstract hierarchical trend explanation to be passed on to a diagnostic system and, detect and remove outliers. The fault diagnostic system houses different kinds of knowledge like rules, temporal patterns, causal models and pattern information. The diagnostic system assists the operator in identifying the root cause of the problems and also passes on this information to both supervisory control and data reconciliation modules. The data reconciliation module estimates the values of the parameters to be sent to the supervisory control module and also provides the regulatory control with the reconciled process data. Supervisory control module would have the complete information about the state of the process. The supervisory system would utilize the trend information and the diagnostic information to

suggest any changes needed at the regulatory control level.

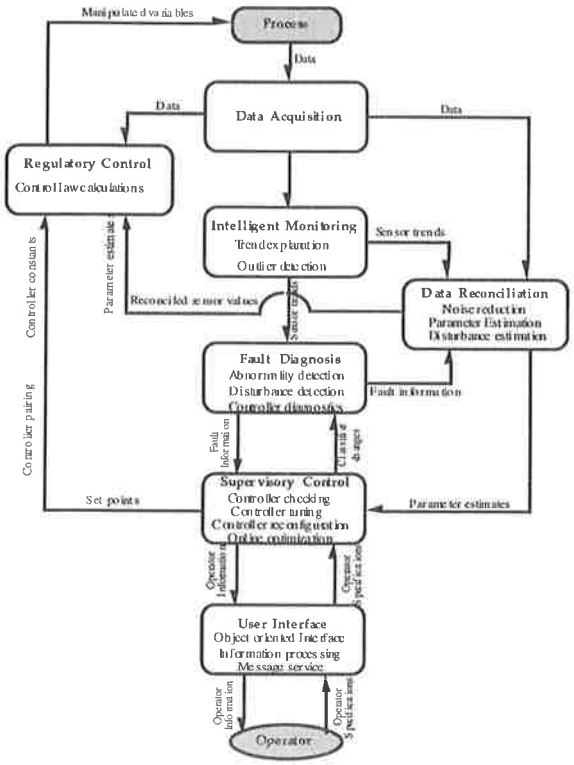


Fig. 2. A framework for Integrated Process Supervision

5.2. Process Monitoring

The important recent developments in the field of process monitoring have been the advances made in statistical process monitoring and syntactic pattern recognition, as noted earlier. The ultimate use of process monitoring is in diagnosis. The integration of these monitoring techniques with diagnosis is still not fully developed. One needs to see improvements in this regard in the future. Also, the recognition of operators as an important part of the operational decision-making has put an onus on the monitoring systems. The monitoring systems should be able to provide information to the operators in a way that they can understand them. These systems should also be able to interact with the operators, take suggestions from them and interpret these suggestions for the other operational tasks. Hence, one needs integration of Natural Language Processors (NLP) into typical monitoring systems in the future.

5.3. Fault Diagnosis

One of the important underlying points in all fault diagnosis methods is the inadequacy of a single method to handle all the requirements for a diagnostic classifier. Though all the methods are restricted, in the sense that they are only as good as the quality of information provided, still some methods might be better suited for a particular problem at hand than others. Hence, hybrid methods where different methods work in conjunction to perform collective problem-solving are an attractive alternative. For example, fault explanation through a causal chain is best done through the use of digraphs, whereas, fault isolation might be very difficult using digraphs due to the inherent ambiguities in qualitative models. Analytical model based methods are superior in this regard. Hence, hybrid methods might provide a general powerful problem-solving approach. Consider another example, where a pattern-based classifier and a trend-based classifier are used sequentially for improved search. The pattern based classifier localizes the search based purely on the spatial organization of various fault patterns. Once this is done, a trend based classifier can take the set of fault hypotheses to see if they can be further distinguished based on the temporal pattern they exhibit. One can hope to improve the resolution characteristics of an overall diagnostic framework by combining various approaches like these.

There has been some work on hybrid architectures. The two-tier approach by Venkatasubramanian and Rich (1988) using compiled and model based knowledge is an example of a hybrid approach. Frank (1990) advocates the use of knowledge-based methods to complement the existing analytical and algorithmical methods of fault detection. The combination of methods allows one to evaluate different kinds of knowledge in one single framework for better decision making. The resulting overall fault detection scheme would have a knowledge base consisting of both heuristic knowledge and analytical models, data base, inference engine and explanation component. These methods provide promising prospects for the solution of general diagnostic problems.

Dynamic simulation in diagnosis:

When one has access to dynamic models of a process, one should take advantage of such models in real-time diagnosis. However, this is not usually done due to the complexity of the models and the difficulties involved in integrating diverse approaches in a single framework. But using a framework such as the one in Figure 2, a hybrid system would be feasible.

The basic idea here is to use a signed digraph for doing fault diagnosis at a first level. The completeness for a diagnostic system using digraphs is usually quite good, whereas, the resolution might be poor. The digraph will come up with a malfunction hypotheses set which would also include the actual fault(s). A prioritizer will then order the faults for further validation. Under a single fault assumption, the faults can be simulated using on-line first principles model. The simulated data and actual data can be compared using trend modeling approaches for validation. Through a hybrid approach like this,

one can hope for improved completeness and resolution in a diagnostic module.

5.4. Data Reconciliation

As mentioned before, most attempts in the past had restricted themselves to linear and/or small systems. With the recent progress in optimization and the emphasis on plant-wide control, people are attempting large-scale nonlinear optimization problems. From a purely computational point of view, if the only bottleneck is the "largeness" of these problems, it may be handled by faster computers,parallel computing and more efficient algorithms. However, there are other issues like local minima problems inherent in many nonlinear process situations which makes the solution to large-scale nonlinear optimization methods very difficult, particularly in real-time. Hence, one needs to think of new ways of formulating the problem and new solution strategies drawing from different fields that might mitigate this complexity. For example, instead of using purely gradient-based approaches, one can think of a combination of qualitative and quantitative approach. Diagnostic qualitative knowledge can be used to reduce the search space and provide good initial guesses thereby enhancing the performance of the data reconciliation module.

5.5. Supervisory control

While there has been a lot of work in regulatory control over the years, much less attention has been devoted to supervisory control issues (Garcia et. al., 1991). As noted earlier, supervisory control includes model updating, controller tuning, reacting to equipment failures, changes in controller configurations or control algorithms to reflect process changes, online optimization, automated startups and shutdowns and so on. The following outlines some of the important issues to be addressed in the context of controller performance.

Identification of out-of-tune controllers:

The simplest kind of supervisory control action one can think of is the monitoring of individual control loops. In doing this a test signal is sent periodically to perturb the closed loop system to a small degree. By using pattern matching techniques on the resultant output of the perturbed system, one can identify out-of-tune controllers or controllers with degraded performance. Once the problematic controllers are identified, they can be tuned using rules in a knowledge-based system or other techniques.

Designing controllers for various inputs:

Generally, there is no single perfect controller for all kinds of input disturbances. A controller designed for a step input in a particular variable might not give acceptable performance if there is a ramp input in that variable. Furthermore a control system cannot be designed to work well for disturbances in all the input variables. These issues might become crucial if the system is being operated under tight quality control requirements. This is another area where one might see future developments in supervisory control systems. A supervisory control system can adaptively toggle between various control laws based on the specification about the state of the plant. This specification about the state of the plant can be provided by a combination of sensor trends, fault hypotheses from the diagnostic system and the estimates from the data reconciliation module.

Update the model to redesign controller:

Other than the input disturbances, there can be some structural changes in the plant model itself. In such a situation the different kinds of controllers previously designed might no longer be valid. This could call for redesigning the controllers. This is another place where one might use the information from the diagnostic system to update the model. Once the model is updated, one can decide about the controllers that are affected by this change in the model. With this new information from the model, a redesign of the controllers can be performed.

Variable pairing selection:

The variable pairing for the controller depends to a great extent on the state of the system. Once there is a change in the state of the system, the original pairing might no longer be optimal. There might exist new pairings corresponding to the state of the system that might provide optimal control action. By using methods like Relative Gain Arrays (RGA) and Singular Value Decomposition (SVD) techniques in conjunction with knowledge-based systems, a supervisory control module can advise the operator of the different options available and suggest an optimal configuration.

Detuning and reconfiguration of controllers:

In case of process upsets one might want to continue production in the system with minimal impact. Having minimal impact on the system might call for reconfiguration of the controllers in the system. This can be done effectively if one has built-in redundancy in the control system in the design stage itself. To this end one might need to detune some controllers and bring into operation other controllers. Having bypass lines and rerouting streams might be another way of moving variability in the process to different locations. Detuning of controller might also be done in the case of unanticipated instability in the system.

User specified, online, interactive design of controllers:

Another kind of activity that the supervisory control module can do is the online interactive design of control parameters. In a typical plant, operating strategies change form time to time. Such changes might necessitate the redesign of control parameters. In some situations, for example, one might want a small settling time without worrying too much about the overshoot in the response. In contrast, in some other situation, one might want as small a

overshoot as possible without any constraint on the settling time. These factors are altered by using the control parameters. In such situations the supervisory control module can have an interactive session with the operator to identify optimal controller parameters.

6. CONCLUSIONS

The main purpose of this paper is to present some perspectives on the current status of integrated process supervision and identify future directions. The focus of the paper is primarily on the integration of low-level and middle-level tasks such as regulatory control, process monitoring, diagnosis, data reconciliation and supervisory control. While these operational tasks may be intrinsically different from each other, they are, however, closely related and can not be treated in isolation. Hence, there exists a strong and clear need for an integrated framework so that the operational decision-making can be made more comprehensively and more effectively. In such a unified framework, it would be relatively easier to see the effect of various choices in some particular task at hand on the different aspects of the overall operational picture.

While the advantages of integrated process supervision are quite clear, achieving it is no simple task as there are many challenges. In this paper, we reviewed these challenges and identified the underlying issues which need to be addressed. In this context, we discuss the key role being played by artificial intelligence methodologies. After considerable experimentation and introspection over the past decade, AI applications in process operations are begining to show considerable promise. AI has provided a means for developing an integrated platform that allows for reasoning with different forms and levels of representation about different tasks. Most importantly, it has helped focus research in academia on issues that are of practical importance but appeared too distant for conventional methodologies. This paper also reviewed the current status of automating the various operational tasks and identified some future directions to pursue towards integration. While we do have a long, challenging road ahead of us towards total integration, and we do need to have much more accomplished to achieve a satisfactory integrated solution to process supervision, the journey so far has been quite encouraging.

7. REFERENCES

Årzén, K. E. (1991). Knowledge-Based Applications in the Process Industry: Current State and Future Directions. IFAC Workshop on Computer Software Structures Integrating AI/KBS Systems in Process Control, Bergen, Norway.

Bakshi, B. R., and G. Stephanopoulos (1993). Wave-net: a Multiresolution, Hierarchical Neural Network with Localized Learning. *AICHE Journal,* **39**, No.1, 57-81.

Charniak, E., and D. McDermott (1984). *Introduction to Artificial Intelligence*. Addison-Wesley Publishing Company, Massachusetts.

Cheung, J. T., and G. Stephanopoulos (1989). Representation of Process Trends, Part I. A Formal Representation Framework. *Comput and Chem Engng* **14**, No. 4, 495-510.

Chow, E. D., and A. S. Willsky (1984). Analytical redundancy and the design of robust failure detection systems. *IEEE Trans. Automat. Contr.*, **AC-29**, 7, 603-614.

Cooper, D. J., and A. M. Lalonde (1990). Process behavior diagnostics and adaptive process control. *Computers. chem. Engng.*, **14**, No. 4/5, 541-549.

Crowe, C. M. (1986). Reconciliation of Process Flow Rates by Matrix Projection Part II: The Nonlinear Case. *AICHE Journal.*, **32**, No. 4, 616-623.

Frank, P. M., and J. Wunnenberg (1989). Robust fault diagnosis using unknown input observer schemes. In: *Fault Diagnosis in Dynamic Systems: Theory and Application*, (Patton, R. J., P. M. Frank, and R. N. Clark, Eds.) Prentice Hall, New York.

Frank, P. M. (1990). Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - A survey and some new results. *Automatica*, 26(3), 459-474.

Fukunaga, K (1972). *Introduction to statistical pattern recognition.* Academic Press, New York.

Garcia, C. E., B. L. Ramaker, and J. F. Pollard (1991). Total process control - Beyond the design of model predictive controllers. *Proceedings of the Fourth International Conference on Chemical Process Control*, February.

Gertler, J., and D. Singer (1990). A new structural framework for parity equation-based failure detection and isolation. *Automatica*, **26**, 381-388.

Hoskins, J. C., and D. H. Himmelbalu (1988). Artificial neural network models of knowledge representation in chemical engineering. *Computers chem. Engng.*, **12**, 881-890.

Iri, M., K, Aoki, E. O'Shima, and H. Matsuyama (1979). An algorithm for diagnosis of system failures in the chemical process. *Comput. & Chem. Engng.*, **3**, 489-493, 1979.

Isermann, R. (1984). Fault detection based on modelling and estimation methods: A survey. *Automatica*, **20**, 387-404.

Janusz, M., and V. Venkatasubramanian (1991). Automatic Generation of Qualitative Description of

Process Trends for Fault Detection and Diagnosis.*Engng Applic. Artif. Intell,* **4**. No.5, 329-339.

Kavuri, S. N., and V. Venkatasubramanian (1992). Combining pattern classification and assumption based approaches for process fault diagnosis. *Comput. Chem. Engng* ., **16**, No. 4, 299-312.

Kavuri, S. N., and V. Venkatasubramanian (1994). Neural network decomposition strategies for large-scale fault diagnosis, *Int. J. Control*, **59**, No. 3, 767-792.

Kramer, M. A., and B. L. Palowitch (1989). A rule-based approach to fault diagnosis using the signed digraph. *AICHE J.*, **33**, No. 7, 1067-1078.

Kraus, T. W., and T. J. Myron (1984). Self-tuning PID controller uses pattern recognition approach. *Control Engineering*, June.

Lapp, S. A., and G. A. Powers (1977). Computer-aided synthesis of fault trees. *IEEE Trans. Reliability*, **R-37**, 2-13.

Liebman, M, J., T. F. Edgar, and L. S. Lasdon (1992). Efficient Data Reconciliation and Estimation for Dynamic Processes Using Nonlinear Programming Techniques *Computers. Chem.Engng*, **16**, No 10, 963-986.

Mah, R. S. H., and A. C. Tamhane (1982). Detection of gross errors in process data. *AICHE J.*, **28**, 828.

Marr, D., and E. Hildreth (1980). Theroy of edge detection. *Proc. Royal. Soc. Lond.*, **B 207**, 187-217.

Reiter, R (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, **32**, 57-95.

Rengaswamy, R., and V. Venkatasubramanian (1992). An Integrated Framework for Process Monitoring, Diagnosis, and Control Using Knowledge-Based Systems and Neural Networks. IFAC Symposium on *Online Fault Detection and Supervision in the Chemical Process Industries* , Newark, DE, USA.

Rich, S. H., and V. Venkatasubramanian (1987). Model-Based Reasoning in Diagnostic Expert Systems for Chemical Process Plants. *Comput.chem.Engng*, **11**, 111-122.

Rich, S. H., and V. Venkatasubramanian (1989). Causality-based failure-driven learning in diagnostic expert systems. *AICHE J.*, **35**, No. 6, 943-950.

Romagnoli, J. A. and G. Stephanopoulos (1981). Rectification of process measurements data in the presence of gross errors. *Chem. Eng. Sci*, **36**, 1849-1863.

Shiozaki, J., H. Matsuyama, E. O'Shima, and M. Iri (1985). An improved algorithm for diagnosis of system failures in the chemical process. *Comput. Chem. Engng.*, **9**, 285.

Stephanopoulos, G (1991). Towards the intelligent controller: Formal integration of pattern recognition with control theory. *Proceedings of the Fourth International Conference on Chemical Process Control*, February, 1991.

Ulerich, N. H., and G. A. Powers (1988). Online hazard aversion and fault diagnosis in chemical processes: The digraph + fault tree method. *IEEE Trans. Reliability*, **37**, 2, 171-177.

Valko, P., and S. Vajda (1987). An Extended Marquardt-type procedure for fitting Error-in-Variables Models. *Computers.Chem.Engng*, **11**, No. 1, 37.

Venkatasubramanian, V. and S. H. Rich [1988]. An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis. *Comput. Chem. Engng.*, **12**, 903-921.

Venkatasubramanian, V., S. N. Kavuri, and R. Rengaswamy (1994). Process fault diagnosis - A Review. Submitted to *AIChE J.*

Willsky, A. S (1976). A survey of design methods for failure detection in dynamic systems. *Automatica*, **12**, 601.

Young, P (1981). Parameter estimation for continuous-time models - a survey. *Automatica*, **17**, 1, 23-39.

A MODEL FOR INTEGRATION OF KNOWLEDGE-BASED COMPONENTS IN EXISTING PROCESS CONTROL SYSTEMS

V. BACVANSKI

Aachen University of Technology, Lehrstuhl für Informatik III, Ahornstr. 55, D - 52074 Aachen, Germany.

Abstract: Integration of knowledge-based and conventional systems is a particularly hard problem because of the complete incompatibility of data structures of this two kinds of systems. The weak points of knowledge-based systems such as poor performance, enormous resource requirements and incomprehensibility hinder a wider acceptance of knowledge-based approaches in process control. The new approach for the integration of knowledge-based and conventional systems provides a unified data model for both worlds, offers a structuring mechanisms for the rule and the fact base and improves the performance. The main contribution of this work is a new statically and strongly typed knowledge-based language fully compatible with C++, which constructs provide for type-safe and efficient integration with conventional components. A particularly important point is that no modification of the conventional code and no conversions of the data are needed, providing for exchangeability of paradigms in process control applications.

Keywords: Knowledge engineering; expert systems; software engineering; programming languages; process control

1. INTRODUCTION

Knowledge-based (KB) and rule-based systems are the most successful areas of artificial intelligence. Their popularity comes from the fact that besides algorithmic knowledge, many problems require the use of heuristics which cannot be efficiently expressed by algorithms. Many heuristics are represented in a form of rules but the conventional programming languages cannot represent and execute the rules directly. Rule-based systems are considered to be the most commonly used of the various types of KB systems, because rules are a natural and very common format used by experts to express problem-solving knowledge in many types of domains (Gonzales and Goforth, 1993). However, in spite of many successful applications, failures of KB systems in industrial applications are not uncommon.

The paper is organized as follows: in Chapter 2 problems with rule-based representations and integration of KB and conventional systems are discussed. The main ideas of the new approach are introduced in Chapter 3. Abstract data type operations and event generation are associated by mechanisms described in Chapter 4. In Chapter 5 handling of generated events, integration on the architectural level and structuring of the rule space are presented. Embedding in the conventional application is presented in Chapter 6. At the end, a summary of what we have accomplished is given.

2. PROBLEMS OF KB SYSTEMS IN PROCESS CONTROL

Problems of KB systems and rule-based knowledge representation will be explained on an example which describes a system for control of groups of pumps in a process control system. The architecture of the system implemented in a conventional programming language is depicted on Fig. 1. It consists of a main module, which uses the PumpControl module for the control of pumps. Information about characteristics of the transported fluids are provided by the module Fluid. Other parts of the system are not of importance for the example.

Problem domain knowledge about pump control is given mainly in a form of rules, while containing some algorithmic parts. The PumpControl module, while implemented in a conventional programming language, exhibits two severe drawbacks.

The system is *incomprehensible* and the knowledge in the system is *not maintainable*: the realization of the rule system is scattered in a large number of *if*-statements of the conventional language; there is no obvious correspondence between rules in the problem domain and rules in the implementation.

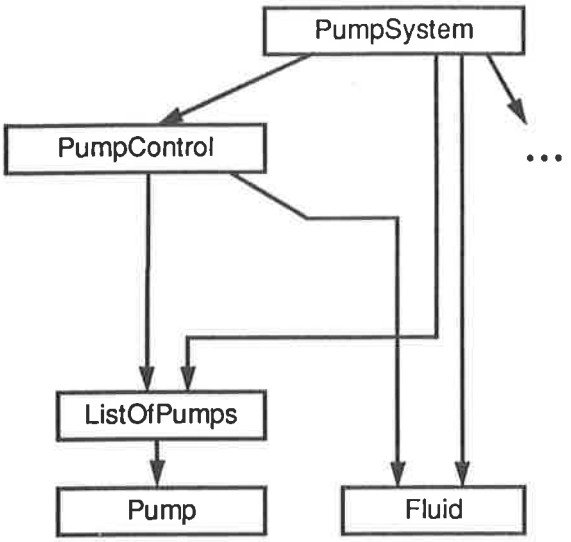


Fig. 1. Conventional Pump System

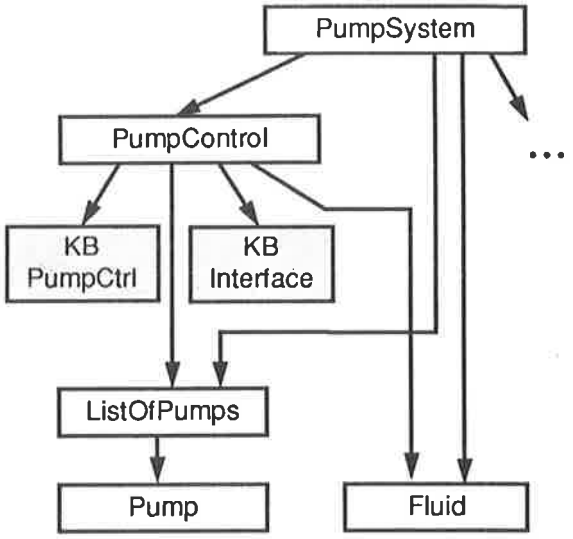


Fig. 2. Pump System - The ES Implementation

The second drawback is that the system is *not reusable*: the inferencing mechanism of the rule system is intertwined with the implementation of rules; the implemented inferencing mechanism cannot be separated and used in a similar application.

However, the implementation of a rule-based system in conventional languages has also an advantage: the *integrability* with other components and *performance* are very good since the system uses the conventional language with its efficient data structures. Additionally, the problem can be divided into several units using the modularization capabilities of modern programming languages.

Trying to represent the knowledge more directly, the developers are now trying to make use of a KB component for the PumpControl. However, the incorporation of the KB system in the application is not easy: an additional nontrivial software component, KBInterface, must be developed for the integration of the KB component and the conventional part. Its main function is to perform translation of data between the KB and conventional components. The resulting architecture is presented in Fig. 2.

This solution exhibits a large number of problems. The inability of KB systems to be integrated into already existing data processing environments is here, as well as is in many other cases, the most important drawback. Papers on development of KB systems report that the integration with conventional systems was the most difficult and time consuming part (Irgon *et al.*, 1990), and in some cases was not achievable at all (Lazzara and Marcinkovski, 1986).

Many of the modern KB systems are able to run on conventional hardware platforms, but their use of completely different languages makes system integration very hard. On the lowest level, their data structures are not compatible, what makes expensive translations necessary in order to achieve the bare communication between two parts of a system. It is often the case, especially on less powerful hardware, that the integration can be achieved only via external media - e.g. local area network or file system (Wong *et al.*, 1988). This problem becomes particularly painful when the system has to process large volumes of data and when complex data structures have to be maintained simultaneously in both systems (Bacvanski, 1992).

Newer generation KB tools targeted to general purpose hardware are often implemented in conventional procedural languages, but it is frequently the case that they simulate mechanisms and data structures of Lisp-like languages. That has as consequence that those as "fully integrable" advertised systems have only elementary data types compatible with conventional languages. There have been attempts to overcome few of the integration deficiencies using various approaches, as described by (Attali and Franchi-Zanettacci, 1987; Butler *et al.*, 1988; van Biema *et al.*, 1990; Collard, 1988; IntelliCorp, 1990; Madhav, 1990; Nuutila *et al.*, 1987). Nevertheless, these attempts introduce improvements to some of the problems, but they do not offer support for the integration on a high level, nor they present a methodology for development of heterogeneous, KB and conventional software components.

Further, KB systems do not have facilities for structuring the system into modules. This results in large monolithic systems which are hard to maintain when the number of rules becomes large. That holds also for the structuring of the fact base - there is no encap-

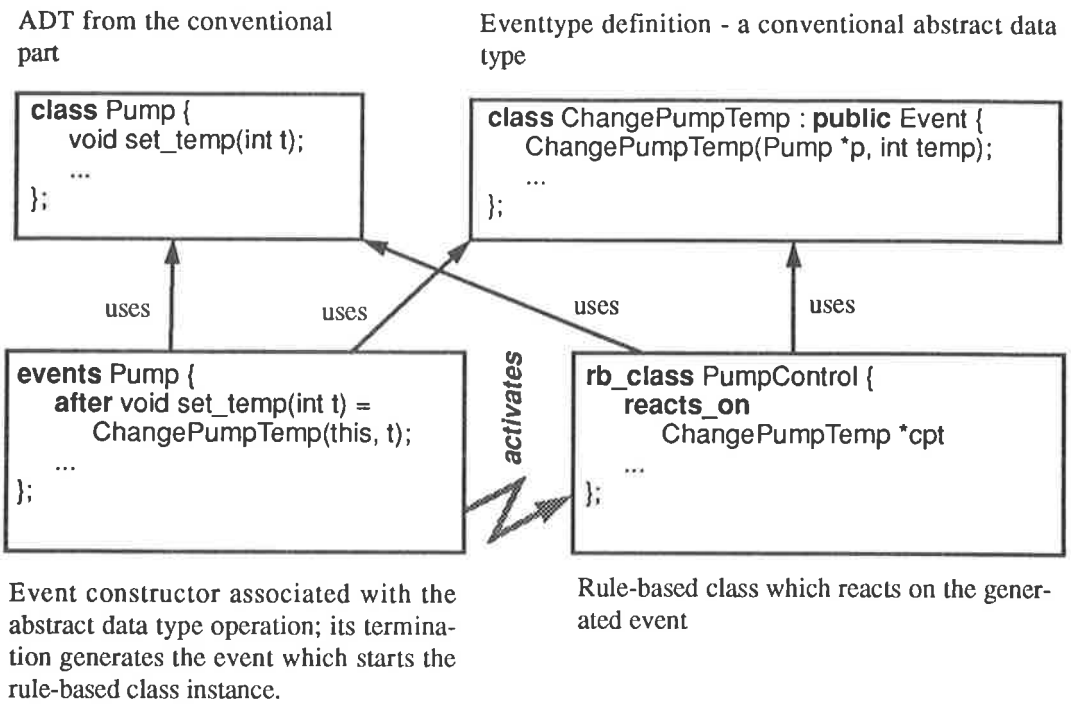


Fig. 3. Integration of Events with Abstract Data Types

sulation mechanism. This means that data in KB systems can be freely modified jeopardizing the semantic integrity which can be enforced by allowing access only to operations defined in the interface of the abstract data type.

3. A NEW MODEL FOR INTEGRATION

From the analysis of the drawbacks of the integration capabilities of existing KB systems and conventional components it can be concluded that a new approach should be developed. The goal is to provide a system which combines good characteristics of conventional and KB solutions. This results in a *new KB language* which is *fully integrable* with a conventional language. In addition, this new language should have *structuring facilities* and constructs which provide a *high-level interface* to the conventional parts of the application, and this interface must be *strongly typed.* A *unified data model* is to be provided in order to circumvent costly translations from one to another data format and to prevent inconsistencies which occur when the KB and the conventional system process replicated data with the same semantics.

In forward chaining KB systems, inference engine is activated by changes of data attributes. Since our approach respects the encapsulation of data, we cannot allow usage of this mechanism. Data in abstract data types (ADT) can be accessed only by operations defined in the interface of the type. Therefore, in our approach, the invocation or the termination of the ADT operation is a significant point during program execution which can start the inference mechanism. The operations of ADTs are associated with creation of events which are handled by special rule units. Data structures of the conventional system thus become the fact base of the KB system.

A language that provides for multiparadigm development of systems conmbinig procedural, object-oriented, and rule-based paradigms is developed as a *rule-based extension of C++*. New concepts and language constructs are defined in a separate language layer which do not cause interference with already existing language constructs. This provides for a full reusability of code written without using the extensions in a standard C++ context.

4. THE EVENT INTERFACE

In order to associate the generation of events with invocation or termination of the ADT operations, a special language construct is introduced. It respects encapsulation and, what is particularly important for real-world applications, *leaves the definitions of ADTs and the data intact*. Further, it enables generation of any event types, as defined by the developer. Arbitrary type definitions can serve as event types, but event types typically contain a reference to the object that changed its state. Additionally, other informations can be contained in the event. Later, the rules in the rule units can make use of this additional information, thus reducing unnecessary pattern

matching and therefore improving the overall performance of the system.

In order to associate an operation of an ADT with an event type, the developer defines first a type that will serve as an event. After that, in the events definition the creation of the event is bound to the invocation or the termination of an ADT operation (Fig. 3). Leaving the conventional language and data intact allows for full reusability of existing conventional components and data.

The event binding definitions are compiled by an *event compiler*, and implementations of the ADT operations are instrumented with statements which create the instances of events and invoke the EventManager of the appropriate event class to perform multicasting of the event instance to the interested parties. The EventManager classes are generated by the compiler and are fully transparent to the developer.

In the event constructor, all data members of the ADT can be used, as well as parameters of the operation.

5. THE RULE-BASED CLASS INTERFACE

The main functions of the rb_class construct are to provide a *structuring* and *encapsulation mechanism* for rules and to facilitate *integration* with other rb_classes and conventional components. Additionally, it contains a pragma for the compiler to direct which inference machine to select for the particular rb_class. The rb_class construct was developed by having in mind structuring concepts of the *programming in the large* methodology.

The most important construct here is the reacts_on clause which defines on which types of incoming events the rb_class reacts. Besides, the from part limits the objects from which the events should come. Since the usage of arbitrary collections is allowed, the name and signature of the function for the membership test is not known by the system. Therefore, after the condition reserved word, the expression which forms the membership test is written. The test can also serve for testing of other, special conditions which control the work of the instance of the rb_class. Fig. 4 illustrates the rb_class for the PumpControl component. Besides from particular objects, a rule-based class can declare that it is interested in events regardless of their source, what is done by the keyword combination from anywhere.

```
rb_class PumpControl {
    reacts_on ChangePumpTemp *cpt
        from List<Pump*> * monitored_pumps
        condition monitored_pumps->
            member(cpt->pump());
    reacts_on ChangeFluidMaxTemp *cfmt
        from anywhere;

    inference_engine ForwardChainer;

    rule PumpTooHot {
        reacts_on ChangePumpTemp;
    }
    ...
};
```

Fig. 4. Rule-Based Class PumpControl

This system supports forward chaining, but the concrete variants of inference engines can be selected from the inference engines in the library. After the declaration of the inference engine, declaration of rules follow. The declarations of rules contain the name of the rule and specifications on which events a rule reacts. Full implementation of the rules iscontained in the implementation part.

Another important novelty in our approach is that the rules can directly access events that caused their execution. Since the events typically contain a reference to the object that changed its value, there is no need to perform pattern matching to find the object. However, pattern matching cannot be eliminated in all situations. In contrary to other approaches, where a special kind of collection must be used for data used in pattern matching, in this model pattern matching is available on arbitrary collections. The matches_on construct in rules can define the way how to iterate over the collection (including plain arrays from the conventional language). A rule from the PumpControl rb_class is given in the Fig. 4.

```
rule PumpMonitor :: PumpTooHot {
    // direct access to the event:
    reacts_on ChangePumpTemp *cpt;
        // local variables for the rule:
        { Pump * p = cpt->pump(); }
    if (p->temp() > p->fluid()->max_temp())
    then {
        operator << "Pump " << p->name()
            << "too hot, slowing down 25%";
        p->set_power(p->power() * 0.75);
    }
};
```

Fig. 5. Rule from the **rb_class** PumpControl

6. EMBEDDING IN CONVENTIONAL APPLICATIONS

The rb_classes are compiled by a compiler which translates them to C++. An interesting characteristics of our approach is that the compiler uses building blocks on a high level of abstraction. It actually generates a whole software architecture for the KB system, while using components from the reusable and extensible library of KB components implemented as C++ classes. Generated classes for rule-based classes and rules inherit from the abstract components given in the KB library. The construction of the building blocks and the organization of the developed software architectures for integrable KB systems are out of scope of this paper.

Translated units have a set of operations which provide for integration with conventional part and which control the inference engine. After creation of the rule-based class instance, the operation set_external_objects connects the data from the conventional part with the reference names in the rule-based class (defined by the from construct). After the initialization, the operation start can be invoked after which the rule-based class instance becomes active and reacts on incoming events.

Since rule-based classes are translated to C++ classes, multiple instantiation is possible. This offers the possibility to partition the observed system and to associate separate rule-based classes for monitoring of different objects or collections. That also improves the performance since the number of objects that must be considered in pattern matching is reduced. The integration of the PumpControl in the system is given in the Fig. 4.

```
// colections of pumps which we will control
List<Pump*> * pump_group1;

// creation of the KB instance
PumpControl pump_ctrl();
...
// pumps are now the "fact base" :
pump_ctrl.set_external_objects(pump_group1);

// instance of the KB system is set to the state where
// it can react on incoming events generated by
// the operations of other classes
pump_ctrl.start();
```

Fig. 6. Integration of the PumpControl Instance

7. CONCLUSIONS AND FUTURE WORK

The new model for integration of KB systems in conventional environments strengths the potential to apply KB systems in process control applications. The most important contributions of this approach are the *strongly typed KB language* which is *fully compatible with C++* and provides a *unified data model* for conventional and KB systems. *No conversions* between data are necessary. Additionally, modern software engineering principles are respected, and new concepts for the integration of conventional and KB systems defined. A particularly important point is that the conventional components do not need to be modified in order to make use of the KB system, nor to be used by KB components. The translation to C++ provides for good performance and small memory requirements.

Another aspect of the project is a construction of a CASE tool which supports building of heterogeneous software systems. This tool is going to be integrated with tools from a conventional software engineering context by using messages over a network. Besides aspects of interconnection and organization of KB systems, applicability of various match algorithms and their coexistence with information hiding mechanisms of ADTs are still to be examined.

Further, it would be particularly interesting to extend the concepts presented in this work with language constructs for real-time programming. This will provide an opportunity to develop high-performance real-time KB systems.

The presented approach can be extended to distributed systems as well. Events in the system do not need to be limited to notifications of rule-based classes in the same program. The EventManager can as well create events that are distributed over a network. Such services can be realized without much effort by using distributed communication facilities, such as ToolTalk (SunSoft, 1991) or CORBA (Object Management Group, 1991).

8. REFERENCES

Abbott, R.J. (1987). Knowledge Abstraction. *Communications of the ACM*, Vol. 30 No. 8, August 1987.

Attali, I., P. Franchi-Zanettacci (1987). Inference System Environment for Ada. *Proc. of the Ada-Europe International Conference*, Stockholm, 26-28 May 1987.

Bacvanski, V. (1992). Software Engineering for Heterogeneous Knowledge-Based Systems. *Proceedings of the Third Annual Symposium of the IAKE*, Washington DC, November 16 - 19, 1992.

Butler, C.W., E.D. Hodil, G.L. Richardson (1988). Building Knowledge-Based Systems with Procedural Languages. *IEEE Expert*, Summer 1988.

van Biema, M., G.Q. Maguire, S. Stolfo (1990). The Constraint-Based Paradigm: Integrating Object-Oriented and Rule-Based Programming. *Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, Vol.II: Software*, January 2-5, 1990.

Collard, P., A. Goforth (1988). Knowledge Based Systems and Ada: An Overview of the Issues. *ACM Ada Letters*, Vol. 8 No. 6, November/December 1988.

Gonzales, A., D. Dankel (1993). *The Engineering of Knowledge-Based Systems*. Prentice-Hall.

IntelliCorp (1990). *ProKappa Reference Manual.*

Irgon, A., J. Zolnowski, K.J. Murray, M. Gersho (1990). Expert System Development: A Retrospective View of Five Systems. *IEEE Expert*, June 1990.

Jacob, R.J.K., J.N. Froscher (1986). Software Engineering for Rule Based Systems. *Proceedings of the 1986 Fall Joint Computer Conference*, IEEE Computer Society Press, Washington, 1986.

Lazzara, A.V., J. Marcinkovski (1986). A Framework for Extending Information Systems with Knowledge Based Processing Capabilities. *COMPSAC '86 Proceedings*, Chicago, Oct. 8-10 1986.

Madhav, N. (1990). *An Ada-Prolog System.* Technical Report No. CSL-TR-90-437. Stanford University, Computer Systems Laboratory, August 1990.

Nuutila, E., J. Kuusela et al. (1987).XC - A Language for Embedded Rule Based Systems. *ACM SIGPLAN Notices*, Vol. 22, No. 9, September 1987.

Object Management Group (1991). *The Common Object Request Broker: Architecture and Specification.* Draft, December 1991.

van de Riet, R.P. (1987). Problems with Expert Systems?. *Future Generation Computer Systems*, No. 3, 1987.

SunSoft (1991). *ToolTalk 1.0 Programmer's Guide.* Part No. 800-6093-11, Rev. A, December 1991.

Wong, F.S., W. Dong, M. Blanks (1988). Coupling of Symbolic and Numerical Computations on a Microcomputer. *Artificial Intelligence in Engineering*, Vol. 3, No. 1, 1988.

Software Integration of Real-Time Expert Systems

Robert K. Chun

Hughes Aircraft Company
Fullerton, CA 92634, USA

Abstract: A technique to achieve real-time computing throughput from rule-based expert systems is discussed. Using compilation, compaction, and parallelization techniques, this research synthesizes a set of concurrently executable Ada tasks from a knowledge base of rules. A prototype compilation system based on this approach has demonstrated speedups in excess of 100X along with increased embeddability of the knowledge base. The work is being extended into a software engineering development environment for expert systems whereby programming constructs from both the procedural and rule-based language domains are made available to the user. The approach strives to present the engineer with programming templates that protect him from the intricate control mechanisms of the inference engine and to enable him to concentrate on problem solving at higher levels of abstraction.

Key Words: Artificial Intelligence; Expert Systems; Heuristic Programming; Inference Processes; Knowledge Engineering; Parallel Processing; Software Engineering

1. Introduction

In recent years, rule-based expert systems have been proposed as a means for implementing complex decision-intensive processes for real-time embedded systems. Numerous characteristics of rule-based systems make them advantageous for the programming of knowledge-intensive systems; however, many real-time applications, such as those encountered in typical process control situations, will require inferencing throughputs exceeding those currently provided by commercial expert system shells. Also, data interfaces with other software modules is not as efficient as would be desired in an integrated system. Finally, the large memory requirements of a full-capability inference engine hampers its embeddability. These problems are expected to worsen as larger knowledge-based systems are developed to handle more sophisticated tasks. Therefore, although rule-based expert systems offer a unique programming paradigm and can be rapidly prototyped using commercial tools, some means of converting them into resource-efficient representations is needed. This paper describes ESCAPE (Expert System Compilation And Parallelization Environment), a prototype tool which has been built as part of a research project examining the application of software engineering and re-engineering principles to rule-based systems in order to achieve the performance levels required for real-time embedded systems.

2. Objectives

The objectives for ESCAPE were formulated by observing the techniques being used by various expert systems development teams working on actual projects. All of the knowledge bases examined were to be implemented in applications where, not only speed, but also embeddability and memory size were important issues. Although there were expected differences in the programming styles of the individual teams, it became apparent that there were some common patterns of development being applied.

In particular, because expert systems technology is relatively new, we observed that projects utilizing a knowledge-based approach usually took a conservative design path and first implemented a prototype. The main purpose of the prototype was to demonstrate the capability and functional correctness of the design specification and the expert system. Since these prototypes were developed using only a minimal amount of manpower and funds, they were usually implemented on a readily available commercial off-the-shelf (COTS) expert system development package. By using the built-in inference engine, user-friendly interface, and extensive debugging facilities provided by the COTS tool, the software development task was simplified considerably and the many advantages of using expert systems were cost-effectively realized.

Unfortunately, there are some serious deficiencies associated with this development scheme. One of the most serious problems is that of slow execution speed. Delays of a few seconds or even minutes to verify proper functional operation of a prototype might be acceptable. However, such processing delays are clearly unacceptable in a fielded version of the expert system operating in a real-time computing environment. The speed problem is caused mainly by the overhead associated with the slow, interpretive inference engine of the expert system. Two other problems arise during the conversion of the prototype into a deliverable version. First, very few COTS packages are directly embeddable with procedural software languages. Second, converting such a prototype into a deliverable is complicated because of the lack of a development methodology during the prototyping phase. In particular, exercising control over the rule firing order in the inference engine's agenda mechanism often proved difficult. The result is that the expert system developed on the COTS tool is useful only to the extent of being a conceptual prototype and must undergo a time-consuming manual re-engineering process before actual fielding on the target platform.

Ideally, one wants to preserve the convenience and cost-effectiveness of using a COTS tool during prototype development while not sacrificing performance in the delivered version. This desire spawned the following short, medium, and long-term goals for ESCAPE:

1) Provide a capability whereby a knowledge base developed on a COTS tool could be efficiently executed, embedded, and integrated with real-time procedural software.

2) Enable the developer to separate pure heuristics from their associated control by giving him programming constructs from both the rule-based and procedural language domains.

3) Establish a software engineering environment for rule-based systems which enforces good design methodology and a framework for hierarchical, reusable, template-based objects.

3. Approach

ESCAPE facilitates these capabilities by automatically converting the COTS prototype rule base into multitasking Ada software which satisfies the performance and embeddability demands of real-time expert systems. Presently, ESCAPE has been augmented with a control specification environment thereby achieving the first and second objectives. The object-oriented template-based representation for rule-based systems represents on-going work. Each of these evolutionary stages for ESCAPE are described below.

3.1) <u>Embedding and Integrating Knowledge-Bases</u>. The overall purpose of the ESCAPE environment is to generate knowledge-based systems that will act as *embedded components* of larger software systems. In typical expert system shells, "embedded" means that the interpretive inference engine executes as an autonomous subprocess of the software system, virtually separate from the embedding application. In ESCAPE, however, the knowledge base (KB) becomes a *standard procedural package* (e.g. an Ada or 'C' library unit) that is integrated into the software system. The KB actually becomes an object of the system with methods to dispatch operations from within the software environment. This approach results in a tight coupling of all components of the system, offering an efficient and practical means to integrate and embed KBs in software applications. ESCAPE uses a synergistic approach incorporating compilation, compaction, and parallelization of a knowledge base to re-engineer it into a more efficient form.

Compilation of the KB is accomplished by transforming the essence of the expert system from its rule-based format into a more streamlined, procedural language structure. As depicted in figure 1, the built-in inference engine of the COTS tool eases programmer development by allowing rules to be incrementally entered into the KB in an unordered fashion. Unfortunately, this convenience requires a significant amount of run time overhead associated with the iterative, interpretive, inference cycle. For a rule base which has a static data and control structure, a significant amount of the processing time is unnecessarily expended in the match and select phases of the inference engine cycle. Compilation can effectively pre-determine the execution sequence for the rules in the KB and instill organized control flow into the structure of the generated procedural code. An overhead task performed once during compilation avoids the penalty of repeated searches during run time.

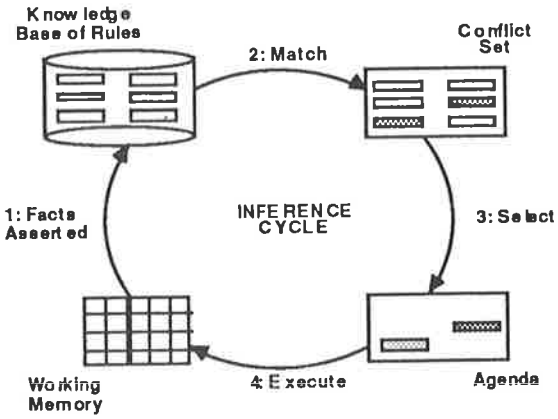


Fig. 1. Interpretive Inference Cycle.

Compaction of code size is achieved by selectively compiling only those features of the inference engine actually used by a particular KB. We observed that a typical KB utilizes only a small fraction of the total set of programmatic features provided by a COTS expert system development shell. By using a selective, RISC-like philosophy during compilation whereby only those programming constructs actually used by a particular KB are converted into equivalent Ada code, unnecessary features can be eliminated from the inference engine offering an additional savings in terms of processing time and memory space.

Parallelization is accomplished by performing an analysis of the data and control dependencies within the KB during compilation to identify and extract the implicit parallelism among rules. By partitioning the KB into groups of rules which are data independent, we can generate a set of tasks which can execute in parallel on separate nodes of a multi-processor machine, thereby converting a sequential process into a parallel one. The number of tasks extracted is left as an adjustable parameter to enable tuning for the target machine architecture. For example, given the number of nodes available on the parallel processor platform, a heuristic goal for ESCAPE would be to partition the rule base into the same number of equal-sized independent tasks.

3.2) Control Knowledge Specification

A knowledge base is intended to capture a set of heuristic statements typically performed by a human expert in assessing a situation. The application of these heuristics is what we term the *control* of the knowledge base. It is important to distinguish between knowledge itself (the rules stating condition-action pairs) and the control (specifying when to enable and fire the rules) [Erm84]. A common problem faced by knowledge engineers developing rule-based systems is how to capture and specify control information without contaminating the knowledge with tool-specific inference engine directives. Adding meta-rules or control conditions to either the predicate or action side of the rules increases the complexity of the knowledge base and destroys the clean, modular nature of the heuristics. Representing inference engine control constructs in the rule base as rules themselves undermines the premise of the rule base programming paradigm and creates a rigid, unmanageable KB. In the ESCAPE environment, we strive to separate knowledge from control by introducing two separate user interfaces -- one for the encoding of knowledge, and the other for the encoding of control.

Separating domain knowledge from control knowledge in rule-based systems has several advantages. In particular, it allows the rule base to be purely declarative in nature, and therefore, simpler. This separation also promotes the application of sound software engineering principles to the development of rule-based expert systems and avoids the performance "downfall" associated with them, i.e. the flatness of the rule base. Instead of having new data enter the system and the entire base of knowledge applied against it to compete for control of the dispatching agenda, the control environment enables the engineer to instill structure into the knowledge base. These reasons justify a separation of domain knowledge from control. By giving the user access to both programming paradigms, he can use the best representation for each type of information he needs to code : rule-based programming constructs for heuristics, and procedural programming constructs for control. In ESCAPE, the control constructs provided are those typical of most procedural programming languages.

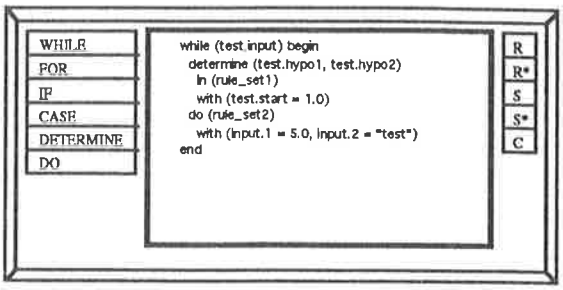


Fig. 2. Meta Interface Main Window

ESCAPE's environment for specifying rule base control was developed using a graphical interactive interface (fig. 2). This method of presentation and organization expresses two key concepts behind ESCAPE's approach to heuristic/control separation.

- No "new language" needs to be mastered by the engineer before productivity can begin. The interface is intuitively centered around the constructs of the control language; the "editor" in which the control will be specified is a menu-based, fill in the blank metaphor. The engineer will, for example, select (from the construct buttons) a while loop -- the editor then builds a template representing the while loop and all its syntax and installs blanks to be filled by the programmer.

- A graphical expression of the control allows for the language to be iteratively complex. In other words, there are default settings for the majority of the control constructs. As the user needs more complex, special purpose control, these defaults can be visualized and edited at various levels of detail. For a user needing only the minimal expression of control, the underlying mechanisms and complexities are hidden.

The left side of the control interface (figure 2) depicts the control constructs that the proposed environment will support (While, For, If, etc.). The right icon bar is where the elements of control are automatically (from the input rule base) and interactively (from the user) summarized into selection windows for use by the control engineer. The "R" icon, for example, when clicked on, will display a scrolling list -- this list identifies all rules existing in the rule base about which control can be asserted. The "R*" icon invokes a similar list structure to display the various clusters of rules to use in organizing the control.

A rule cluster (or "knowledge island") is a group of rules which inference on a related set of data. For example, in a KB which diagnosed automobile drivetrain faults, two possible clusters might be engine related rules and transmission related rules. Note that the rule clusters in the KB are automatically identified by ESCAPE during the parallelization step through data dependency analysis. In this manner, the engineer need not be burdened with separating the rules into logical groups during the development stage (although doing so would probably increase their readability). These clusters generally represent the macro-level

heuristics which an engineer needs to exert control over. For instance, he might specify that transmission related faults are to be handled only after all engine related faults have been processed. A typical control programming session flows in the following general fashion:

(1) Click on the button representing the desired control construct (While, For, If, Case, Determine, Do, Package, etc.).

(2) In the text window, the syntax for the selected control construct is displayed in template fashion (i.e., with blanks to be filled in).

(3) From the rule and rule cluster icons, select the control program variables to fill the blanks in the template. Alternatively, additional control constructs can be selected to fill the template blanks (e.g., nested loops, nested ifs).

(4) Package the set of control constructs synthesized into a "procedure" representing a unified task to be performed over the rule base.

3.3) Template-Based Knowledge Engineering

The basic "types of procedural language control" described above form the most basic statements of control that can be said about the use of the domain knowledge. However, an engineer working at this level of detailed design may lose focus on the higher-level structure of the problem. Therefore, we see the need to build a higher level of control abstraction and expression. To achieve this long-term goal, we offer an expandable set of templates that capture typical KB tasks, allow the user to select a template for use, and then allow for the encoding of knowledge and control in each applicable aspect of the template. For example, an A* search [Kor90] template could be built into the ESCAPE environment. Below, we describe Heuristic Search as one such high level problem structure and as a scenario of a knowledge template which could be prototyped in our ESCAPE system. We refer the reader to [Cha86] for a good discussion of several other problem solving structures whose incorporation into our template base is ongoing research.

The Heuristic Search Template

Many expert tasks can be viewed as sets of seemingly disjoint heuristics that need to continually assess the environment state and suggest appropriate actions to this state. Each "set of heuristics" -- a subtask in the expert domain -- applies to a set of input assignments (input state) and outputs one or more output assignments (output state(s)). This process can be viewed as a search for the "best" output state derivable from the input constraints via application of the heuristic subtasks. Thus, a built-in search template that provides the framework for a generalized search algorithm along with the most commonly used variations of constituent heuristic subtasks would be useful to the programmer. Figure 3 demonstrates the process behind the Heuristic Search template:

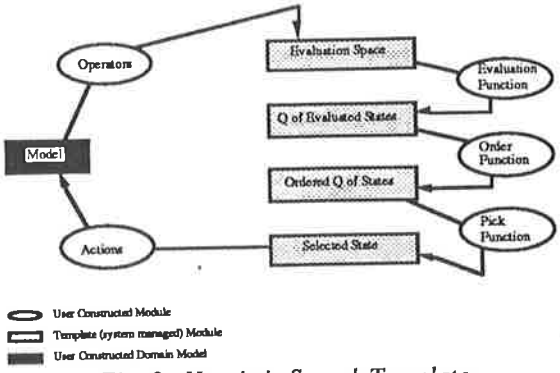


Fig. 3. Heuristic Search Template

- The clear ovals represent knowledge-based tasks that the engineer constructs in the control environment and imports to the organizational template.
- The shaded boxes represent template functions that are managed by the system and automatically expanded at compile time to form a complete embedded reasoning task.
- The black box represents the data model constructed for the domain and imported into the organizational template.

There is a tight interaction between the organization metaphor, the meta subtasking, and the encoding of the knowledge. Within the entire KB development process, the only "language" the engineer needs to be aware of is that of the domain shell. The Meta Environment is an interactive point-and-click high level specification and the Organization Templates generate forms in the domain shell for required data and knowledge input. Thus, in three levels of problem description (domain, control, and template), the underlying Match-Select-Act cycle of a rule-based inference engine was never referred to. Instead, the problems were formed around template organization structures invoking domain subtasks. The user is thereby protected from the details of the specific underlying inference engine shell. Problem decomposition along the levels of Organization, Representation, and Control alleviate the need to encode anything but domain knowledge at the level of the rule base. This is the exact goal the ESCAPE environment is striving to achieve: top-down expert system design such that all issues of control or problem solving organization are removed and captured separate from the domain knowledge base.

4. Implementation

We completed a prototype to evaluate the effectiveness of our approach. We chose to interface to a commercially available object-oriented expert systems shell, Nexpert Object. For the target compiled language, we chose Ada. Although the following discussion uses rule syntax peculiar to that of Nexpert and tasking mechanisms available in Ada, the concepts can be generically adopted to almost any rule-based expert system which needs to be converted into parallelized, compiled form. The prototype was intended to operate on a sub-set of functions found in the Nexpert KB. This sub-set of inferencing capability

4. CONTROL SYSTEM

A control system can be seen as a collection of loosely coupled data handlers, producers and consumers of time-discrete data communicating in a local area computer network. Common examples of data handlers are: *Process Interface* interacting with the controlled process and supplying current measurements to and receiving control commands from other handlers; *Data Base* storing data and control commands in a persistent data base; *Data Visualizer* displaying time-dependent data as two and three-dimensional computer graphics and video images; finally *Process Animator* displaying animated process flowcharts and interacting with the human operator. Other handlers that are of primary interest here execute instances of particular computational classes.

The key feature of a data handler is its software structure involving a data interface, a function library, a piece of real-time software and a graphical user interface.

4.1. Data Interface

The data interface, a common data bus lying at the core of the system architecture, provides a collection of procedures for the manipulation of data objects.

Data objects are repositories for temporal data. Each data object has a circular buffer storing a limited amount of the most recent data sorted by a time key of fine granularity. Due to its record-like structure, a data object can be made persistent, i.e. it can be associated with a relation within a relational data base. For long-lived objects large amounts of data can be stored in this way.

However, a data object is not merely a passive data store. Any handler can register itself as a recipient of object's data. The object will then supply the handler with current data for further processing. This data-driven mode of communication is crucial to the processing of data in real time.

Using the data interface, a handler can create a new data object and delete an existing one, store data in an object, register itself as a data recipient, wait for and receive incoming data and finally read data using a time and a component selection. Turning back to the example of two functions connected by a common data stream, the producer-consumer relation can now be expressed in terms of the data interface calls. A handler creates a data object S and successively stores in it results computed by the function A; a data handler (not necessarily a different one) registers itself as a recipient of S and provides B with data incoming from A.

Structurally, the data interface is a software layer hiding three data bases related to data objects: (1) *a volatile data base* consisting of bounded buffers and data routing software, (2) *a permanent data base* consisting of persistent relations, and (3) *a remote data base* involving network software for referencing data objects across the computer network.

4.2. Function Library

Since data handlers essentially are interpreters, the computations they carry out have to be represented as data structures.

The relational data model can be used to efficiently model such distinct computational objects as difference equations, crisp and fuzzy rules, linguistic variables, membership functions and multilayered neural networks, provided that a normalized data representation for them can be found. For example, a numerical model can be defined by a few relations for named constants and variables, expressions converted to the executable postfix form, parameter lists for input and result variables, and subprogram hierarchies. A fuzzy controller can be defined by three tables, one for the linguistic variables, one for the membership functions and one for the fuzzy rules. A backpropagation neural controller also requires three tables, one for the network layers, one for the layer units, and one for the weight matrices. The function frames, i.e. generic data streams and generic stream components, require two more relations common to all computational classes.

4.3. Real-Time Software

Obviously, from the multitude of predefined library functions only a subset is needed within an active application. When starting an application, the relational description of involved functions is transformed into resident data structures for efficient processing in real time.

The primary requirement for a real-time task is that it has to be completed within a specified period or by a deadline. There are many scheduling algorithms for real-time tasks aimed at meeting time constraints. For example, the rate monotoning scheduling theory (Sha and Goodenough (1990)) ensures that all real-time tasks meet their deadlines, provided that the overall time utilization lies below a certain bound.

However, the task execution time is never constant and is often unbounded due to transient overloads. A typical example of a transient overload is a high-variance response time of numerical iterations, AI search algorithms or communication requests. The method of imprecise computations (Liu et. al. (1994)) helps to maintain the schedulability of time-critical tasks in a system suffering from transient

overloads. A time-critical task contains a mandatory part which must be always complete and an optional part which completes under normal conditions producing a final or a precise result. Occasionally, however, the task is caused to leave the optional part unfinished and produces partial or imprecise results.

A simpler version of the imprecise computation scheme can be proposed for a data flow control system. Again, a control task consists of a mandatory part of varying duration and an optional or a 'last-chance' part of predictable duration. The mandatory part may or may not be completed by the deadline. In the former case, the task is able to deliver a desired result. In the latter case, the task leaves the mandatory part as soon as the time limit is to be reached and enters the last-chance part to provide default or approximate results.

To achieve predictable scheduling of time-critical computations, the time bounds like execution time, execution deadline, communication delay or space between events must be known. A practical method of evaluating the temporal behavior of a real-time system is run-time monitoring, i.e. collecting of performance data during a test or a normal operation. As the main activity in the data flow scheme is the event-driven propagation of time-stamped data, it is feasible to observe the timing related to data and, thus, the timing related to the functions themselves.

In summary, real-time software for a control system require an appropriate scheduling schema to guarantee predictable response times and to cope with transient overloads. For a recent survey of scheduling methods and other aspects of real-time computing see Krishna and Lee (1994).

4.4. Graphical User Interface

Each function class requires a development environment to create new class instances and to combine them into a control application. The core of the development environment is a library of function prototypes and ready-to-run applications. The visible part of the development environment is a graphical user interface.

For the numerical class, a user interface consisting of a syntax-oriented text editor and menu-driven dialogues for compiling and testing numerical expressions is appropriate. A truly graphical interface for editing the membership functions and displaying the control surface is required for a fuzzy controller. A matrix editor for a Kalman filter and a state-space observer as well as a graphic editor showing the topology of a neural network and strength of connections among the network units are further examples of class-dependent user interfaces.

When working with many function classes, the problem of guiding the user through different graphical interfaces arises. This problem will be discussed in the next section.

5. DATA FLOW GRAPHS

A visual representation of data flow is a hierarchized directed graph, whose nodes stand for functions and whose edges stand for streams.

The abstraction of hierarchy allows to replace a number of related functions by a single graph node in a recurrent manner. A node representing one hierarchy level is called *a virtual node. A real node*, on the contrary, represents a computable function, i.e. a process model or a control algorithm. Control engineers are very well familiar with the technique of decomposing complex applications into a hierarchy of simpler functions via multi-level functional block diagrams, i.e. hierarchized directed graphs.

A software tool for constructing directed graphs is a *graph editor* (for an example of an extendible graph editor see Newbery Paulisch (1993)). The graph editor supports the user by the automatic layout, checks for connection constraints and graph completeness, navigation within the hierarchy, graph persistence, etc.

The graph editor generates two different descriptions of a complete and a correct graph. The first one preserves precisely the graph hierarchy and the visual layout. This form is necessary to reconstruct a graph retrieved from a library. The second form is a ready-to-run, or a 'flat', graph free of auxiliary symbols like virtual nodes. It only consists of real nodes arranged in a topological order following the 'first-producer-then-consumer' rule. This form is necessary to run the application.

Again, the relational data model provides the underlying paradigm for describing data flow graphs. One group of relations defines data types of stream components, graphical symbols for nodes and connections, and frames for function prototypes. Another group describes the graph hierarchy and visual properites. The last defines executable graphs.

The graph editor can be regarded as a primary (or class-independent) graphical user interface interacting with secondary (or class-dependent) interfaces provided by individual data handlers. For example, if the user is going to change interactively some parameters of a running model, he navigates within the graph hierarchy until he points at a real node representing the model. As the internal structure of the model is unknown to the graph editor, it sends a message to the data handler which in turn provides the user with an appropriate dialogue.

6. SUMMARY

This section summarizes main notions used to describe the system architecture.

A control application is a collection of process models and control algorithms (represented as data flow functions) and data objects (represented as data streams). A data object is an active repository for time-discrete data. Functions are linked together via streams and have temporal properties necessary to schedule them in real time. Function entities are class members. Class examples are: Unstructured, numerical, rule-based, neural network and fuzzy controller. A control application is defined by the relational data model and represented as a directed hierarchized graph. A control system consists of a library of function prototypes, a library of ready-to-run applications and distributed software called data handlers. Data handlers execute control applications in real time.

7. DEFICIENCIES

There are three main sources of possible limitations of the proposed architecture:

- The communication and synchronization mechanisms intrinsic to the data flow scheme,
- the relational data representation, and
- the real-time programming issues.

As shown in Section 2, the data flow scheme offers a communication mechanism restricted to only two cases, non-waiting producer and non-waiting or waiting consumer. However, many control applications require a more sophisticated interaction among control tasks. For example, a recent system for a computer-aided design of a real-time robot controller ORCADD (Simon et. al. (1993)) employs as much as eight modes of interaction among the real-time module tasks. The module tasks, organized as a finite-state automaton, synchronize their activities by exchanging signals. Data flow functions, on the contrary, communicate by exchanging data. This suggests that a data flow function should be rather compared to a more complex ORCADD structure called a robot task, a collection of module tasks performing desired robotic actions. It follows that the limitation discussed here can be overcome by introducing a new function class implementing required interactions, in this case for finite-state automata. The disadvantage of this solution is that an instance of the new class becomes a rather weighty entity serving as a data interface for its internal tasks.

As described in Section 4.1, the structure of a data object is limited to a one-level data record because of the one-to-one mapping to a persistent relation. Mapping of multilevel data structures is more demanding as it requires many relations and a design procedure called data normalization. This raises the question whether a true object orientation could provide more flexibility in data structuring.

The realm of real-time programming poses some implementation-dependent problems which are hard to solve without appropriate language constructs. As a programming language Ada was chosen as it supports concurrent and to some degree real-time programming. Ada's support of concurrency is limited to tasks which are basically suitable for implementing data handlers. A typical data handler consists of two Ada tasks, *a transporter task* waiting for new data and carrying them to *a server task* which carries out actual computations. Unfortunately, Ada lacks a language construct for an asynchronous transfer of control. This means that it is not possible to interrupt a busy server task when a time limit elapses. Another deficiency of Ada is the lack of a simple synchronization mechanism such as a monitor or a protected region which is necessary to implement imprecise computations efficiently. Finally, a static priority allocation for tasks severely limits scheduling schemes which can be implemented using Ada. Fortunately, the emerging new standard for Ada9X is likely to bring much needed improvements (for details see Baker and Pazy (1991)).

8. PROTOTYPE

A prototype system called Regulus is being implemented to show the feasibility of the presented concept. Regulus is entirely programmed in Ada, as it is a general-purpose language supporting large-scale programming, software reusability, readability and modularity in addition to concurrency just mentioned. Two programming techniques related to Ada are worth mentioning, as they have a positive impact on the software quality and development time. First, this is an extensive use of generic packages for abstract data types (the concept of abstract data types is basic for object-oriented programming) implementing dynamic data structures such as ordered lists and binary trees. Second, this is building up functionality in layers by hiding complexity behind clear and simple interfaces.

Presently, a data interface supporting large data objects with up to 1000 components and providing a processing rate of several thousand data items per second is completely implemented. For the persistent data base Rdb with dynamic SQL has been used. The underlying network software for the remote data base is DECnet and TCP/IP. A prototype of the graph editor was developed using OSF/Motif (Przibylla (1993)). For the numerical class, a stack-

oriented, computationally complete machine (inspired by the MATLAB interpreter) built around a numerical library and capable of evaluating arithmetical and logical expressions in the inverse Polish notation has been developed. Handlers for other classes are in preparation. Imprecise computations will be fully implemented using the new Ada standard.

A major step in the system development process is the introduction of a new data handler. Our experience shows so far that the design of a new class library is a straightforward task requiring only a few days, whereas programming the real-time software takes much longer. Writing an elaborated, user-friendly graphical interface, however, seems to be the most time-demanding task.

9. APPLICATION

The first application of the prototype is a system for a model-based control of a waste incineration pilot plant. Several hundred analog signals acquired from a conventional process control system are processed using polynomial functions and simple statistics. In addition, a model of combustion based on mass and heat balances and a predictive model for setpoint calculation of combustion temperature, air flow, fuel flow and oxygen concentration are also computed in real time. Current development includes a closed-loop control for the setpoints calculated by the predictive model. Further plans comprise a real-time expert system for knowledge-based plant startup, supervision and shutdown. Finally, existing FORTRAN programs for statistical process control shall be integrated using the frame concept.

10. CONCLUSIONS

The data flow approach to the software integration for real-time process control is appealing because it expresses the basic duality between data and computation in a simple and elegant way. This approach is probably well suited for control applications in which the emphasis is more on computation than on communication. One of the key features of the proposed architecture is the pervasive use of the relational data model. This suggests that a control system based on this architecture is particularly well suited for large-scale applications.

REFERENCES

Agerwala, T. and Arvind (1982). Data Flow Systems: Guest Editor's Introduction. *IEEE Computer Magazine*, Vol. 15, No. 2, pp. 10-13.

Baker, T. P. and Pazy, O. (1991). Real-Time Features in Ada9X. In: *Proc. Real Time Systems Symposium*, San Antonio, 4-6 Dec., pp. 172-180.

Benveniste, A. and Aström, K.J. (1993). Meeting the Challenge of Computer Science in the Industrial Application of Control: An Introductory Discussion to the Special Issue. *IEEE Trans. on Automatic Control*, Vol. 38, No. 7, pp 1004-1010 (companion papers appeared in Automatica, Vol. 29, No. 5).

Davis, A.L. and Keller, K.M. (1982). Data Flow Program Graphs, *IEEE Computer Magazine*, Vol. 15, No. 2, pp. 26-41.

Elmqvist, H. and Mattson, S.E. (1989). Simulator for Dynamical Systems Using Graphics and Equations for Modelling. *IEEE Control Systems Magazine*, Vol. 9 , No. 1, pp. 53-58.

Krishna, C.M. and Lee, Y.H. (1994). Special Issue on Real-Time Systems (Ed.). *Proceedings of the IEEE*, Vol. 82, No. 1.

Linkens, D. A. (1993). CAD for Control Systems (Ed.), Marcel Dekker, New York.

Liu, J. W. S., Shih, W. K., Lin, K. J., Bettati, R, Chung, J. Y. (1994). Imprecise Computations, *Proceedings of the IEEE*, Vol. 82, No. 1, pp. 83-94.

Lu, Yong-Zai (1992). The New Generation of Advanced Process Control, *Control Engineering*, March, pp. 21-23.

Newbery Paulisch, F. (1993). The Design of an Extendible Graph Editor, *Lecture Notes in Computer Science*, Vol. 704, Springer-Verlag, Berlin, Heidelberg, New York.

Przibylla, S. (1993). Ein Werkzeug zur visuellen Programmierung von Datenflußmodellen (Master's Thesis in German), Universität Karlsruhe, Fakultät Informatik.

Simon, D., Espiau, B., Castillo, E., Kapellos, K. (1993). Computer-Aided Design of a Generic Robot Controller Handling Reactivity and Real-Time Control Issues. *IEEE Transactions on Control Systems Technology*, Vol. 1, No. 4, pp. 213-229.

Sha, L., Goodenough, J.B. (1990). Real-Time Scheduling Theory and Ada, *IEEE Computer*, Vol. 23, No. 4, pp. 53-62.

ACKNOWLEDGEMENTS

This work has been supported by funds of the PSA research project (Projekt Schadstoff- und Abfallarme Verfahren) of the Kernforschungszentrum Karlsruhe. The author thanks Dr. Andreas Jaeschke for his encouragement in pursuing this work, Dr. Karl Mittag for making his models available, Rolf Kerpe for his help in applying the prototype and Erika Maier and Maike Schröder for their comments on the manuscript.

CONTROLLER VERIFICATION USING QUALITATIVE REASONING

E. GAZI, W.D. SEIDER and L.H. UNGAR

Department of Chemical Engineering, University of Pennsylvania, Philadelphia, PA 19104/6393

Abstract. The performance of a heterogeneous (gain scheduling) controller is verified, for the control of a CSTR with a reversible exothermic reaction, in the face of sizable disturbances and under uncertainty. A qualitative reasoning technique, NSIM, is used to build and simulate semi-quantitative models with both parametric and non-parametric uncertainty. Bounds on the state variables are determined that are guaranteed to include all of the solutions to the uncertain model, and hence, this technique can be used for automatic proof construction.

Key Words. Control system analysis; chemical industry; qualitative reasoning; semi-quantitative models; heterogeneous controllers;

INTRODUCTION

In this manuscript, a combination of simple controllers is used for the control of a non-linear chemical process, and closed-loop stability is proven, when there is uncertainty in the model and sizable disturbances are anticipated. Since many models of chemical processes are incompletely known, it is of practical importance to design controllers that perform well when the dynamic behavior of the plant differs from that described by its model. Model uncertainty can be either structural (non-parametric) or parametric; that is, the terms in the equations that govern the plant may not be known exactly, or the parameters and the disturbances may be known only within rough bounds. Although techniques exist to deal with parametric uncertainty, the representation and simulation of nonlinear models with inexact functional relationships has not been addressed as effectively. A qualitative reasoning technique is proposed here for this purpose.

The problem of predicting behavioral bounds on uncertain systems has been addressed in several ways in control theory. For linear systems, through the use of Internal Model Control (IMC), Zafiriou and Morari (1989) have established a methodology for the design of robust controllers in the face of parametric uncertainty. Sensitivity analysis (Thornton, 1987) is often used to investigate the effect of perturbations, but involves uncertainty in parameters or initial conditions only. Monte Carlo techniques are also used to compute behavioral bounds. However, they have been applied only with uncertain parameters (see Gazi *et al.* (1994b) for a recent extension to non-parametric uncertainty), are computationally expensive (especially as the number of uncertain parameters increases), and their predictions are only guaranteed in the limit of an infinite number of samples.

Qualitative reasoning methods have been developed to handle imprecise models using the rules of qualitative mathematics (Forbus, 1984; Kuipers, 1986). They deal with states of incomplete knowledge better than differential or difference equations, since qualitative descriptions of parameters (e.g., the level of a tank between 0 and *full*) and functional relationships (e.g., pressure monotonically increases with the level in the tank) are sufficient. Thus, they permit the construction of models without assumptions about functional forms or specific values for uncertain constants. Qualitative simulations have been completed for simple chemical processes by Dalle Molle *et al.* (1988, 1990). More complicated processes are difficult to handle qualitatively because the results are too ambiguous to be useful. To circumvent this, numerical information, such as bounds on the variables and functions, has been added (Berleant and Kuipers, 1991; Kay and Kuipers, 1993). These so-called *semi-quantitative* models yield results that are more precise. Vinson and Ungar (1993) have used this technique for process monitoring and fault detection and diagnosis.

In this work, the NSIM algorithm (Kay and Kuipers, 1993) is used to verify the stability of a CSTR with a reversible exothermic reaction, controlled by a combination of simple controllers. In the next section, a brief description of the process and the controller is provided, followed by a description of the NSIM technique, as well as some results.

CSTR PROCESS

The methodology is illustrated for the control of a CSTR in which the reversible exothermic reaction

$$A \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} R$$

is carried out. Note that this reactor, inspite of its rather simple reaction, exhibits a maximum in the conversion as a function of temperature, that presents special control problems. Three ODEs, two mass balances and an energy balance, model the process illustrated in Figure 1:

$$\frac{dA}{dt} = \frac{1}{\tau}(A_i - A) - k_1 A + k_{-1} R \qquad (1)$$

$$\frac{dR}{dt} = \frac{1}{\tau}(R_i - R) + k_1 A - k_{-1} R \qquad (2)$$

$$\frac{dT}{dt} = \frac{-\Delta H_r}{\rho c_p}(k_1 A - k_{-1} R) + \frac{1}{\tau}(T_i - T) \qquad (3)$$

where $k_1 = C_1 \exp\{-Q_1/RT\}$ and $k_{-1} = C_{-1}\exp\{-Q_{-1}/RT\}$. The parameter values and the steady-state operating conditions given by Economou *et al.* (1986) are reproduced in Table 1. A, R and T represent the state variables of the system, and are the concentrations of the species and the temperature in the reactor, respectively. The inlet temperature is selected as the manipulated variable. A control objective is to operate the reactor as close to the maximum conversion as possible, while maintaining the stability of the closed-loop system in the face of sizable disturbances. The equilibrium conversion increases with the reactor temperature at lower temperatures and decreases at higher temperatures (Figure 2a). Thus, the gain of the plant changes sign as a function of the reactor temperature, which can cause a controller designed with a linearized model and a fixed gain to become unstable.

A heterogeneous controller (Kuipers and Åström, 1992) is used to control the process. It combines simple control elements across regions (usually overlapping) of state space, with a smooth transition between the different regions. It offers the advantage that each element (usually a P or PI controller) has well known and theoretically proven properties and is implemented easily. The global control law, $u\{x\}$, is defined as the weighted average of the local control laws, $u_i\{x\}$:

$$u\{x\} = \sum_i \alpha_i\{x\} u_i\{x\} \qquad \sum_i \alpha_i\{x\} = 1 \qquad (4)$$

where x is the vector of state variables and $\alpha_i\{x\}$ is a monotonic (not necessarily linear) function of x, that may be regarded as a measure of the *appropriateness* of applying the local control law i. As illustrated in Figure 3, controller 1 is used when $0 \le x \le a$, controller 2 when $b \le x \le c$, and controller 3 when $d \le x$. In the intermediate regions, ab and cd, combinations of controllers 1 and 2 and 2 and 3 are used, respectively. Such heterogeneous controllers can thus be viewed as an extension of gain scheduling (Shamma and Athans, 1990), where a specific control law is selected for a given operating region with smooth transitions between controllers as the process shifts between operating regions. The ability of heterogeneous controllers to guarantee stability (robustness) and good performance, in the face of disturbances, for simple linear processes is demonstrated by Kuipers and Åström (1992). They are also applied for the control of this CSTR process by Gazi *et al.* (1993).

A heterogeneous controller, having three elements, is implemented herein. It consists of a proportional controller having a negative gain in the low-temperatures region, a proportional controller having a positive gain in the high-temperature region and a zero-action (dead-band) controller in the region around the maximum of the conversion-temperature curve; i.e.,

$$T_i = T_{i_s} + \text{low}\{T\}\, K_\ell\, (R-R_s) + \text{norm}\{T\}\, K_n\, (R-R_s) + \text{hi}\{T\}\, K_h\, (R-R_s) \qquad (5)$$

where the functions, low{T}, norm{T} and hi{T}, represent the appropriateness measures for the low, normal and high temperature regions, respectively, as defined in Figure 2b, and K_ℓ, K_n, K_h are the corresponding controller gains with $K_\ell < 0$, $K_n = 0$ and $K_h > 0$ (Figure 2a). R_s is the set-point for R and T_{i_s} is the associated inlet temperature.

RESULTS

Although the controller is designed based on the nominal process model, it is of practical importance to verify that the closed-loop plant will remain stable in the face of disturbances, even when there is significant model uncertainty. A qualitative reasoning technique is used herein as a means to represent and simulate the inaccurate model.

QSIM (Kuipers, 1986) provides a framework for developing models in the form of qualitative differential equations (QDEs), which are abstractions of ordinary differential equations (ODEs), where the values of the variables are described qualitatively and the functional relationships between the variables may be known incompletely. The constraints are qualitative expressions involving the common mathematical operations, such as addition, multiplication, and differentiation, as well as arbitrary monotonic functions. Given a QDE and a qualitative description of an initial state, QSIM derives a tree of qualitative state descriptions, where the paths from the root to the leaves of the tree represent the possible behaviors of the system.

Usually functional relationships and parameters are computed using experimental data and are expressed with some uncertainty. In NSIM (Kuipers, 1989; Kay and Kuipers, 1993), an extension of QSIM, parameters may be assigned lower and upper bounds and functions may be bounded by lower and upper envelopes, thus forming Semi-Quantitative Differential Equations (SQDEs). With these bounds,

NSIM eliminates behaviors that are inconsistent with the numerical information, while placing bounds on the remaining behaviors. NSIM produces bounds on the trajectories of the state variables that are guaranteed to include *all* of the behaviors consistent with the approximate model.

Suppose that the SQDEs are written as a system of equations of the form $\underline{x}'=\underline{f}\{\underline{x}\}$, where f_i is an expression composed of addition, subtraction, multiplication, and division operations and arbitrary monotonic functions (monotonically increasing, M^+, or decreasing, M^-). Given a SQDE and an initial condition, NSIM derives and numerically integrates an *extremal* system of ODEs, whose solution is guaranteed to bound all solutions of the SQDE. The extremal equations are generated by computing, for each state variable x_i, the lower and upper bounds on the first derivative (i.e., the expressions $L\{f_i\}$ and $U\{f_i\}$) using the rules in Table 2, as proven by Kay (1991). The table is applied recursively to the subexpressions of f_i. Minimal and maximal equations are derived to define a *dynamic envelope* for each state variable, and hence, a n-th order system results in a 2n-th order system.

Once the extremal system is found, it is integrated using a standard integration technique such as the Runge-Kutta method. It is theoretically proven (Kay, 1991) that, when the extremal system bounds the solution of the SQDE at t = 0, it bounds the solution at all times.

Simulations of the reactor with the heterogeneous controller, using NSIM, were carried out for several disturbances under parametric and non-parametric uncertainties. The goal was to verify the conditions under which the controller is guaranteed to keep the process stable. As explained in Gazi *et al.* (1994a), NSIM determines bounds that are guaranteed to include all of the solutions to the SQDEs, but these are conservative. Thus, when the NSIM bounds are stable, it is guaranteed that the family of ODEs represented by the SQDEs is also stable. However, when a NSIM bound is unstable, no conclusions are possible. At least one of the sets of ODEs represented by the SQDEs is unstable (but this may be a physically unrealistic one).

Gazi *et al.* (1994a) identify the source of the conservatism of the NSIM algorithm and propose improvements to reduce it. Figure 4 shows the response of the system for a disturbance that adjusts the reactor temperature such that T = 435 K, as predicted by NSIM with and without the improvements. Even with small parametric uncertainty, the bounds produced by the pure NSIM algorithm diverge, although the closed-loop system is stable.

Using the improved NSIM technique, it can be shown that the plant remains stable for a range of the controller gains. Figure 5 shows the bounds on the response for a step change in A_i ($A_i \in$ [0.85, 0.90] mol/L), when $K_\ell \in$ [-1050, -950] and $K_h \in$ [90, 100].

Moreover, stability can be proven in the face of sizable parametric and non-parametric uncertainties and disturbances. As an example, stability is ensured for the case when a disturbance is introduced that adjusts the reactor temperature within [415, 420] K, as illustrated in Figure 6. In this case, there is parametric uncertainty ($A_i \in$ [0.85, 0.90] mol/L) and non-parametric uncertainty in the rate constants:

$$11\ T \exp\{-Q_1/RT\} \le k_1 \le 5200 \exp\{-Q_1/RT\}$$
$$0.99\ 10^6\ T \exp\{-Q_{-1}/RT\} \le k_{-1} \le 2.4\ 10^3\ T \exp\{-Q_{-1}/RT\} \quad (6)$$

For larger disturbances, however, the NSIM bounds may diverge, due to conservatism, even when the plant is actually closed-loop stable, as shown in Gazi *et al.* (1994b).

CONCLUSIONS

NSIM can be used effectively for controller verification. It utilizes functions and parameters that are imperfectly known to determine bounds that are guaranteed to include all possible solutions of the uncertain system. Thus, it can be used for automatic proof of stability. Given a set of semi-quantitative differential equations, NSIM automatically derives an *extremal* system of ODEs, by bounding the first derivatives of the state variables. Although the bounds produced by the pure NSIM algorithm are very conservative, an improved algorithm (Gazi *et al.*, 1994a) can provide useful results.

This technique is used to verify closed-loop stability for a CSTR with a reversible exothermic reaction, under parametric and non-parametric uncertainty and in the face of sizable disturbances. It is shown that a combination of simple controllers is adequate to keep the reactor stable under the class of disturbances considered, for considerable model uncertainty and for a range of controller gains.

As compared to existing techniques, NSIM offers the advantage of providing proofs for nonlinear systems, even with non-parametric uncertainty. However, the results, even of the improved NSIM, may be very conservative, especially for highly coupled systems. Therefore, NSIM is more effective for combinations of simple controllers such as in the heterogeneous controller used herein. The technique would be more difficult to implement for model-based controllers (e.g., a Model-predictive Controller) where the NSIM bounds should be too broad to draw meaningful conclusions.

In the future, the technique will be tested for more complex chemical processes with sizable uncertainties and disturbances, including faults. An important application of this methodology is likely to be for the automatic verification of *safety net* control actions to prevent plants from moving into hazardous operating regimes when faults are encountered. In this application, the stability of the controller can be tested with respect to a library of

faults that often occur in the chemical industry.

Acknowledgments. The authors are grateful to B.J. Kuipers and H. Kay for their help. Partial funding provided by the Computer Integrated Engineering Program of the NSF under Grant No. DDM-9114080 is gratefully acknowledged.

REFERENCES

Berleant, D., and B.J. Kuipers, *Bridging the Gap from Qualitative to Numerical Simulation*, Technical Report AI91-158, Artificial Intelligence Laboratory, University of Texas, Austin, 1991.

Gazi, E., W.D. Seider, and L.H. Ungar (1993). "Control of Nonlinear Processes Using Qualitative Reasoning", *Comput. Chem. Eng*, **18**, S189.

Gazi, E., W.D. Seider, and L.H. Ungar (1994a). "Control of Nonlinear Processes Using Qualitative Reasoning", submitted to *Automatica*.

Gazi, E., W.D. Seider, and L.H. Ungar (1994b). "Control of Nonlinear Processes Under Non-parametric Uncertainty", Proceedings of IFAC Symposium ADCHEM '94, May 25-27, Kyoto, Japan.

Dalle Molle, D.T., and T.F. Edgar, "Qualitative Modeling of Chemical Reaction Systems", in *Artificial Intelligence in Process Engineering*, Academic Press, Boston, 1990.

Dalle Molle, D.T., B.J. Kuipers, and T.F. Edgar (1993). "Qualitative Modeling and Simulation of Dynamic Systems", *Comput. Chem. Eng*, **12**, 853.

Economou, C., M. Morari, and B. Palsson (1986). "Internal Model Control. 5. Extension to Nonlinear Systems", *Ind. Eng. Chem. Proc. Des. Dev.*, **25**, 403.

Forbus, K.D. (1984). "Qualitative Process Theory", *Artificial Intelligence*, **24**, 85.

Kay, H., *Monitoring and Diagnosis of Multi-Tank Flows Using Qualitative Reasoning*, Master's Thesis, The University of Texas, Austin, 1991.

Kay, H., and B.J. Kuipers, "Numerical Behavior Envelopes for Qualitative Simulation", in Proceedings of the National Conference on Artificial Intelligence (AAAI-93), AAAI/MIT Press, in press, 1994.

Kuipers, B.J. (1986). "Qualitative Simulation", *Artificial Intelligence*, **29**, 289.

Kuipers, B.J. (1989). "Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge", *Automatica*, **25**, 571.

Kuipers, B.J., and K. Åström, "The Composition and Validation of Heterogeneous Control Laws", *Automatica*, in press, 1994.

Shamma, J., and M. Athans (1990). "Analysis of Gain Scheduled Control for Nonlinear Plants", *IEEE Trans. Automatic Control*, **3518**, 898.

Thornton, K.W., "Sensitivity Analysis and Simulation Experimentation", in *Systems and Control Encyclopedia - Theory, Technology, Applications*, Madan O. Singh (editor), Oxford, 4227, 1987.

Vinson, J. M., and L.H. Ungar, "Qmimic: Model based Monitoring and Diagnosis", in *Proc. American Control Conference*, IEEE Service Center, San Fransisco, 1880, 1993.

Zafiriou, E., and M. Morari, *Robust Process Control*, Prentice Hall, Englewood Cliffs, NJ, 1989.

Table 1 Constants and steady-state operating conditions for the CSTR process (Economou *et al.*, 1986).

τ	= 60 s	A_i	= 1.0 mol L^{-1}
C_1	= 5 x 10^3 s^{-1}	R_i	= 0.0 mol L^{-1}
C_{-1}	= 1 x 10^6 s^{-1}	A	= 0.4912 mol L^{-1}
Q_1	= 10000 cal mol^{-1}	R	= 0.5088 mol L^{-1}
Q_{-1}	= 15000 cal mol^{-1}	T_i	= 435.9 K
R	= 1.987 cal mol^{-1} K^{-1}	T	= 438.4 K
$-\Delta H_r$	= 5000 cal mol^{-1}		
ρ	= 1 kg L^{-1}		
c_p	= 1000 cal kg^{-1} K^{-1}		

Table 2 Translation table for extremal expressions. Let $\beta\{f_i\}$ be the lower or upper bound on x_i' (β=L or β=U). x_i is the state variable with derivative f_i, x_j is any other state variable, c is a constant, and A and B denote any subexpression. The subscript ℓ (u) denotes lower (upper) bound or envelope. Reproduced from Kay and Kuipers (1993).

e	L{e}	U{e}
c	c_ℓ	c_u
x_j	$L\{x_j\}=x_{j\ell}$	$U\{x_j\}=x_{ju}$
x_i	$\beta\{x_i\}$	$\beta\{x_i\}$
A+B	L{A}+L{B}	U{A}+U{B}
AxB	L{A}xL{B}	U{A}xU{B}
A-B	L{A}-U{B}	U{A}-L{B}
A+B	L{A}+U{B}	U{A}+L{B}
-A	-U{A}	-L{A}
$M^+\{A\}$	$M^+_\ell[L\{A\}]$	$M^+_u[U\{A\}]$
$M^-\{A\}$	$M^-_\ell[U\{A\}]$	$M^-_u[L\{A\}]$

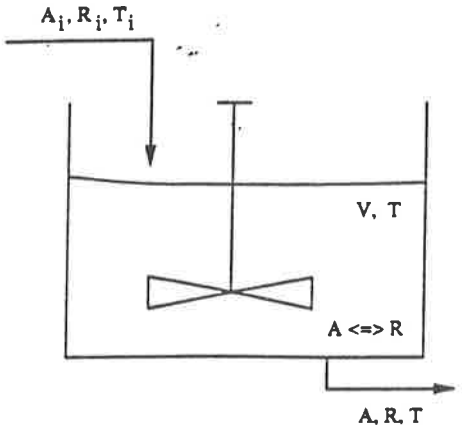


Fig. 1. CSTR with reversible exothermic reaction.

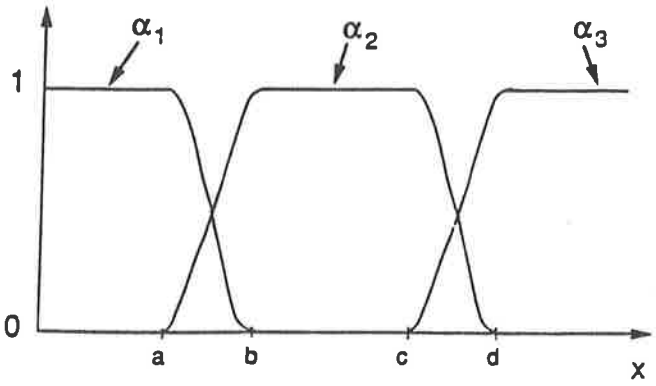
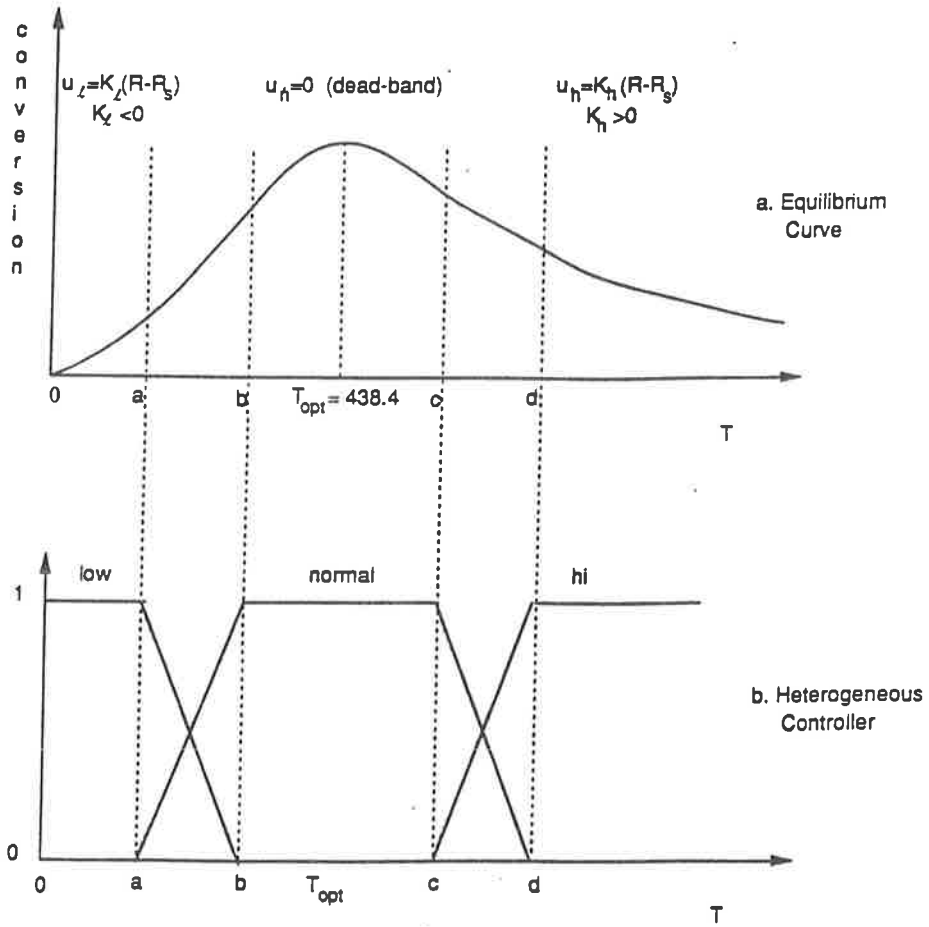


Fig. 3. Appropriateness measures for a 3-element heterogeneous controller.



$$T_i - T_{i_s} = low^{*}u_l + normal^{*}u_n + hi^{*}u_h$$

Fig. 2. Equilibrium curve for the exothermic reaction A<-> R in a CSTR, and the heterogeneous controller.

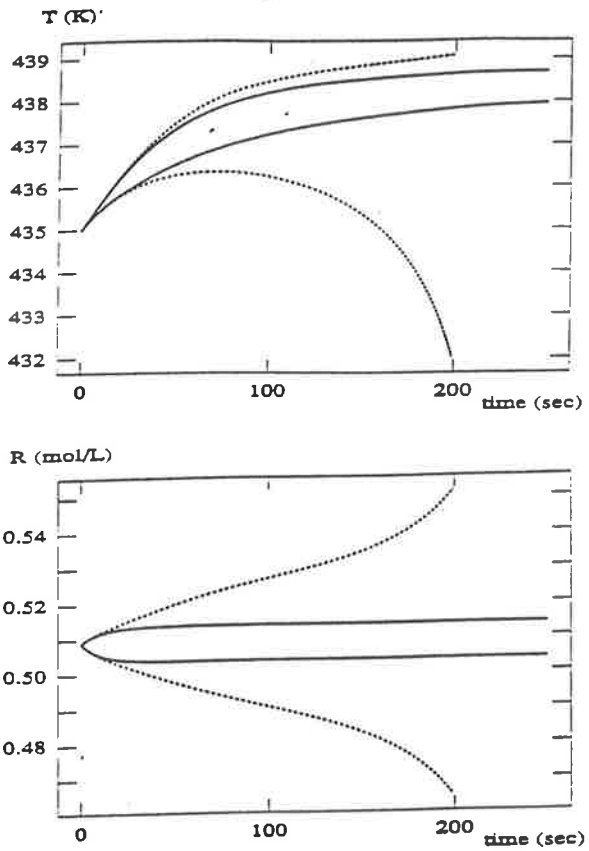


Fig. 4. Bounds on the reactor response for a disturbance that adjusts the reactor temperature to 435 K, with (___) and without (- - -) the NSIM improvements, and $C_1 \in [4900, 5100]$.

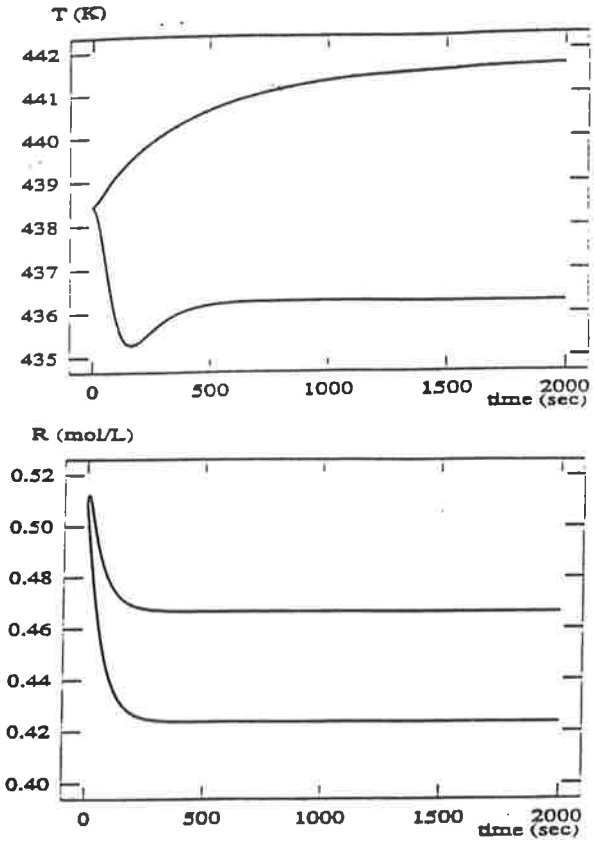


Fig. 5. Bounds on the reactor response for a step change such that $A_i \in [0.85, 0.90]$, $K_l \in [-1050, -950]$, $K_h \in [90, 100]$, and $C_1 \in [4800, 5200]$.

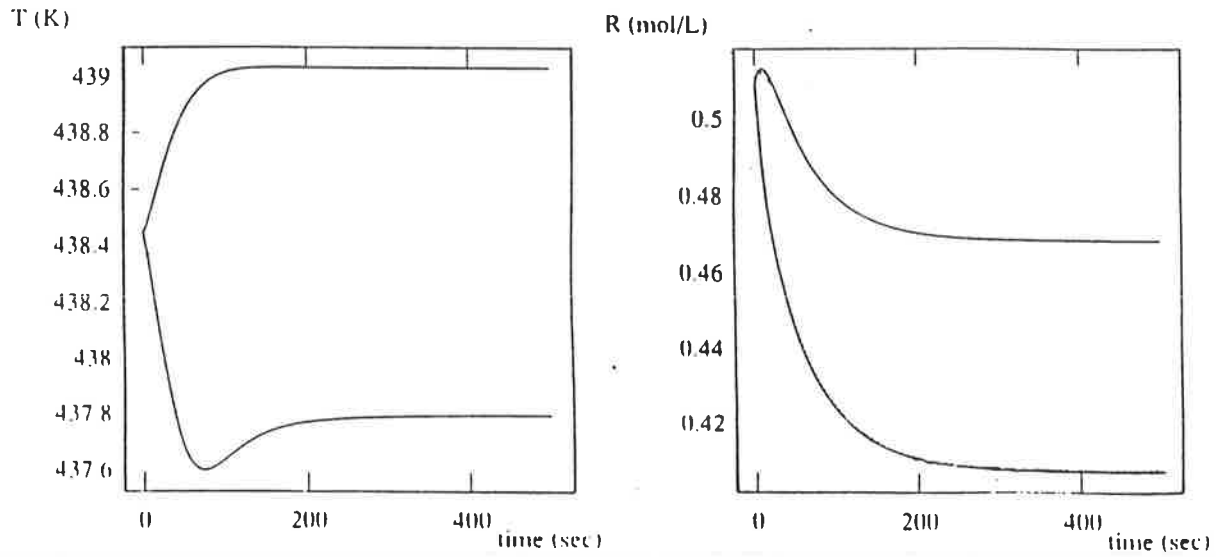


Fig. 6. Bounds on the reactor response for a disturbance that adjusts the reactor temperature such that [415, 420] K, with $A_i \in [0.85, 0.90]$ and non-parametric uncertainty in the rate constants (Eqn. (6)).

TIME-DEPENDENT SYSTEM KNOWLEDGE REPRESENTATION BASED ON DYNAMIC MPLD

Y-S. HU and M. MODARRES

Center for Reliability Engineering, Department of Materials and Nuclear Engineering
University of Maryland, College Park, MD 20742, USA

Abstract: In this paper, the Dynamic Master Plant Logic Diagram (DMPLD) is introduced for representing the time-dependent behavior of complex physical systems (feedback, time-lagged dependency, autocorrelation, trend, etc.). Experts may decompose a complex system hierarchically, and model the dynamic behavior of its subsystems and components separate from each other. A DMPLD is a hierarchical structure which enable a user to decompose a dynamic system into dynamically stable and independent subsystems. Using a DMPLD, degree of success (or failure), approximate full-scale system physical values, and transition effects in the system can be used. An example of using DMPLD for controlling a dynamic system is presented.

Key Words: Reliability theory, System analysis, Time-domain analysis, Dynamic response, Fuzzy control, Expert systems, Knowledge engineering, Intelligent building control.

1. INTRODUCTION

As we look back upon the history of science and engineering, we cannot help but notice the importance of knowledge representation. Knowledge representation supports a medium to communicate and share knowledge among people. A good representation method may also become a useful tool for system analysis and design.

As a research area in its own right, knowledge representation has evolved within the field of artificial intelligence (AI), where it continues to play a central role. Improvement in computer capabilities has enabled us to transfer specific fields of knowledge into an expert system. The procedure for transferring system knowledge into a computer acceptable representation is a two stages process:

1. development of a logical representation of the general system and its component interactions,
2. development of a dynamic representation describing physical behavior of the system and its components.

While modelling a system-level expert system, the knowledge representation often involves elements of inescapable complexity. It has been shown by many researchers (Simon, 1982; Rasmussen, 1985) that most of the complex systems are formed through some hierarchic evolution. As such, they can be best described and represented through hierarchic frameworks. For system knowledge representation, logic diagrams (e.g., fault tree, event tree, success tree, goal tree, master plant logic diagram, etc.) are proven to be powerful hierarchic methods to represent the system knowledge (Kim and Modarres, 1987; Poucet, 1990; Chen and Modarres, 1992; Dezfuli, Modarres and Meyer, 1994; Hu, Modarres and Marksberry, 1994). While these representations are popular for performing the first stage described above, they are limited to model the time-dependent behavior of the system (i.e., how the system or its subordinates acts or reacts to the external or internal environmental changes occurred during its operating). The modeling of the dynamic behavior (i.e., the second stage described above) is the focus of this paper.

A variety of mathematical simulators have been developed and applied to plant operations. Thus, book knowledge may be transfered into complex physical relationships to simulate and predict the dynamic behavior of the system. However, this would be very costly, inefficient, and most of the time unnecessary.

Some researchers such as de Kleer (1984) and Felkel (1990) have proposed dynamic models based on naive physics to simplify the mathematical relationships of systems and infer behavior from the system structure. Other researchers (Kosko, 1992; Sugeno 1993) have used fuzzy logic concept for modelling dynamic relationships. The naive physics has high potential for representing complex relations. The problem is, however, that for large systems in use today, the complexity of the knowledge will be prohibitive for a naive physics approach. A decomposition algorithm (e.g., MPLD) can alleviate some of the shortcomings. However, MPLD is a static and logically based approach. In this paper, we will report the extention of the traditional MPLD models to dynamic modelling of complex systems. The MPLD method has been the basis for knowledge representation (the first stage) in our research.

The MPLD has proven to be a powerful method for modeling complex physical systems (e.g., nuclear power plant), and clearly shows the interrelationship between functions, systems, subsystems and components, as shown in Fig. 1. The MPLD can logically and hierarchically display the functions, subfunctions and the manner in which various hardware, software, and people interact with each other to achieve these functions. In essence, the hierarchy of the MPLD is displayed by a dependency matrix in form a lattice. For each subsystem, hardware, human, etc. , the success paths are shown on the top. On the left side of the lattice, the set of supporting elements needed are displayed. Since these supporting elements may require other supporting elements, another hierarchy and interrelationship may

also be shown to display this complexity. Informal relationships such as common cause failures may also be modeled in MPLD structures. More details about MPLD is discussed by Hunt and Modarres (1987), Modarres (1992), Dezfuli *et al.* (1994), and Hu *et al.* (1994).

Since one important characteristic and advantage of the MPLD is its lattice representation which models system-to-system (or subsystem-to-subsystem) interactions, it allows for an easy integration of dynamic fuzzy logic concept into the classical MPLD. In this concept, the physical behavior is assumed to be represented by interactions between various levels of hierarchy. The degree of interactions is governed by the physics laws which describe this interaction. Complex physical relations can be represented by fuzzy relationships (rules). Therefore, the system can be modeled using time-dependent fuzzy logic. Accordingly, the logical relationships modeled in an MPLD is accompanied by a corresponding physical relationships by fuzzifying such relationships. Therefore, it is highly desirable to develop a new diagram that combines the MPLD (the first stage process) and the fuzzy logic concept (the second stage process) together to analyze the dynamic effect or behavior of the system (e.g., due to an incipient failure). This new diagram is called Dynamic Master Plant Logic Diagram (DMPLD).

The basic concept of using the Dynamic Master Plant Logic Diagram (DMPLD) to model the transition behavior of plant systems has been discussed by Hu and Modarres (1994a, 1994b), and will be reviewed in this paper. Conceptually, the DMPLD allows the experts to specify the degree of success (or failure) in realized a desired operating function, approximate temporal values of the govering parameters, and transition effects of a subsystem on other subsystem or on the system, etc..

In this paper, time-dependent behavior of physical systems is classified and discussed in Section 2. Time-dependent fuzzy logic and the basic concepts of the DMPLD are reviewed in Section 3. Examples of using the DMPLD to organize different types of time-dependent behavior are discussed in Section 4. As an example, the DMPLD representation for an intelligent building control system is discussed.

2. TIME-DEPENDENT BEHAVIOR OF PHYSICAL SYSTEMS

Conceptually, time-dependent behavior can be classified in a variety of different ways. Table 1 shows a classification based on response time. As such, time-dependent behaviors of a physical system has one or more of the following properties:

- Transition.
  A transition addresses an ongoing process which a system is transferring from one static state to another in a deterministic time. A transition can be stopped only if the condition is no longer held or the target static state is reached. For example, when a tank is leaking (a behavior), the tank transitions from a full level (state) to no water (another static state). If we stop the leak, behavior is changed to another stable state.
- Trend.
  Trend is the potential that a system will reach a specific state (probabilistic output) based on the past or current deterministic history of the system. For example, if we continue pumping water into the tank, it will overflow sometimes in the future. This is different than a transition, since tank-overflow (behavior) is not ongoing, but it will happen if we do not change the current trend.
- Scheduled Transition (or Trend).
  A scheduled transition (or trend) is a standby status that system is scheduled (not ongoing) to start a transition (or a trend), sometimes in the future. For example, assume a redundant battery can only provide power for 5 minutes. As such, once the redundant battery is enabled, the system is scheduled to lose its power 5 minutes later if no other power source is enabled within this period.
- Uncertain Transition (or Trend).
  An uncertain transition (or a trend) addresses that the behavior is caused randomly because of the uncertain nature of a system status or lack of knowledge about the system response. A transition (or a trend) will be triggered if such a random input happens. For example, an operator accidently pressing a wrong button may trigger a transition (or a trend).

Table 2 shows another popular classification which is based on variable input-output relations (Pankratz, 1991). By such, some major time-dependent behaviors are:

- Time-Lagged.
  Time-lagged relationship assumes that the output variable is affected by the current and the past data of other variables.
- Autocorrelated.
  Autocorrelated (self-correlated) addresses that the output variable may depend on its own past values.
- Feedback.
  Feedback addresses that the system returns part of the output as its own inputs, especially so as to modify the output.

We will show in the next section that a DMPLD can represent and model the time-dependent behaviors classified in Tables 1 and 2.

Table 1 Time-Dependent Transition/Trend Behavior

		OUTPUT RESPONSE TIME			
		past	current	deter-ministic future	prob-abilistic future
INPUT TIME	past	static	transition	transition	trend
	current	---	static	transition	trend
	deterministic future	---	---	scheduled transition	scheduled trend
	probabilistic future	---	---	uncertain transition	uncertain trend

Table 2 Major Time-Dependent Behaviors Classified by Variable Relations

	current X(t)	current X(t)	current X(t)	current X(t)
	which is a function of			
current X		✓		✓
past X		✓		
current Y	✓		✓	
past Y	✓			
	is called			
	time-lagged	auto-correlated	static	feedback

Y ∈ { variables other than X }

3. DYNAMIC MPLD

DMPLD is a logic-based diagram. It is an extention of the GTST and MPLD decomposition method by using the time-dependent fuzzy logic. The GTST structure was initially developed as part of a DOE sponsored research (Hunt and Modarres, 1984) to assess and improve information systems in a nuclear plant. Subsequently, it has been further developed and extended to a number of other applications (Kim and Modarres, 1987; Hunt and Modarres, 1987; Birky, McAvoy and Modarres, 1988; Chung and Modarres, 1989; Sebo, Marksberry and Modarres, 1989; Chen and Modarres, 1992; Modarres, 1992; Dezfuli *et al.*, 1994; Hu *et al.*, 1994).

MPLD is designed for two purposes. One is to show systems or subsystems, their functions. For this an GTST is used. The other purpose is to represent the interrelationships between various subsystems, especially between the main subsystems and the so-called support subsystems (subsystems that power, control, cool main system). A GTST_MPLD diagram is shown in Fig. 1. More detail about GTST_MPLD analysis and application has been discussed in (Hu *et al.*, 1994).

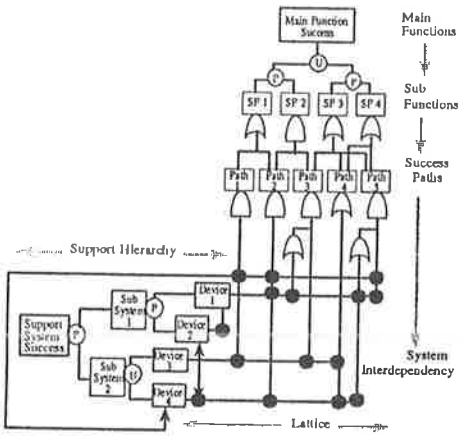


Fig. 1. A GTST_MPLD Diagram

Once GTST_MPLD is used to decompose a large system into locally isolated subsystems, DMPLD is applied to model the dynamic behavior of these subsystems, as shown in Fig. 2.

In this section, we first introduce the basic concepts of time-dependent fuzzy logic, and then explain the notations and symbology of a DMPLD. Examples of using DMPLD to model and represent time-dependent behaviors are discussed at next section.

3.1. Time-Dependent Fuzzy Logic Operations

Fuzzy logic is based on the concepts of fuzzy sets and symbolic logic (Zadeh, 1976). Here, "time-dependent" addresses cases where the system states are also a function of time.

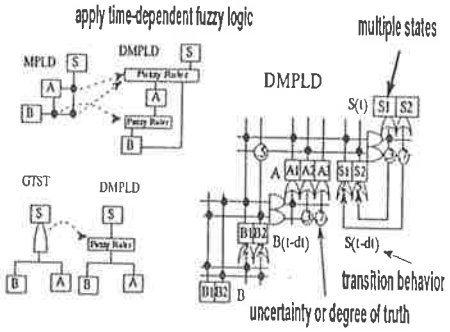


Fig. 2. From GTST_MPLD to DMPLD Modelling

Similar to the Boolean logic algebra, dynamic fuzzy logic may be represented by slightly different expressions. Classical fuzzy logic operations are time-independent (i.e. static). The major relations of interest in static fuzzy logic are described below:

1. Commutative Law: $A \cap B = B \cap A$; $A \cup B = B \cup A$
2. Associative Law: $A \cap (B \cap C) = (A \cap B) \cap C$; $A \cup (B \cup C) = (A \cup B) \cup C$
3. Distributive Law: $A \cap (B \cup C) = (A \cap C) \cup (A \cap B)$; $A \cup (B \cap C) = (A \cup C) \cap (A \cup B)$
4. Idempotent Law: $A \cap A = A$; $A \cup A = A$
5. Absorption Law: $A \cap (A \cup B) = A$; $A \cup (A \cap B) = A$
6. Negation Law: $\overline{(\overline{A})} = A$
7. deMorgan's Law: $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$; $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$
8. Kleene's Law: $(A \cap \overline{A}) \cup (B \cup \overline{B}) = B \cup \overline{B}$; $(A \cap \overline{A}) \cap (B \cup \overline{B}) = A \cap \overline{A}$

*Complementation Law: Since there are overlaps between fuzzy sets,

$A \cap \overline{A} \neq \emptyset$

$A \cup \overline{A} \neq \Omega$

Kleene's law is a degenerated form of the Complementation law. In fuzzy logic, Kleene's law is used to replace the Complementation relation. Typically, time-dependent (i.e., dynamic) fuzzy logic obeys the static operation relations described above. However, since the time effect is considered, we have introduced new rules to represent the dynamic behaviors.

Let capital A and B denote fuzzy sets. For a fuzzy set A, when its state is changed or the set is activated, A^t indicates that a delay time t is required to perform this change or activation. Suppose F(•) denotes a logic relationship of component sets. Accordingly, $F(\bullet)^t$ represents requirement of a delay time t to perform this relationship, when the state of one or more of its components are changed. The time-dependent operation rules which are supplemental to the rules discussed above are:

9. Delay law:

$$A^0 = A$$
$$(A^a)^t = A^{t+a}$$
$$(\overline{A^t}) = \overline{A}^t$$
$$F(\bullet)^0 = F(\bullet)$$
$$F(\bullet)^t = F(\bullet^t)$$
$$F(A^a, B^b, \ldots)^t = F(A^{t+a}, B^{t+b}, \ldots)$$

3.2. Time-Dependent Fuzzy Logic Evaluations

The procedure of fuzzy logic evaluation is not universal. Bellman and Giertz (1973) argued that [also discussed by Alsina, Trillas, and Valverde (1983)], under reasonable hypotheses (especially distribution law), the only truth-functional logical evaluation for fuzzy sets are the usual min-max operations. In this research, however, since the delay time is considered, this evaluation shall be time-dependent. Let Dmf(S(t)) denote the degree of membership function of S(t) at a given time t. For inputs A and B, the fuzzy evaluation procedure for S(t) may be defined as:

$S \doteq A^a \cap B^b$ → Dmf(S(t))
= min[Dmf(A(t-a)), Dmf(B(t-b)), T(S)],

$S \doteq A^a \cup B^b$ → Dmf(S(t))
= min[max[Dmf(A(t-a)), Dmf(B(t-b))], T(S)],

$S \doteq$ NOT A^a → Dmf(S(t))
= min[1 - Dmf(A(t-a)), T(S)],

$S \doteq A^a$ → Dmf(S(t))
= min[Dmf(A(t-a)), T(S)].

where T(S) is the uncertainty or the degree of truth of the relation described by the symbol $\doteq$.

To understand the importance of the dynamic logic operations, let's compare the following two logic relations:

$$A = X \cap Y \quad \text{and} \quad B = X \cap Y^t$$

where t is a delay time and $X = \overline{Y}$.

In Boolean set A, when X and Y change their states (due to a given behavior), A will change its state simultaneously. Fuzzy set B is different in that Y takes a delay time t to change its states. Figure 3 shows the results of A and B when we switch the states of X and Y. Pulses are generated when X changes slowly between 0 to 1.

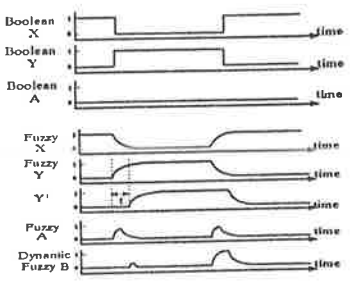


Fig. 3. A Comparison Between $A = X \cap Y$ and $B \doteq X \cap Y^t$; where X = NOT Y and T(S) = 1.

3.3. Notation for Uncertain Relations in DMPLD

In the previous section, the degree of true relationship T(S) between various systems or subsystems was considered. Such a degree of truth can be represented by fuzzy logic. Figure 4 shows all the major uncertainties modeled in term of a fuzzy logic in a MPLD, where

- state of A (or B) may be uncertain;
- relationships between S, A and B may be uncertain;
- logic relation $\cup$ (or $\cap$) may be uncertain.

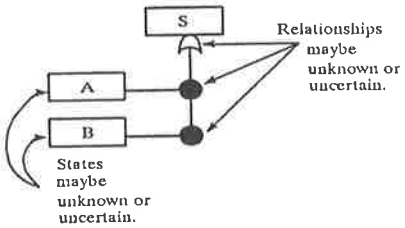


Fig. 4. MPLD Uncertain Relations

For example, consider a relationship S where we are uncertain whether it is equal to A $\cup$ B or A $\cup$ C. Assuming that 40% S = A $\cup$ B, and 60% S = A $\cup$ C, the notation may be:

$$S = [\, T_1 \cap (A \cup B)\,]_{T1\,-\,0.4} \cup [\, T_2 \cap (A \cup C)\,]_{T2\,-\,0.6}$$

If T_1 and T_2 represent the possibility of outcomes for S = A $\cup$ B or S = A $\cup$ C respectively, S may be either A $\cup$ B or A $\cup$ C.

If T_1 and T_2 are degree of truth relationship (fuzzy relationship), S may be both A $\cup$ B and A $\cup$ C, in which sum of degrees may not necessarily add up to one. For example,

$$S = [\, T_1 \cap (A \cup B)\,]_{T1\,-\,0.7} \cup [\, T_2 \cap (A \cup C)\,]_{T2\,-\,0.6}$$

The above expression assumes that membership function of S will not be larger than 0.7.

3.4. Notations and Symbols of DMPLD

Figure 2 showed a basic DMPLD structure. Table 3 summarizes the notations of the DMPLD. Four types of logic gates are designed to represent the fuzzy logic rules. Additionally, five different dependency-matrix nodes are used in DMPLD to describe the probability and the degree of truth in relationships. Using these notations and symbols, we may organize a DMPLD to improve the MPLD to represent the time dependency and the uncertainty.

4. EXAMPLES OF DMPLD REPRESENTATION

When we talk about the DMPLD representation, we are talking about a family of models (diagrams). That is, for representing a physical behavior, there is no unique DMPLD model; rather, experts can come up with varieties of DMPLDs and different numbers of nodes and layers, fuzzy sets, and transition logic. However, these DMPLDs should yield approximately similar results.

Table 3. Basic Symbols of DMPLD Based On Dynamic Fuzzy Logic

Notation	Description
	'AND' gate: the minimum value of inputs will be the output value.
	'OR' gate: the maximum value of inputs will be the output value.
	'AND' transition gate: if a new value is reached before its previous value is sent out, the new value will override the old value, and the transition time is recounted accordingly.
	'OR' transition gate: the values will be sent out sequentially according to their arrival time.
	Uncertain node: the number inside the node represents the uncertainty of relationship. The minimum between the input degree of membership function (dmf) and the uncertainty is selected as the output.
	Solid bullet node (i.e., certain node): represents a certain relationship that directly propagate the input dmf to the output dmf.
	Hollow nodes (with no number inside): means certain negation in which the output dmf is (1 - input dmf).
	Uncertain negation: a hollow node with a bar above the degree of certainty inside. The output dmf is the minimum between the uncertainty and (1 - input dmf).
	Simple-crossed line: means no relationship (i.e., independent nodes).

4.1. Basic Structures in DMPLD

The structures in DMPLDs can be grouped into seven basic classes: static, uncertain (or priority) output, uncertain (or weighted) input, scheduled, time-lagged, autocorrelated, and feedback. A DMPLD model (diagram) is assembled from these basic structures.

- Static.
  A static DMPLD represents a linear or nonlinear physical relation which is certain and not varied with time. For example, consider a give behavior described by the analog curves which relate O(t) to A(t) and B(t). Figure 5(b) represents an equivalent DMPLD which describes these analog relations in a fuzzy relation model. Various fuzzified values of A(t) and B(t) yield the simulation result shown in Fig. 5(c).
- Uncertain (or Priority) Output.
  When the output is not unique for given inputs, we take advantage of uncertain nodes in the lattice of the DMPLD. For example, as shown in Fig. 6, one may turn on a fan, or open a window while feeling hot. And, of course, for different people, the comfort range is varied, so the fan speed is also uncertain.
- Uncertain (or Weighted) Input.
  Sometimes, we are uncertain about the effect of specific input, or the input has only partial effects. Uncertain nodes can be applied on the left lattice of a DMPLD to represent an uncertain or weighted input. For example, in a light rain, experience of driver would not affect safety. For heavier rain, degree of safety depends on the driver's experience. Assuming that a threshold exist at almost 0.7 (degree of experience), Fig. 7 shows such a DMPLD representation of the problem.

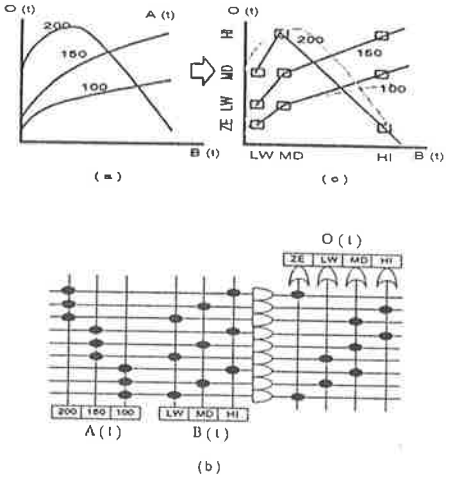


Fig. 5. A Static DMPLD

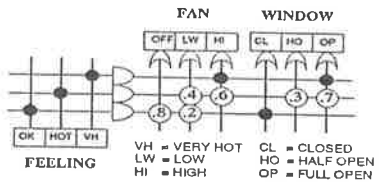


Fig. 6. A DMPLD with Uncertain Outputs

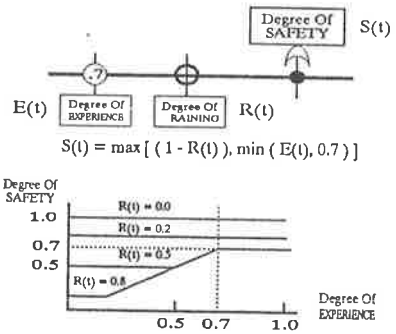


Fig. 7. A DMPLD with Weighted Input.

- Scheduled.
  Some outputs happen after a require time has elapsed. As such, transition logic gates can be applied to represent such this effect. For example, as shown in Fig. 8, an automatic dishwasher takes 15 minutes to wash, 15 minutes to rinse, and 30 minutes to dry dishes. (one can also use transparency of rinsing water and humidity of drying air to control the timer. Such a DMPLD is static.). At t=0, the washer is turned on to either start a wash, a rinse or dry.
  Here, the AND transition gate performs a crisp state change when it reachs the delay time.

- Time-Lagged.
Sometimes, the output depends on not only the current inputs but also on the past inputs. For example, the rate of acceleration can be calculated from the current and the past velocity, as shown in Fig. 9.
- Autocorrelated.
Autocorrelation addresses an output which depends on its own values in past. Sometimes, while we have difficulty to identify the inputs describibg a specific behavior (e.g., stocked), we may assume the hidden input-output relation may be represented via an autocorrelation. For example, Figure 10 shows a system's output which depends on its previous two outputs. Here, the OR transition gate performs a continueous full-scale state change.
- Feedback.
When the output is affected by the inputs and in turn, the inputs are affected by the output, a feedback relation exists. The structure of a DMPLD with feedback is similar to an autocorrelated DMPLD but the output can return to itself (as autocorrelated) or other variables, as shown in Fig. 11.

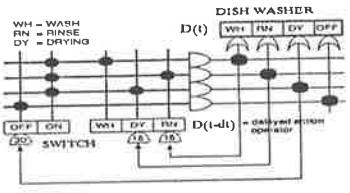


Fig. 8. A Scheduled DMPLD

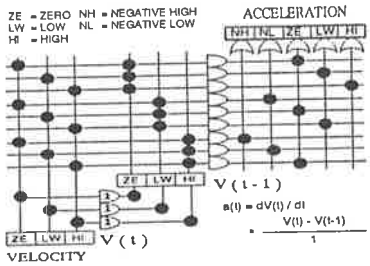


Fig. 9. A Time-Lagged DMPLD

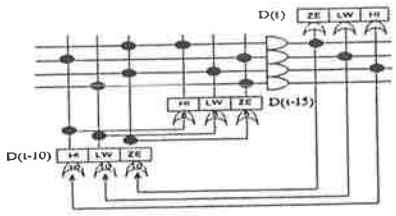


Fig. 10. An Autocorrelated DMPLD

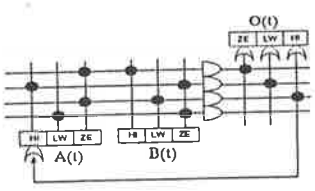


Fig. 11. An Autocorrelated DMPLD

4.2. Case Study: Intelligent Building Control System

To illustrate the GTST_DMPLD process, in this section, an intelligent building control system is discussed as an example. An intelligent building which can sense inside and outside environment, decide the efficient way of providing a convenient, comfortable and productive environment for its occupants. It can respond to occupants' need through computer-based models. A variety of modern technologies (e.g., thermal control, fire protection, computer network, security system, etc.) have been applied to the building control system to achieve important features.

To model an intelligent building control system via a DMPLD, first major system function-support relationships are decomposed and displayed. As shown in Fig. 12, the GTST_MPLD method is used to represent the complex interrelations of the building systems. For each function or system, more details can be shown. For example, in Fig. 13, "Human Comfort and Health" goal can be decomposed into eight control functions: temperature, humidity, air motion, dust, odors, acoustics, lighting and aesthetics. Support systems, for example, centeralized air cond. system and lighting system can also be decomposed into subsystems.

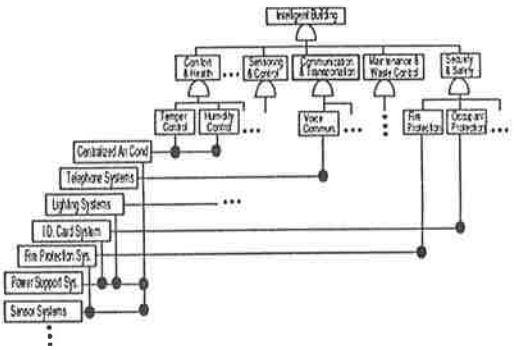


Fig. 12. A GTST_MPLD Decomposition of an Intelligent Building Control System

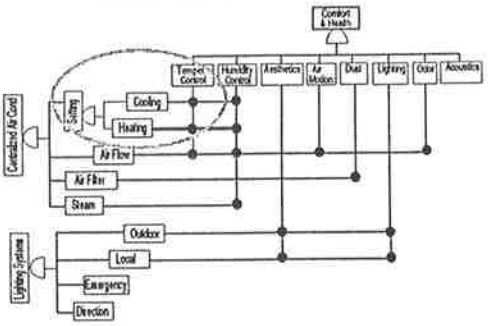


Fig. 13. The GTST_MPLD Decomposition of Subgoal "Comfort & Health"

After the whole building control system has been decomposed into small subsystems, the DMPLD is applied to model the local control processes and system dynamics. For example, Figure 14 shows an air condition system schematic diagram (Bradshaw,1985). The DMPLD of MIX for ROOM2 is shown in Fig. 15. This DMPLD models the dynamic aspects of the relationships encircled in the MPLD of Fig. 13. While occupants set a temperature, MIX will adjust the ratio of cooling and heating air to match the setting temperature. Since the

temperature of cooling and heating air depends on the temperature of outdoor air, the air mixed ratio also depends. ROOM1 may use the similar DMPLD but the fuzzy ratio of cooling air must be higher [e.g., a set (20%,50%,100%)] because of radiant heat. The computer simulation result is shown in Fig. 16. We model the room temperature to depend more on the setting temperature than the outdoor temperature.

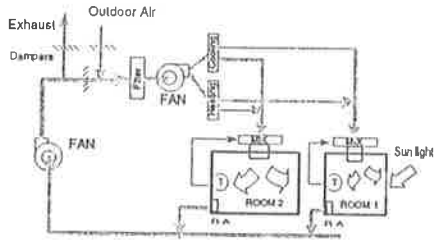


Fig. 14. An Air Condition System Schematic Diagram

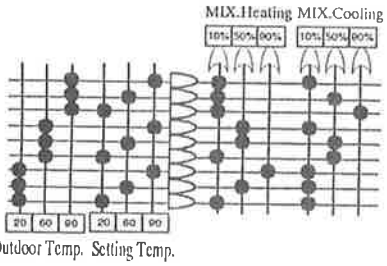


Fig. 15. The DMPLD for Fig. 14

5. REFERENCE

Alsina, C., Trillas, E., and Valverde, L. (1983). On some logical connectives for fuzzy sets theory. *J. of Mathematical Analysis and Applications*, 93:15-26.

Bellman, R., and Giertz, M. (1973). On the analytic formalism of the theory of fuzzy sets, *Inform. Sci.* 5: 149-156.

Birky, G., McAvoy, T., and Modarres, M. (1988). An expert system for distillation control design, *Computers and Chemical Engineering J.*, vol. 12. No. 9/10 1045-1063.

Brashaw, V. (1985). *Building Control Systems*, John Wiley & Sons, New York, P.P. 173-175.

Chen, L-W., and Modarres, M. (1992). A hierarchical decision process for fault administration, *Computers and Chemical Eng. J.* Vol. 16, No. 5: 425-448.

Chung, D., Modarres, M., and Hunt, R.N. (1989). GOTRES: an expert system for fault detection and analysis, *Reliability Eng. & Syst. Safety J.*, 24, 76-289.

De Kleer, J., and Brown, J.S. (1984). A qualitative physics based on confluences, *Artificial Intelligence*, 24: 7-83.

Dezfuli, H., Modarres, M., and Meyer, J. (1994). Application of REVEAL_W™ to risk-based configuration control, *Reliability Eng. and Safe. J.*.

Flekel, L. (1990). Modeling techniques for on-line expert systems in nuclear power plants, *Proceedings of the Topical Meeting on Advances in Human Factors research on Man/Computer Interactions: Nuclear and Beyond*, Nashville, TN.

Hu, Y-S., Modarres, M., and Marksberry, D. (1994). A knowledge-based system approach to reactor safety assessment, Submitted for publication to *Reliability Eng. and Sys. Safety J.*.

Hu, Y-S., and Modarres, M. (1994a). An introduction to dynamic MPLD model, *Proceeding of the PSAM-II Conference*, San Diego, CA.

Hu, Y-S., and Modarres, M. (1994b). Evaluating system behavior through dynamic MPLD modelling, Submitted for Publications to *Computers and Chemical Eng. J.*.

Hunt, R.N., and Modarres, M. (1987). Integrated economic risk management in a nuclear power plant, *1984 Annual Meeting of Soc. for Risk Analysis*.

Hunt, R.N., and Modarres, M. (1987). Performing a plant specific PRA by hand - a practical reality, resented at *the 14th INTER-RAM Conf.*, Minneapolis.

Kim, I., and Modarres, M. (1987). Goal tree-success tree model as the knowledge-base of operator advisory systems, *Topical Meeting on Artificial Intelligence and other Innovative Computer Applications in Nuclear Industry*, Snowbird, Utah.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall International Editions, Englewood Cliffs, NJ.

Modarres, M. (1992). Application of the master plant logic diagram in PSAs during design, *ANS Topical Meeting on Risk Management-Expanding Horizons*, Boston.

Pankratz, A. (1991). *Forecasting with Dynamic Regression Models*, John Wiley & Sons, New York.

Poucet, A. (1990). STARS: knowledge based tools for safety and reliability analysis, *Reliability Engineering and System Safety*, 30: 379-397.

Rasmussen, J. (1985). The role of hierarchical knowledge representation in decisionmaking and system management, *IEEE Trans. on Sys., Man, and Cybernetics*, Vol. SMC-15, No2, p.p. 234-243.

Sebo, D., Marksberry, D., and Modarres, M. (1989). RSAS: a reactor safety assessment system, *Proc. of the 7th Power Plant Dynamics, Control and Testing Symposium*, Knoxville, TN.

Simon, H. (1982). *The Sciences of the Artificial*, The MIT Press, p.p. 217-221, Cambridge, MA (1982).

Sugeno, M., and Yasukawa, T. (1993). A fuzzy-logic-based approach to qualitative modeling, *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 1: 7-31.

Zadeh, L.A. (1973). Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Trans. Syst.*, Man. Cybern., Vol. 3: 28-44.

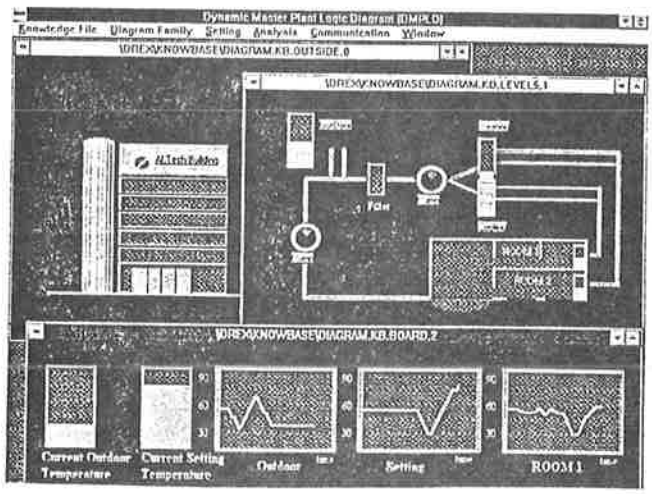


Fig. 16. The Computer Simulation Result for Fig. 15

Combining Multilevel Flow Modeling and Hybrid Phenomena Theory for efficient design of engineering systems

A. GOFUKU* and M. LIND**

** Institute of Automatic Control Systems, Technical University of Denmark, DK-2800 Lyngby, Denmark (on leave from Institute of Atomic Energy, Kyoto University, Gokasho, Uji, Kyoto, 611, Japan)*

*** Institute of Automatic Control Systems, Technical University of Denmark, DK-2800 Lyngby, Denmark*

Abstract. A combination of Multilevel Flow Modeling (MFM) and Hybrid Phenomena Theory (HPT) is proposed for efficient design of engineering systems. The MFM is a methodology to model an engineering system from the standpoint of means and goals. One useful application of the HPT is to derive mathematical equations describing a system's behavior. A conceptual design procedure composes of 1) goal decomposition, 2) listing up design constraints, 3) function assignment, 4) finding out suitable parts/components and their configuration, and 5) examination of the design's behavior. The paper presents extensions to MFM representing detailed information about goals and parts/components configuration so as to be applied to the steps 1) to 3). The HPT is applied for the step 5). Two design examples by the proposed combination are discussed.

Key Words. Computer-aided system design; Functional Modeling; Behavior calculation; MFM; HPT

1. INTRODUCTION

A design process can be described as an iterated activity comprising 'generate' and 'test' procedures. In the 'generate' part, a designer make a draft design by managing the decomposition of goals into subgoals, instantiation of design plans, invention of new parts/components, new combination of existing parts/components, modifications of similar designs, and design constraint satisfaction.

Brown, et al. (1989) categorize design activities into three classes by mainly considering the subprocesses in the 'generate' part of design. According to their classification, 'class 1 design' is defined as an open-ended creative process resulting in ill-specified goals. 'Class 2 design' is characterized by the existence of powerful problem decompositions but with the need of substantial modification. The last class, 'class 3 design', is a routine process where problem decomposition and compiled design plans are known but the design task is still too complex.

For the 'class 3 design', knowledge-based approaches such as rule-based systems or case-based reasoning can be applied. However, it is considered to be difficult to perform even 'class 2 design' by using only knowledge-based systems at current status of artificial intelligence technology. Therefore, an effective co-operation between humans and computers is necessary for the other two classes of design. That is, an effective knowledge-based design support system is needed to support the decomposition of goals into subgoals, instantiation of design plan, invention of new parts/components, new combination of existing parts/components and so on.

In this paper, a combination of Multilevel Flow Modeling (MFM) (Lind, 1990; 1992; 1993; 1994) and Hybrid Phenomena Theory (HPT) (Woods, 1991; 1993) is proposed for the efficient design ('class 2 design') of an engineering system within the process control domain. The MFM is a methodology for modelling an engineering system from the standpoint of means and goals. It has been applied to diagnostic, planning and man-machine interface problems. The purpose of the HPT is to combine qualitative reasoning and numerical calculation and for example to derive mathematical equations describing a system's behavior. In the combination of MFM and HPT, the MFM is applied to support designers in the decomposition of goals into subgoals, the representation of design constraints, the assignment of functions to goals/subgoals and representation of parts/components configuration. The HPT is applied to derive the behavior of a designed system needed in the 'test' part of design activity.

2. A KNOWLEDGE-BASED SUPPORT SYSTEM FOR FUNCTION-BASED DESIGN

In a conceptual design procedure of the 'class 2 design' of an engineering system, one can observe that designers perform the following steps to complete their design: 1) to decompose the design goals into sub-goals, 2) to list up design conditions and constraints, 3) to assign functions to achieve each decomposed goal/subgoal, 4) to identify suitable parts/components and their configuration to realize the functions, and 5) to check if the behavior of the resulting design satisfies the goals/subgoals and design conditions. In this paper, a function is defined as a useful behavior, i.e. it is describing teleology and is a selected subset of all possible behaviors, following the definition of function in the MFM. The discussions of functions are detailed in Lind (1993; 1994).

In the design procedure mentioned above, it is very important not only to decompose goals into subgoals with suitable grain size and complexity, but also to decide what functions to be assigned to each goal/subgoal. The decomposition and assignment are left to designers in ordinary design activities and depend on the quality of design. After the assignment of functions, designers can rather easily find suitable parts/components by comparing the definition of a function and the specifications of parts/components under the condition that no new part/component needs to be invented. From the standpoint of attaching importance to function assignment, the authors call this kind of design procedure a 'function-based' design procedure.

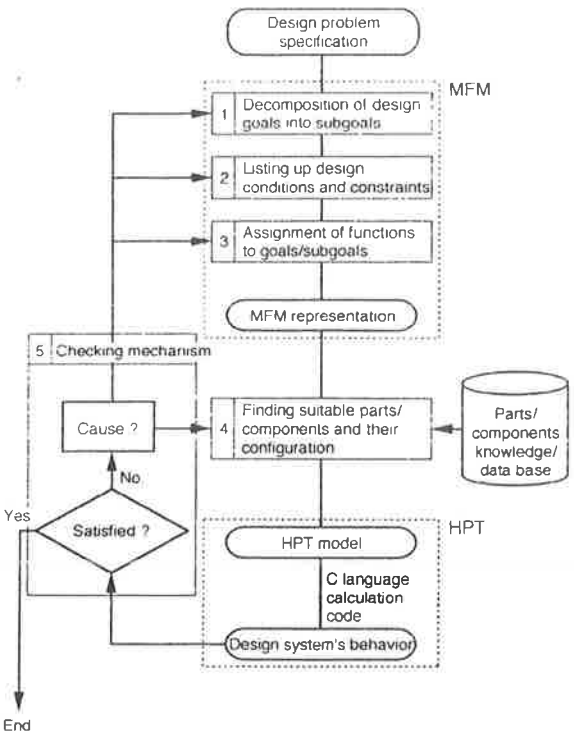


Fig. 1. Schematic of function-based design procedure

The authors are now developing a knowledge-based interface system to support designers for a conceptual design of the 'class 2 design'. The main idea of the knowledge-based support system is to combine effectively the MFM and HPT. A schematic of the function-based design procedure and the application of the MFM and HPT is shown in Fig. 1. The parts/components data/knowledge base contains the specifications, behavior knowledge, applicability knowledge, and so on of parts/components. The MFM is applied to the steps of goal decomposition and function assignment, while the HPT is used for system behavior derivation and calculation. The HPT representation in Fig. 1 is a description (set of statements) of a system according to the HPT syntax. Although there are several knowledge-based subsystems needed to complete the interface system, this paper focuses on a discussion of the application of MFM and HPT.

3. MULTILEVEL FLOW MODELING AND ITS EXTENSION FOR DESIGN PROBLEMS

The MFM (Lind, 1990; 1992; 1993; 1994) represents intentional aspects of a system from the standpoint that a system is an artifact, that is, a man-made purposeful system. The relationships among system goals, subgoals, plant functions, and control functions to achieve goals/subgoals are represented by a means-end structure. The MFM also represents a system along a whole-part dimension, that is, a system is represented by a multiple of descriptions on different levels of aggregation. System functions are represented by a set of mass, energy, activity, and information flow substructures on several levels of abstraction. Mass and energy flow substructures model plant functions, while activity and information flow substructures model operator actions and functions of the automated control systems.

A distinctive feature of the MFM is the use of a set of primitive function concepts to describe flows of material, energy, activity, and information. The flow function concepts excluding control functions and their associated symbols are shown in Fig. 2. Using these concepts, it is possible to represent knowledge of a system capturing the intentions of designers of plant and its control systems.

The current MFM model represents a system by mainly the information about functions among the information about goals, functions, behaviors, and structures, that is, 1) goal-function relationships and 2) mass, energy, and information flows of

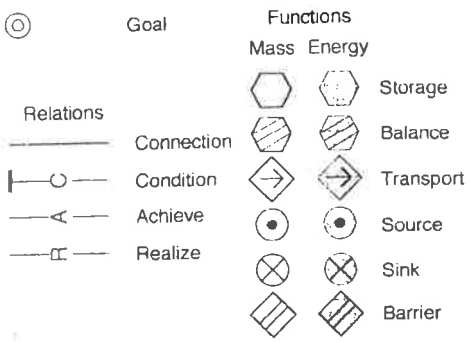


Fig. 2. Symbology of goals, functions and their relationships

functions. In order to apply the MFM to design problems, it is necessary to represent in a MFM model the relationships among goals and abstracted information about both the components that realize functions and their configuration. Of course, some of this information is already represented in a current MFM model. These are dependencies among goals represented by the hierarchy of goals. However, some conversion or interpretation is needed to obtain such information. Because the different kinds of information should be separately given to avoid confusion in the understanding of a system, the information about goals, functions, and structures must be separately presented for a designer. This necessity and restriction leads one to represent necessary information for design problems by multiple layers, that is, goal, function, and structure layers. The specifications of goals/subgoals and relationships of goals/subgoals are represented in the goal layer. The relations between goals/subgoals and functions are modeled in the function layer. In the conceptual design phase, the aim is not to precisely model the physical structure of a system. Only the recognition of the same part/component is considered to be important. Therefore, the structure layer keeps only what component realizes a function and what functions are realized by the same part/component (identity information) among other information related to the structure of a system.

What information should be represented in the goal layer and by what notation ? These problems are important in order to represent design conditions and constraints. In the ordinary design activity, there are conflicting conditions designers must manage by constraint satisfaction. In addition, it is important to plan the action sequence to solve divided subgoals (Gofuku, 1993). This is closely related with the priority of goals. Table 1 summarizes the information to be represented in the goal layer. The notation of these relations is shown in Fig. 3. The 'quantity specification' of a goal represents quantitative conditions that the functions directly connected to the goal should satisfy. A 'support condition' represents the conditional relation between two goals. The set of

Table 1 Information to be represented in goal layer

Information about goals	contents of information
1) quantitative specification	quantitative constraints the functions directly connected to a goal should satisfy
2) support relation	information about the conditional relation among goals
3) dependency relation	there are some influences among goals in the point of goal achievement
a) sympathy relation	achievement of one goal help another goal to be achieved
b) conflict relation	achievement of one goal prevent another goal to be achieved
4) design priority	information about the priority among goals in conflict relation

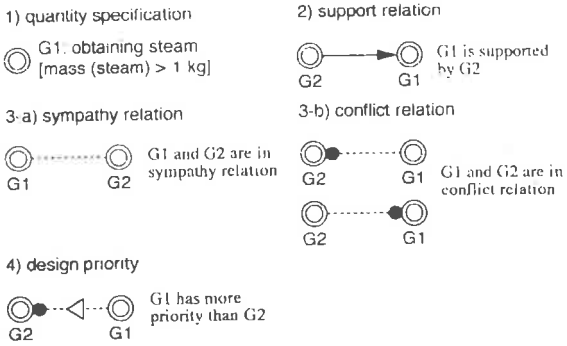


Fig. 3. Symbology of goals/subgoals relationships

support conditions forms a tree of sufficient conditions. Among the relations listed, the support relation is represented implicitly by and can be derived from the condition-achievement relations in the current MFM. When the achievement of a goal has some influences to the achievement of another goal, the two goals are in a 'dependency relation'. There are two kinds of dependency relation, that is, 'sympathy relation' and 'conflict relation' according to the sign of influences. One must specify the priority of the goals when the goals are in conflict relation.

4. IMPLEMENTATION OF THE HYBRID PHENOMENA THEORY

The HPT (Woods, 1991; 1993) integrates three types of knowledge, that is, topological, phenomenological, and behavioral aspects of a system. The HPT generates state space equations so that behavior of a system can be calculated from the symbolic descriptions of its structure and phenomenological knowledge. The idea of the HPT comes out from a symbolic framework derived from the Qualitative Process Theory (QPT) (Forbus; 1984) and the HPT extends the QPT in the direction of quantitative reasoning. The basic representational entity in the HPT is the 'influence' introduced in the QPT. Influences express how the value of a given variable is affected by one or a set of other variables when an event occurs such as boiling, or switching on an electric component in

the system. An influence gives rise to a term in a state space equation in the HPT although it specifies what can cause a quantity (the representation of a parameter for an object) to change in the QPT. There are two kinds of influences called 'algebraic influence' and 'dynamic influence'. An algebraic influence specifies how the influenced value is computed from a set of influencing variables. On the other hand, a dynamic influence specifies how a term affecting the derivative of the influenced variable is computed.

Woods (1993) mentions several applications of the HPT. One useful application of it is to be utilized as a method to derive mathematical equations describing a system's behavior by combining qualitative reasoning and numerical calculation. The present authors apply the HPT to calculate the behavior of a designed system in order to check the satisfaction of its design goals, conditions and constraints.

The present authors have implemented the HPT in Smalltalk-80 (Objectworks/Smalltalk (ParcPlace Systems, 1992)). At the present stage of the implementation, there are some restrictions for the specification of knowledge used in the HPT. In addition, the authors use a straightforward approach to implement the subsumption mechanism of HPT to ensure consistency of derived state space equations in cases where two or more objects (parts/components) are treated in one object. This is based on the consideration that it is unusual to treat several objects in one object and to make detailed equations for one object among the objects. Woods (1993) points out that the straightforward approach has significant side-effects. In spite of this it is applied in the current implementation. The authors have developed an additional part, that is, generating C language code to calculate the system behavior from the generated state-space equations.

5. EXAMPLE DESIGN PROBLEMS

This Section discusses two example design problems: (1) a hotplate system discussed by Woods (1993) and (2) a central heating system. The MFM model and the behavior calculation are discussed for the example (1). Only the MFM model is shown for the example (2).

5.1. Hotplate Example

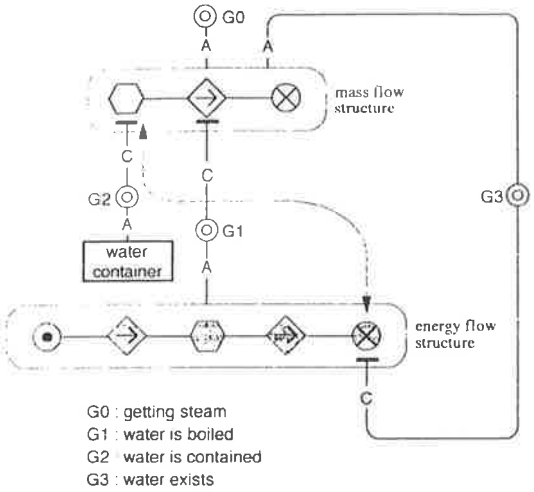
This example is 'to design a system producing steam of 0.5 kg in 20 minutes after switching on the electric system'. The reason the authors adopt the example is that the knowledge and data of the hotplate example described in Woods (1993) can be used in the HPT part of this function-based design.

A designer may specify the main goal G0 (getting steam) and represent goals and functions relationships as the mass flow structure in Fig. 4 (a). Then, the goal G1 (water is boiled) is connected as the condition of mass transfer function and this goal is achieved by an energy flow structure. G2 (water is contained) is a goal for the mass storage. Because water can not be energy sink if it does not exist, the goal G3 (water exists) is a condition of energy sink. This goal is achieved by the mass structure and therefore this chain of achievement and condition relations forms a loop in the MFM model.

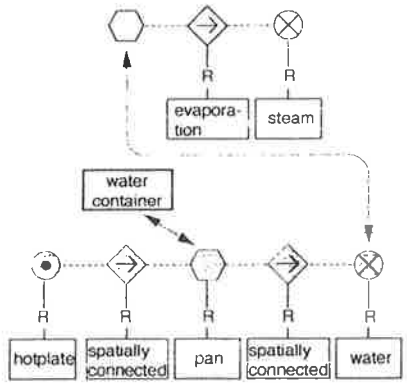
Secondly, suitable parts/components are assigned to functions, although several possible candidates corresponding to the functions are supposed. By searching the parts/components, suppose the 'water', two 'pans', and two 'hotplates' are selected. The mass capacities of the two pans are different ('pan-1': 0.45 kg, 'pan-2': 0.6 kg). The output powers of the two hotplates are different ('pan-1': 1 kW, 'pan-2': 1.5 kW). The designer will select 'pan-2' and 'hotplate-2' because 'pan-1' can not store the water of 0.5 kg and the hotplate with greater heating capacity is better in the restricted time for production of steam. The assignment results are represented in the structure layer as shown in Fig. 4 (b). During the component assignment, the designer recognizes that the mass storage and energy sink should be realized by the same part/component. The structural requirement 'water container' and the energy storage are also recognized to be realized by the same part/component. These recognitions are represented by the fine lines with arrows at both ends in Fig. 4 (b).

In parallel with the component assignment, the relationships among goals are represented in the goal layer. The results are shown in Fig. 4(c). The quantity specification of G1 is described from the viewpoint of physical dynamics. From the goals and functions relationships, one can obtain the support relations shown in the figure. The goals G1 and G2 are in a conflict relation because one can not obtain the steam of 0.5 kg in 20 minutes if the transferred energy per unit time to the water is too small. Suppose G1 has higher priority than G2 in this case. This means that the system is designed to obtain the steam as much as the system can produce in 20 minutes if the two goals are inconsistent.

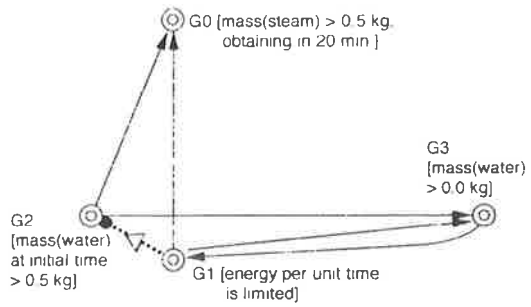
Finally, the satisfaction of all goals is checked based on the behavior of the designed system. By



(a) Function layer



(b) Structure layer



(c) Goal layer

Fig. 4. MFM model for hotplate example

using the authors' implementation of the HPT, the state space equations and their corresponding C program are generated. The calculation results of the designed system's behavior indicates that the steam of only 0.379 kg will be obtained in 20 minutes. This means that the goal G0 is not satisfied. Then, the designer abandons the satisfaction of the goal G2 by the priority setting shown in Fig. 4 (c). The designer will choose 'pan-1'. The calculated behavior of the modified system based on re-generating the state space equations indicates that the steam of 0.405 kg will be obtained in 20 minutes. The modified system is a constraint satisfaction solution for the simple design problem.

5.2. MFM Model of a Central Heating System

This example problem is 'to design a central heating system maintaining the temperature of given room at 20 °C'.

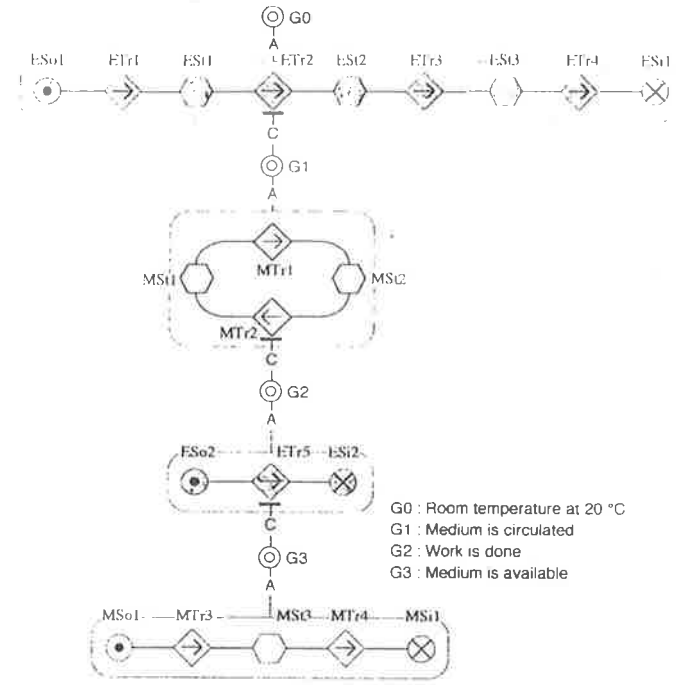
Firstly, goal/subgoals are recognized and the functions to achieve the goals are represented in the function layer of the MFM model. A designer specifies the main goal G0 (room temperature at 20 °C) and recognizes the need of energy transport to the room. Then, an energy structure to achieve the goal is represented as shown by the upper energy flow structure of Fig. 5 (a). The goal G1 (medium is circulated) is connected as the conditions of the energy transfer functions. The goal G1 is achieved by the circulation circuit and this is represented as shown by the upper mass flow structure. Because the medium (water) can not be circulated without adding force, one of the two mass transport functions should be conditioned by the goal G2 (work is done). This goal is achieved by the lower energy flow structure in the figure. The goal G3 (medium is available) is recognized to be necessary in start-up time and in an anomalous situation such as medium leakage. This goal is achieved by the bottom mass flow structure.

Then, the detailed information of each goal is given and relationships among goals are explicitly represented in the goal layer of MFM model.

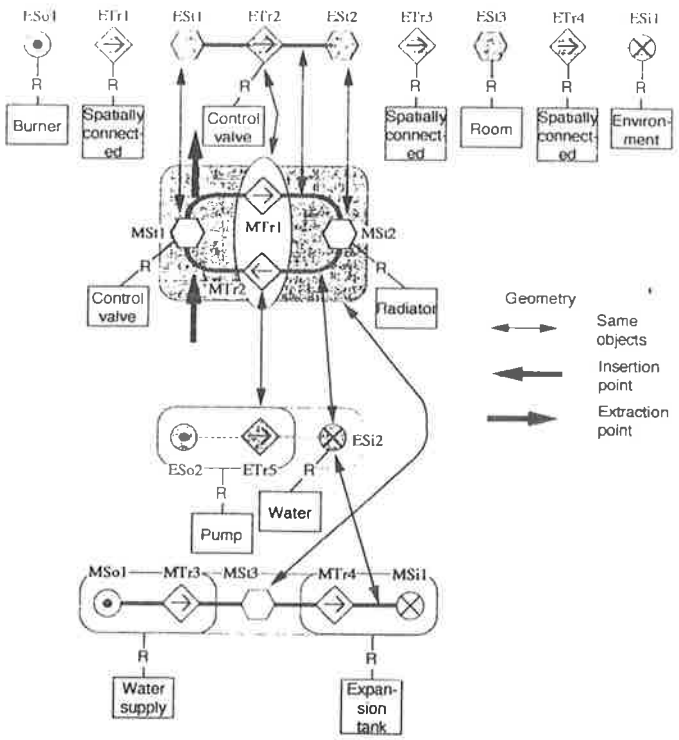
The information in the structure layer is represented as shown in Fig. 5 (b). For example, the energy storage function ESt1 and the mass storage function MSt1 is recognized to be realized by the same component. The bold lines and an arrow connecting them indicate that the designer decides to use the same medium (water) in these parts. The medium realizes the energy sink function ESi2 and this fact is indicated by the fine line with arrows connecting ESi2.

6. CONCLUSIONS

This paper proposes a combination of the MFM and the HPT to execute a 'function-based design' efficiently for the 'class 2 design' of an engineering system within the process control domain. The MFM is applied to support designers to express necessary information of a design. The multi-layers composition of a MFM model and the representation of the relationships among goals/subgoals are shown to be able to represent systematically the necessary information. The HPT is applied to calculate system behavior needed in the 'test' part of design activity. Two simple design exercises by the proposed combination of MFM and HPT are discussed.



(a) Function layer



(b) Structure layer

Fig. 5. MFM model for a central heating system

The authors are presently developing (1) a knowledge-based subsystem to indicate to a designer suitable parts/components to a function or a set of functions and (2) a graphical interface to input a multi-layers MFM model on a workstation. In addition, a subsystem to check design products is needed to be developed to complete the knowledge-based designer support system for function-based designs.

7. ACKNOWLEDGEMENT

The authors present their thanks to Dr. E. A. Woods of SINTEF for his useful discussions on their HPT application.

8. REFERENCES

Brown, D. C. and Chandrasekaran, B. (1989). Design Problem Solving - Knowledge Structures and control Strategies. *Research Notes in Artificial Intelligence,* Pitman Publishing, 32-35.

Forbus, K. D. (1984). A Qualitative Process Theory. *Artificial Intelligence*, **24**, 85-168.

Gofuku, A. and Yoshikawa, H. (1993). A Methodological Study on Organizing an Intelligent CAD/CAE System for Conceptual Design of Advanced Nuclear Reactor System. *Proc. of M&C+SNA'93 Mathematical Methods and Supercomputing in Nuclear Application*, Vol. 1, 454-465.

Lind, M. (1990). *Representing Goals and Functions of Complex Systems - An Introduction to Multilevel Flow Modeling.* Institute of Automatic Control Systems, Technical University of Denmark, Report No. 90-D-381.

Lind, M., Larsen, M. N. and Osman, A. (1992). Applications of Multilevel Flow Modeling. *Proc. of Int. Conf. on Design and Safety of Advanced Nuclear Power Plants*, Vol. IV, 42.3-1 - 3-10.

Lind, M. (1993). Functional Architectures for Systems Management and Control. in: *Interactive Planning for Integrated Supervision and Control of Complex Plant*, Final Report Project: 4937-92-08-ED ISP DK.

Lind, M. (1994). Modeling Goals and Functions of Complex Industrial Plant. to be published in *Applied Artificial Intelligence.*

ParcPlace Systems (1992). Objectworks/Smalltalk Release 4.1 User's Guide.

Woods, E. A. (1991). The Hybrid Phenomena Theory. *Proc. of IJCAI-91*, 1138-1143.

Woods, E. A. (1993). The Hybrid Phenomena Theory. *Doctorial thesis of Norwegian Institute of Technology*, NTH 1993:60.

A SIMULATION ENVIRONMENT FOR EVALUATION OF KNOWLEDGE BASED FAULT DIAGNOSIS SYSTEMS

Alenka Žnidaršič†, Victor J. Terpstra*, Henk B. Verbruggen*

†Jožef Stefan Institute
Department of Computer Automation and Control
Jamova 39, 61111 Ljubljana, Slovenia
Phone: +38 61 159 199 (int. 606); Fax: +3861 161 029
E-mail: alenka.znidarsic@ijs.si

*Delft University of Technology
Department of Electrical Engineering, Control Laboratory
P.O. Box 5031, 2600 GA Delft, The Netherlands

Abstract: In the paper a simulation environment for evaluation of knowledge based fault diagnostic systems is presented. The contribution deals with the concept of the environment that provides an integration of qualitative and quantitative models. This concept is presented on a simulated water-column process controlled by feedback. The simulation environment is used to evaluate the potentials of Multilevel Flow Modelling (MFM) approach to system diagnosis. The environment is realized within the real-time expert system tool G2.

Key words: Fault diagnosis, multilevel flow models, simulation, model-based reasoning,

1. INTRODUCTION

The main issue in designing a fault detection and diagnosis system is what kind of models are needed for a particular process in order to analyze and solve the fault detection and diagnosis (FDD) problem successfully. In addition, one has to think of the following:

- knowledge about system, process or object; it could be analytical (in terms of first principles) or empirical relating to what is obtained directly from experience,
- perception of faulty behaviour: a set of features, symptoms and patterns that are assigned to a faulty behaviour and
- decision about faults: a path from observable symptoms to a class of faults.

A plethora of models is available to solve this problem, ranging from purely qualitative to analytical (quantitative). In this paper, two different types of models have been used in order to understand diagnostic problems.

First models are used instead of a real physical system. They are integrated in the concept of the proposed "simulation environment", where they can support two different tasks: to understand the principles of process and diagnostic analysis and to model a process used in a validation phase of a diagnostic system. The latter provides a qualitative presentation of a process. A Multilevel Flow Modelling (MFM) approach (Larsson, 1992; Lind, 1988) is used to support reasoning about symptoms and faults. From that point of view it acts as a part of a diagnostic expert system.

This paper is structured as follows. In the second part the concept of simulation environment is briefly outlined. Basic ideas of MFM approach to the diagnosis are stressed in the third part. As a case study, MFM diagnostic approach is evaluated on a simulated water column process in the fourth part. Following the experience obtained from the simulation environment, some ideas for further improvement of the MFM diagnostics are presented in the fifth part. The paper ends up with concluding remarks.

2. CONCEPT OF THE SIMULATION ENVIRONMENT

Under the notion of simulation environment we mean a set of integrated tools and methodologies that serve to solve certain problem by means of simulation (Juričić et al.,1994; Krašcvec et al.,1994).

When setting the concept of the environment we tried to account for the three important concepts (Cellier, 1990; Ohren and Zeigler, 1979; Matko et.al.1992):

- concept of separation of the model from experiment which implies flexibility and simplicity,
- concept of modularity (particular units occur in the self-contained blocks that fit the input-output specification),
- concept of experimental frame (refers to the ability of the simulation environment to enable experiments on the model that highlight the relevant aspects of the problem).

In order to meet these requirements the simulation environment must incorporate:

a) modular organization,
b) object - oriented structure and
c) it must be "open", i.e. it must allow inclusion of the human factor.

For these reasons the simulation environment is implemented in G2 (GenSym Co., 1992). G2 combines three paradigms: rule - base inference, object - oriented programming and procedural programming. Because of those features, we recognized it as a tool which is appropriate for implementing the concept of simulation environment together with all the requirements demanded.

The simulation environment consists of the three different and independent modules (Fig.1):
- a simulated process module,
- an alarm definition module and
- a fault diagnosis module.

The *simulated process* module represents the process behaviour under normal working conditions in the closed loop. A model, which includes the important physical structure of the system, is used. A process is described in terms of its components and relations that exist between them.

Each process component is modelled separately as an object inside G2 with its own behaviour and attributes. The way in which a component behaves, is described by physical (analytical) equations and qualitative rules. Behaviour of the whole system is generated by the analysis of internal behaviour of each component along with relations to other components.

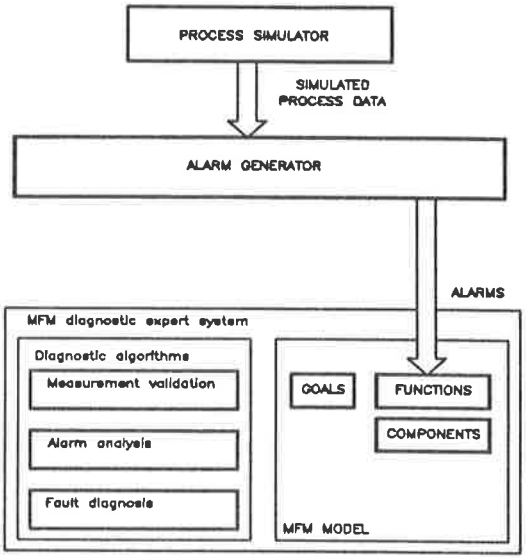


Fig.1. Realization of the MFM diagnostic ES and its place in the simulation environment

Measurements of process variables constitute basic information for diagnosis. The diagnostic methods developed on the MFM models need as an input a set of measured flow signals. But measurements do not always relate directly to the level of process representation (flow values of the MFM functions), therefore also other types of information must be used. They can be obtained with one of the following methods: additional sensors, additional estimation using data transformation (parameter or state estimation methods, statistical methods) or additional evaluation based on human observations.

Independently of a simulated process module, the *alarm definition module* has been developed. A procedure for each modelled component has been defined, which prescribes the way for obtaining the data from its observable variables. The alarm definition module performs also a detection function. Rules with "crisp" alarm limits have been used.

The simulated process has been used to test the efficiency with which the diagnostic system can recognize possible causes of malfunctioning. From this point of view it is possible to simulate different faulty behaviours of the process. All possible faults on the physical components are known beforehand. For every possible fault a procedure, which introduces this fault into the corresponding physical

component, is implemented in the *fault module*. For some faults, it is also possible to define its "magnitude". By activating the procedure the corresponding fault is injected in the simulated process.

3. THE MFM APPROACH TO FAULT DETECTION AND DIAGNOSIS

Multilevel Flow Modelling (MFM) methodology has been introduced by Lind (Lind, 1988). In the MFM approach the process is represented by means of sets of interrelated flow structures describing mass and energy at different levels of abstraction. At the highest level, general objectives of a particular process are described. At the lower levels these objectives are decomposed into more specific functions and eventually into the physical components (Lind, 1988). The representation is suitable for continuous processes whereas the topology of the mass and energy flows is described. Hence the relevant qualitative aspects of plant function in certain operational regime are encountered.

Goals, functions and components depend on each other and can be put in relations by means of the achieve (or achieve by control), condition and realize relations. Based on MFM graphs three different strategies for detection and diagnosis have been proposed (Larsson,1992): for measurement validation, alarm analysis and diagnosis.

A significant contribution to the implementation of the MFM methodology has been done by Larsson (Larsson,1992) through the design of an MFM Toolbox within G2. It consists of the two parts:

- module for developing an MFM model of the process (definition of data structures and graphic elements for building MFM graph),
- a module with a rule base that performs a diagnostic reasoning task. Several groups of rules and procedures are implemented: a rule base for syntax control of MFM models, measurement validation, alarm analysis, consequence propagation and fault diagnosis.

The Toolbox has ben developed by Larsson as part of his thesis work and was made available to the Control Laboratory at Delft University of Technology as part of a mutual research exchange. In the Toolbox the algorithms for diagnosis are independent of the process description. Therefore, the developer of the expert system needs only to construct an MFM model for his process using the Toolbox.

4. A CASE STUDY: A SIMULATED LABORATORY PLANT UNDER CONTROL

4.1. Process description

The laboratory plant consists of a large container (see Fig. 2), from which water is pumped via pipes and valves into the water column (Butler, 1990).

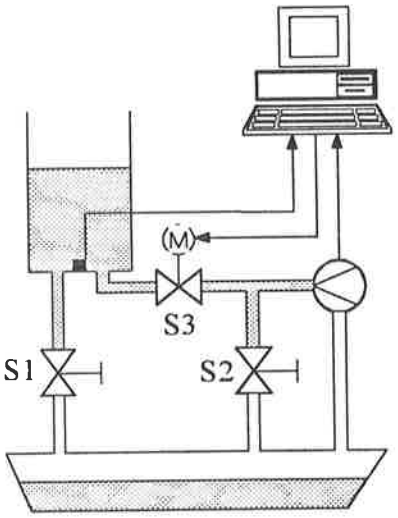


Fig.2. A water - level process

Valve S3 is control valve connected via an electropneumatic transducer to the control computer. Valve S2 (by-pass valve) is necessary to protect the pump from blocking when S3 is closed. Valve S1 determines the water outflow from the column.The water level in the column is measured using a pressure sensor at the bottom of the water column.

4.2. MFM model of the laboratory plant

Process analysis was performed using a semantic network to understand the topological and functional structure of the process. As a result, the MFM representation is obtained as shown in Fig 3.

The main goal (G1) is: *"Keep the level of the water in the column at the required position."* This main goal can be achieved by the flow network (a water flow) controlled by a manager function (M1). In this case PI controller (PF12) acting on the control valve is used.

The primary flow circuit starts at the source PF1 (water container), continues through the transport function PF2 (pressure source), balance function PF3, transport function PF4 (control valve) into the storage PF5 (water column) and through the transport function PF6 (manual valve) back to the sink PF7 (water container).

Water can be pumped from the container if the

pressure source works properly. In the MFM model, the transport function PF2 (pump the water) is conditioned (C1) by the subgoal G2: *"Keep the pump running."* The transport function is available if the subgoal is fulfilled. Electrical energy needed to run the pump is described as an energy flow from a source PF8 (power supply), via transport PF9, power switch, and to the sink PF10, motor of the pump.

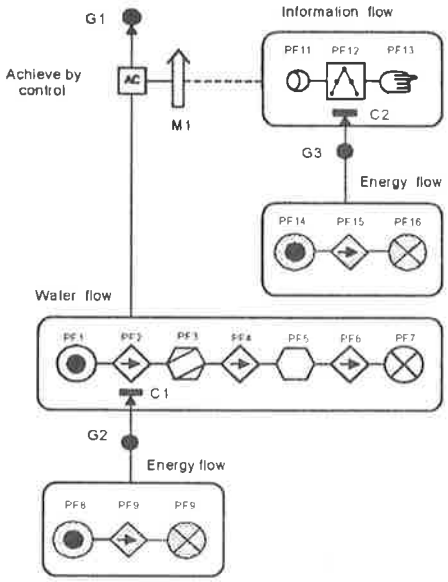


Fig.3. The MFM model of the process

The implementation of the control task (M1) is described as an information flow circuit. Measurements of laboratory process are provided using an observer function PF11 (sensor). The decision about control action is made by a PI control algorithm (PF12) and the control output is proceeded to the control valve through the actor function (PF13).

Some functions are directly related to the physical components: PF1 to the water container, PF2 to the pressure source, PF4 to the control valve, PF5 to the water column, PF6 to the manual valve and PF7 to the water container, etc. In the water flow circuit there is also an additional balance function (PF3). It is not connected to any of the physical components, but it has to be present because of syntax reasons.

The MFM model is a simplified representation of the real process. The simplification depends on the purpose of the model. We have to be aware that a diagnostic system using a simplified model can not recognize faults in the unmodeled parts of the system.

4.3. Experimental results

In order to evaluate the diagnostic system based on MFM of the plant a series of experiments is performed on the simulated process running in parallel with the MFM diagnostic expert system inside G2 (Žnidaršič, 1993).

The following assumptions have been made:

a) all possible faults of the components are previously known and modelled by means of changes in parameters of the process model,
b) deliberate sensor configurations could be applied,
c) sensors are functioning correctly and
d) the process operates in the steady state.

The first assumption is relevant for completion of the process simulator and not for the MFM based diagnosis which implicitly encounters a set of possible faults (e.g. leaks, breakdown in energy supply, etc.). Assumptions c) and d) guarantee that only component faults provoke alarms, otherwise the MFM based diagnostic module can be confused, i.e. can put in relation alarms with the component faults.

A set of experiments have been done, each encountering the following steps:

1. Define the set of measurable process quantities (levels, flows, etc.).

2. Start the process simulation by defining the reference value for the water level in the column. Wait until the process reaches the steady - state.

3. Trigger a single fault or a combination of faults in the simulated water-level process,

4. Apply the MFM diagnostic expert system, which runs in parallel with the simulated process to diagnose malfunctioning of the simulated process.

5. Analyse the diagnostic results by comparison of the diagnostic explanation with actual causes of malfunctioning.

The experimental results are shown in Table 1. Each experiment is defined by a set of injected faults and a set of process quantities available by measurements. As a result of each experiment a set of revealed faults is produced (dashed cells). Injected faults and measurable quantities are marked with "●". Tick "X" denotes those quantities the qualitative states of which could be communicated by the operator.

The obtained results could be summarized as follows. Diagnostic precision depends on the site of faults and sensor positions (measurable quantities). Generally, with increase of the measurement sites relevant to the attributes in MFM model the diagnostic precision increases. This holds particularly

in cases when measurable attributes are closely related to the failed component. If this is not the case some of the alarms have to be guessed and, as a consequence, wrong fault isolation could emerge. Multiple faults can be diagnosed when the alarm situation can not be explained with only one fault.

information. It will react too late. For example, if PF6 is decreased (due to stuck S1), feedback will force PF4 to decrease as well. MFM diagnostic module might be confused and look for causes of decreased flows both on outlet and inlet of the water column (branch with pump and control valve).

	INJECTED FAULTS							"OBSERVABLE" QUANTITIES						
	water container	pump	pump power supply	control valve	water column	manual valve	controller power supply	water container water quantity	pump pressure	pump power switch (on/off)	control valve water flow	water column water level	manual valve water flow	controller power supply
1	●											●		
2					●							●		
3											●	●		
4					●						●	●		
5	●								●	X	●	●		
6			●						●	X	●	●		
7		●			●				●		●	●		
8			●		●				●	X	●	●		
9			●						●			●		
10		●			●				●			●		
11	●							●			●	●		
12	●				●			●		X	●	●		
13	●		●					●			●	●		
14					●			●	●			●		
15							●					●	●	X
16						●						●	●	

Table 1. Experimental results

The treatment of alarms related to the flow function depends on the interval within which they are transferred to the MFM model. As soon as a symptom in the simulated process is recognised, the corresponding alarm is transferred to the MFM model. Alarm on the flow function should be assigned as a primary failed. Later on, another new alarm is discovered in the model. In this case, the fault propagation algorithm guesses also the failure state of the first one only as a consequence of this new alarm. Because the primary failed function is covered with a secondary failed one as a result of propagation rules, some information about faults can be lost. The problem is refereed to as a "loss of diagnostic discrimination". When this problem appears, the diagnosis of malfunctioning is correct but is not complete.

Furthermore, the concept of goals in the MFM syntax is questionable in case of feedback systems, e.g. in case of leak in the column the goal G1 will be maintained by feedback (the controller will force the pump to provide more water). Malfunctioning can be recognized from the control voltage changes, but the diagnostic system does not use this additional

If there were no feedback, MFM would properly identify the outlet branch as source of fault. How to encounter feedback in MFM diagnostic inference is an issue for further research.

5. DISCUSSION AND FURTHER ENHANCEMENTS

The potentials of MFM methodology for diagnostics are not fully classified yet. Anyway, MFM is an interesting process representation which could be associated to a hierarchy of objects and groups of objects convenient for vertical and horizontal diagnostic searches. Through this study some issues open to further research have been identified.

MFM representation operates with flows topology. This is but one aspect of the system behaviour which often reflects only a part of causal relationships in the system. For example, the MFM model in Fig.3 is in principle not able to account for information about pressures in the circuit. That means that correct diagnosis could be expected only in those components whose causal relationships could be described only by flows (e.g. in water

container or water column). But in case of pumps the flow alone could lead to inference only about failures (e.g. pump blocked). In case of faults (e.g. increased friction in the bearings of the pump) proper diagnosis is possible provided the pressure drop on the pump is available, what is beyond the scope of MFM.

It can be concluded that diagnostic "discrimination" of MFM is quite loose. For example, valve and pump linked in series are regarded as a module that carry out the transport function. Any degradation of this function (detected through the primary alarm) can be assigned to the module but not unambiguously neither to valve nor to the pump.

In order to increase the diagnostic discrimination additional diagnostic procedures should be applied depending on what measurement data are at disposal. That means that MFM performs only a part of top - down diagnostic search. The search on the lowest level should be done by other approaches (e.g. fault trees or parameter estimation method).

A way to increase the diagnostic efficiency (i.e. shortening of the time required to reveal a fault) in current MFM approach would be by making use of mathematical models (e.g. static models) of the process when setting alarms. In that case the goal should be formulated as "keep the process in the working point" so that the diagnostic search will be triggered whenever this goal is violated. Note that this does not influence the diagnostic discrimination.

6. CONCLUSIONS

The contribution of this paper regards the concept of simulation environment, which can be used to understand FDD problems. The main tasks that the system developer can perform using simulation environment concerns the analysis of the diagnostic on one side and evaluation of a FDD system. Therefore, the main features of such experimental environment are modular organization, object oriented structure and possibilities to include human operator.

By performing experiments on a simulated water level process within the simulation environment some properties of MFM approach to diagnosis have been evaluated. The advantage of MFM is decomposition of the process in the hierarchy of goals, functions and components. This leaves the opportunities for top - down diagnostic search. However, experiments indicate that MFM has restricted potentials in what regards diagnostic discrimination since it relies on flow topology which is just a part of causal relations in the process.

ACKNOWLEDGEMENT

The first author wishes to acknowledge financial support for this research from KFA Jülich and TEMPUS Office Brussels. Authors also acknowledge dr. Larsson from Control Laboratory at Lund Institute of Technology for providing them with software support for MFM methodology.

REFERENCES

BUTLER, H. (1990). Model reference Adaptive Control: Bridging the gap between theory and practice. Doctor's thesis: Delft University of Technology, Department of Electrical Engineering (Control Laboratory) Delft, The Netherlands

CELLIER, F.E. (1991). Continuous System Modeling. Springer - Verlag New York.

HIMMELBLAU, D.M. (1987). Fault Detection and Diagnosis in Chemical and Petrochemical Processes. Elsevier Scientific Publishing Company Amsterdam - Oxford - New York.

G2 Manual, GenSym Corporation (1992).

ISSERMANN, R. (1984). Process fault detection based on modelling and estimation methods: A survey. Automatica, 20, 387 - 404.

JURIČIĆ, Đ, E. KRAŠEVEC and M.RAK (1994). Simulation environment - a tool for analysis and design of a system for fault detection and diagnosis. IFAC Symposium SAFEPROCESS '94, EESPOO, Finland (will appear)

KRAŠEVEC, E., M.RAK and Đ. Juričić (1994). Modelling and simulation of a system for fault detection and diagnosis. Proceedings of the IMACS Symposium on Mathematical Modelling, Technical University Vienna, Austria, Vol.4, pp. 711 - 714.

LARSSON,J.E.(1992). Knowledge - Based Methods for Control Systems. Doctor's thesis: Department of Automatic Control, Lund Institute of Technology, Lund.

LIND, M.(1988). Diagnosis using Multilevel Flow models (Diagnostic strategies for P96 Demonstrator.

MATKO, D., R.KARBA and B. ZUPANČIČ (1992). Simulation and Modelling of Continuous Systems, A Case Study Approach, Prentice Hall, New York.

ÖREN T.I, B.ZEIGLER (1979). Concepts for advanced simulation methodologies, Simulation, 32 (3), pp. 69 - 82.

ŽNIDARŠIČ, A. (1993). Model-based diagnosis using MFM models for a water-level process. Working Report, Delft University of Technology, Department of Electrical Engineering (Control Laboratory) Delft, The Netherlands.

MULTI-PARADIGM REASONING FOR MOLECULAR BEAM EPITAXY CONTROL

T.V. CUDA, J.P. BAUKUS, R.L. SELIGER and D. CHOW

Hughes Research Laboratories, 3011 Malibu Canyon Road, Malibu, CA 90265, USA

Abstract. We discuss the application of AI and Knowledge Based Systems technology to process control for Molecular Beam Epitaxy (MBE). The key is the use of a multi-paradigm expert system architecture combining case-based and rule-based reasoning. This architecture addresses traditional issues in knowledge based systems technology, especially difficulties in knowledge acquisition and the brittleness of traditional rule based systems. To date this work has covered both II/VI and III/V material.

Key Words. Aritficial intelligence; expert systems; inference processes; information science; knowledge engineering.

1. GOALS

The multi-paradigm system is used to capture expertise in MBE growth design and operation. The inputs include layer compositions, thicknesses and doping profiles; i.e. the desired structure's description. The output of the system is a schedule of MBE operating parameter values appropriate to growing the desired structure. Source temperatures, substrate temperature and shutter sequences are examples of such parameters.

This accomplishes several goals. Current MBE growth systems require Ph.D.-level personnel to determine the parameters required for successful growth runs. One goal of our system is to enable less skilled personnel to determine the growth parameters necessary to achieve a desired structure; or at least allow them to operate with as little expert intervention as possible. In a similar fashion, the system allows a highly skilled expert to achieve a given structure with less trial and error. All of this cuts costs and allows experts to concentrate on further research.

Finally, the system is useful as a knowledge store. This is valuable to individual experts and at a laboratory level as a way to preserve expertise independent of the movement of personnel. One use of knowledge storing is in the university setting, where graduate students leave on a regular basis; such a system could preserve their knowledge and pass it on to the next generation bringing them quickly along the learning curve. This is of obvious importance in industry as well.

2. TECHNICAL DESCRIPTION - GENERAL APPROACH

The multi-paradigm reasoning architecture combines a particular knowledge structure and inferencing method.

Although the architecture can be used to produce a traditional backward or forward chaining rule-based system, it can also combine a set of rules with a case-based reasoner. For a general account of case based reasoning (that does not involve the

addition of rules as in our multi-paradigm system) see Kolodner *et al.* (1989) and Slade (1991). This is the main, and unique, feature of the inferencing method. Cases that represent previously solved problems, called exemplars, are stored. These exemplars consist of a set of input values and a corresponding output set. With each exemplar there is associated, through a hierarchy, a set of rules. The rules represent the theory, or domain knowledge, that is relevant to that exemplar; some rules may be shared by more than one exemplar. For a given exemplar, E, the rules associated with E, when applied to E's inputs, yield E's outputs. When a problem is presented to the system it is matched against the exemplars using the case-based reasoner. Roughly speaking, the output that is derived is that which is generated by the rules of the most closely matching exemplar. (Alternatively, one may wish to receive several outputs from the n closest matching exemplars.) Essentially, the exemplars act as a filter for which set of rules to apply to a problem.

The output for a problem case is not, in general, the same as that of any of the exemplars, but is based on the rules of the closest exemplar (or exemplars). The reason for the difference is, of course, that the input parameters of the problem case are most likely different than the input parameters of even the closest matching exemplar. Therefore, the rules of the closest matching case will derive a different output than that which is on the closest matching exemplar.

To understand the advantage of the multi-paradigm approach, one must consider a common problem with attempting to produce a standard rule base. Typically experts cannot explain precisely why they make a particular decision. They can give some rough rules of thumb, but there are often contexts under which such rough rules do not apply. Furthermore, it is difficult for experts to characterize, in any precise fashion, what all of the exceptions to the rules are. With a standard rule based system this is a serious stumbling block to knowledge acquisition. If one enters the rules without knowing the exceptions, then they fire too often; they fire in cases where an exception holds. If one tries to acquire a complete and consistent set of rules, one typically finds that the expert cannot provide them, or that it is simply too much of a burden. Issues surounding knowledge acquisition and encoding when developing an expert system with a straight rule based architecture have been well documented in the past (e.g. Aikins, 1983; Aikins, 1993; Hayes-Roth *et al.*, 1983; Stefik *et al.*, 1982; Stefik *et al.*, 1993).

With the multi-paradigm system, the expert describes a previously solved problem (its inputs and outputs) and then states the rules (domain knowledge) applicable to it. For an early account of a multi-paradigm approach see Aikins (1983). The expert never needs to explicitly describe what the exceptions to the rules are. It is only necessary to give a case description and the rules that hold for it; it is not necessary to say what it is about a given case that makes the associated rules the proper ones. Then other cases are described where other rules hold; again it is not necessary to describe why different rules hold in the different cases.

Since rules are associated with particular exemplars, they are used only in those contexts where they are appropriate; not in contexts that represent an exception to them. Because of this association with exemplars there is no need for the rules to be consistent across exemplars, whereas consistency would be demanded, and often extremely difficult to obtain, in a traditional rule-based system.

As there is no need to write individual rules that extend over an entire domain space, the multi-paradigm architecture handles areas of rapid change, or even discontinuities, in

the space well. This is because one can write rules that are entirely different, even conflicting, to cover different areas of the problem space, and then associate them with exemplars that draw a particular input problem to the appropriate rule set. This will be illustrated below.

3. TECHNICAL DESCRIPTION - SPECIFIC CASE

The structures grown consists of ten to fifteen layers of different semiconductor materials. Each layer is described by its composition, thickness and doping density, and an entire structure is grown layer by layer. The parameters that control the growth include shutter timings, cell and substrate temperatures and ramp-rates. For the multi-paradigm expert system, an input consists of the desired structure's description, that is its layer sequence.

One prima facie approach would be to store descriptions of previously grown structures as exemplars and match the desired structure against them. It was found, however, that the growth parameters for each layer depend only on its composition, thickness and doping density and on the composition of its adjacent layers and the substrate used; the rest of the structure is irrelevant to growing a particular layer. So, to determine the growth parameters for each layer, and hence for the whole structure, we need only match each layer and its adjacent layers to previously grown layers and their adjacent layers; we need not match entire structures.

The exemplars therefore consist of triples of layers from previously grown structures instead of whole structures, and we match triples of layers from the desired structure against these exemplars. The exemplar triples contain descriptions of the center layer, for which the growth parameters have been determined, and the layers below and above it. A triple from the desired structure consists of descriptions for the center layer, for which one is trying to derive the growth parameters, and the layers below and above it. If the entire desired structure consists of n layers, then there are n triples from which one derives the growth parameters for the n layers. (In the description of a triple, the layers below or above a given layer may include the substrate, or if we are at the end of the structure, the empty layer.)

Matching on triples, instead of entire structures, has the advantage that one previously grown structure provides many exemplars, and one can grow structures unlike any previously grown ones, and still make use of past experience. This is because although a given entire structure may not look anything like a previously grown one, each of its triples may look much like some triple in some previously grown structure. This allows a relatively few previously grown structures to cover a wide range of the case space.

A set of rules is specified and stored with each exemplar. These rules are the ones that the expert deems relevant for determining the growth parameters for the exemplar's center layer. When a problem case, i.e. a structure for which one is attempting to derive the growth parameters, has one of its triples match closely to an exemplar, the rules associated with that exemplar are applied in order to determine the growth parameters necessary for growing the desired triple's center layer.

Some rules may be associated with more than one exemplar. In fact, there are a few rules that apply to all exemplars. But one of the main features of this approach is that most rules are associated with just one exemplar and need not be applicable to a wide range of cases; they only need to be applicable to cases that are similar to the exemplar with which they are associated.

This exemplar-rule structure allows one to use relatively simple rules. One never has to struggle to determine complicated

general rules that form a complete and consistent rule set. General rules are included only if they are simple to acquire and code. Often, generality is sacrificed for simplicity in the rules and the case matching acts as a filter so that limited rules of thumb are not utilized in domains that fall beyond their scope.

Here is an example to illustrate. The following is one of the exemplars, call it E1, that we have stored in the current system.

InGaAs, undoped 42Å
AlAs, undoped 24Å
InGaAs, undoped 15Å

Now assume that we wish to grow the following structure, call it S1.

InGaAs, n+ 2000Å
InGaAs, n 500Å
InGaAs, undoped 17Å
AlAs, undoped 30Å
InGaAs, undoped 45Å
AlAs, undoped 30Å
InGaAs, undoped 17Å
InGaAs, n 500Å
InGaAs, n+ 1μm
InP, substrate

Layer 4 (counting up from, but not including, the substrate) in S1 is AlAs and its relevant triple is

InGaAs, undoped 45Å
AlAs, undoped 30Å
InGaAs, undoped 17Å

We see that this triple matches closely to E1. Hence the system will use the rules from E1 to set the growth parameters for layer 4 in S1. For example, the following rule, rule-1, is associated with E1, as well as some other exemplars, for setting the Al cell shutter time.

If
    Desired Substrate is InP,
Then
    Al_Cell_Shutter_Time =
    [(Stored_Al_Cell_Shut_Time *
    Desired_Layer_Thickness)/
    Stored_Layer_Thickness]

Where Stored_Al_Cell_Shut_Time and Stored_Layer_Thickness are variables that take on the values of the Al cell shutter time and stored layer thickness for E1. This same rule is associated with more than one exemplar and these variables take on different values when used with different exemplars.

A similar rule, using an appropriate constant, derives the Al cell shutter timing for a different substrate.

Rule-1 is very simple, and it is only applicable in cases very similar to E1. We can utilize such a simple rule only because the case matcher filters when it is used. For example, when we attempt to grow layer 7 from S1, it will not match well against E1, but rather some other exemplar. Thus the rules from this other exemplar, and not rule-1, will be used.

Without the multi-paradigm approach, that is in a straight rule based system, rules such as rule-1 would be disastrous. Rule-1 does not even contain any mention in its antecedent that we are attempting to grow a layer with an Al component. Consider what would happen in a straight rule based system if rule-1 were present. When we went to grow layer 7, rule 1 would fire and open the Al shutter even though no Al is desired in layer 7.

Without the multi-paradigm approach, we would need to encode rules that were much more complicated than rule-1. For example a rule such as rule-1* below.

If
    Substrate is InP
    Desired Composition
        Contains Al
    Desired Thickness is $T \pm \Delta T$
    Desired Doping Density is D
    Al Cell Temperature is $C \pm \Delta C$
    As Cell Temperature is $C' \pm \Delta C'$
Then
    Al Shutter Time = F()

Where after the equality sign would come some function of the parameters in the antecedent of the rule.

There are several features to notice about rule-1*. First, it contains much more information in its antecedent than rule-1, and some of the quantities in its antecedent are not input parameters. For example, Al cell temperature is not an input parameter, it is a growth parameter that we are trying to derive. Since some of the parameters in rule-1* are not input parameters we are forced to derive their values from yet other rules. That is, in a straight rule based system we would need to have considerable chaining amongst the rules.

One way to look at it is that the cases contain a great deal of the expertise in the system, and this expertise needs only be filled out by a few simple rules for purposes of extrapolating to similar cases. Once we match to an exemplar a great deal is already determined. For example, once we matched to E1, it was already determined what the Al cell temperature would be, it would be the same as that used when E1 was grown, and we did not need to consider it in the rule to derive the Al cell shutter timing. Hence we did not need to mention Al cell temperature in the rule, nor did we need to chain to another rule to derive Al cell temperature.

We use the Al cell temperature of E1 because in cases that are similar to E1, we know, from our MBE experts, that we should scale them by adjusting shutter timings and not temperatures. For exemplars that should be scaled by changing a cell temperature, we utilize other rules. But the point is, once the system matches to an exemplar, a great deal is determined. The rules at that point do not need to do too much work. For example they do not need to determine whether to scale by temperature, or shutter timings, or both, and then derive all of the relevant values by chaining together.

A normal rule base must cover the entire problem space, and this can be extremely difficult if discontinuities are present. With the multi-paradigm approach, discontinuities are taken care of by utilizing numerous rule bases, that contain differing rules representing the differences from one part of the problem space to another. The issue of which rule base to use is then decided by the case matcher. Although it is difficult to measure, we have certainly found that in many domains implementing an expert system is far easier with the multi-paradigm architecture than with a straight rule base.

This multi-paradigm approach greatly enhances, and essentially automates knowledge acquisition. This is because much of the knowledge is stored in the exemplars, and the exemplars can be automatically acquired from any standard

computer file. (Even if they must be typed in, it is still a simple task that does not require an expert.)

Once the exemplars are acquired, one must "manually" acquire the necessary rules from the experts, but this is much system. It is much easier to show an expert a stored exemplar and ask how to scale it, than it is to acquire the complete rule set that would be necessary in a traditional rule based system.
The expert need not give general, complete, and consistent rules. Just rules of thumb for scaling the particular exemplar in question. We do not need, for example, complicated rules defining when to scale by temperature, when by shutter timings and when by both.

4. CONCLUSION

Whether or not the multi-paradigm approach is superior to a traditional rule base depends on both the availability of data with which to form exemplars and the complexity of the rules, if one were to write a traditional rule base. Essentially, the better one can cover the problem space with exemplars, the less work the rules have to do. So, if exemplars are easy to come by, as they are in our MBE domain, and the rules would be complicated in a traditional rule base, then it is best to use the multi-paradigm approach. However, if exemplars are difficult to come by, and/or the rules for a traditional rule base would be simple, then a straight rule based system may be best.

The multi-paradigm architecture outlined above is meant to achieve ease of knowledge acquisition and encoding while producing a system that can free expert time and act as a knowledge store. It depends on the development of a general reasoning approach that has many uses. In the area of process control we have thus far focused on MBE, but the multi-paradigm architecture promises to be useful in other control domains as well.

This work was funded in part by ARPA under Agreement Number MDA972-93-H-0005.

5. REFERENCES

Aikins, J.S. (1983). Prototypical knowledge for expert systems. *Artificial Intelligence*, **20**, 163-210.

Aikins, J.S. (1993). Prototypical knowledge for expert systems: a retrospective analysis. *Artificial Intelligence*, **59**, 207-211.

Hayes-Roth, F., D.A. Waterman, and D.B. Lenat, eds. (1983). *Building Expert Systems*, Wesley, Reading, MA.

Kolodner, J.L. and R.L. Simpson (1989). The MEDIATOR: Analysis of an early case-based problem solver. *Cognitive Science*, **13**, 507-549.

Slade, S. (1991). Case-based Reasoning: A research paradigm. *AI Magazine*, **12**, 43-55.

Stefik, M., J.S. Aikins, R. Balzer, J. Benoit, L. Birnbaum, F. Hayes-Roth, and E.D. Sacerdoti (1982). The organization of expert systems, a tutorial. *Artificial Intelligence*, **18**, 135-173.

Stefik, M., J.S. Aikins, R. Balzer, J. Benoit, L. Birnbaum, F. Hayes-Roth, and E.D. Sacerdoti (1993). Retrospective on "The organization of expert systems, a tutorial". *Artificial Intelligence*, **59**, 221-224.

CANCELLATION CONTROLLER BASED ON FUZZY RELATIONAL MATRIX AND COMPARISON WITH OTHER CONTROL ALGORITHMS

Igor Škrjanc, Drago Matko
Faculty of Electrical and Computer Engineering
Tržaška 25, 61000 Ljubljana, Slovenia

Abstract. In the paper the cancellation controller based on the inverse fuzzy relational model is presented. The inverse fuzzy relational model is obtained using the input error model. The proposed algorithm is implemented on nonlinear process which has varying time constant and the process gain according to the operating point. The comparison between the cancellation controller based on inverse process fuzzy relational model and robust classical algorithms shows some advantages of the first algorithm. The described algorithm can be a useful tool in the case the nonlinear system control.

Keywords. fuzzy control, identification

1. INTRODUCTION

The cancellation controller based on the inverse fuzzy relational model represents one of approaches where the inverse dynamics model of the process plant is required. The inverse dynamics model approach to the control is very well known in the area of the neural networks systems.

After introduction in the first section, in the second section the identification algorithm is presented. The model of the process is represented by means of the fuzzy relational matrix with fuzzified input variables. In the first step of this identification algorithm the input-output data of dynamical system are collected. In the next step the fuzzy variables should be defined for all input and all output variables of the observed dynamical system. All input-output data have to be fuzzified, because the algorithm deals with fuzzy values. The most important part of algorithm is the definition of the fuzzy model structure. The fuzzy relational matrix $\boldsymbol{R}$ of the observed process is calculated on the basis of the fuzzified data. To obtain the optimal relational matrix an optimisation algorithm has to be used.

In the third section the cancellation controller based on fuzzy relational matrix model is described. Its application to a nonlinear plant and comparison with robust PI and GPC algorithms is presented in the fourth section. The nonlinearity of the plant exhibits as variable process gain and variable time constants.

2. THE FUZZY RELATIONAL MATRIX MODEL

The cancellation controller in our case is based on the inverse fuzzy relation model. The fuzzy relational model could be

obtained using the identification algorithm (4) which represents a slightly different approach from that proposed by E.Czogala and W.Pedricz (3). The key element of this method is based on simplified defuzzification which could be in the case of two input and one output model expressed as

$$y = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\mu_{ij}r_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m}\mu_{ij}}, \qquad (1)$$

where μ_{ij} represents the element of matrix $\boldsymbol{S}$ which is composed of input variables fuzzy sets as follows

$$\boldsymbol{S} = \boldsymbol{\mu}_{u1}^{T} \circ \boldsymbol{\mu}_{u2} \qquad (2)$$

where $\boldsymbol{\mu}_{u1}$ and $\boldsymbol{\mu}_{u2}$ stand for fuzzy sets of input variables $u1$ and $u2$, respectively and $\circ$ stands for compositional operator which can be a simple scalar product or *min* operator.

The dimension of $\boldsymbol{S}$ matrix which represents actually the structure of the model depends from the dimension of input fuzzy sets $\boldsymbol{\mu}_{u1}$ and $\boldsymbol{\mu}_{u2}$. If the the dimension $1 \times m$ for vector $\boldsymbol{\mu}_{u1}$ and dimension $1 \times n$ for vector $\boldsymbol{\mu}_{u2}$ are assumed then the dimension $m \times n$ for the fuzzy data matrix $\boldsymbol{S}$ is obtained. The fuzzy relational matrix $\boldsymbol{R}$ which represents the fuzzy relational matrix model of the process consists of elements r_{ij} described in equation 1 and where $i = 1, \cdots, m$ and $j = 1, \cdots, n$. The both matrix can be expressed as follows

$$\boldsymbol{S} = \begin{bmatrix} \mu_{11} & \mu_{12} & \cdots & \mu_{1n} \\ \mu_{21} & \mu_{22} & \cdots & \mu_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{m1} & \mu_{m2} & \cdots & \mu_{mn} \end{bmatrix}, \qquad (3)$$

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix}. \qquad (4)$$

The matrix $\boldsymbol{R}$ represents the transformation from the input fuzzy domain to the crisp output domain. The fuzzy relational matrix can be considered as a mathematical model of the observed process.

The vector $\boldsymbol{r}$ and $\boldsymbol{s}$ are formed of the matrix $\boldsymbol{R}$ and $\boldsymbol{S}$ elements, respectively and can be described as follows

$$\boldsymbol{r} = (r_{11}r_{12}\cdots r_{1n}\cdots r_{m1}r_{m2}\cdots r_{mn})^{T} \qquad (5)$$

and

$$\boldsymbol{s} = (\mu_{11}\mu_{12}\cdots\mu_{1n}\cdots\mu_{m1}\mu_{m2}\cdots\mu_{mn})^{T}. \qquad (6)$$

The equation 1 can be according to the proposed notation described in the following vector form

$$y = \frac{\boldsymbol{s}^{T}\cdot\boldsymbol{r}}{\boldsymbol{s}^{T}\cdot\boldsymbol{I}} \qquad (7)$$

where $\cdot$ stands for scalar product and $\boldsymbol{I}$ defines a vector of ones with the same dimension $(n \cdot m \times 1)$ as $\boldsymbol{s}$

$$\boldsymbol{I} = (1, 1, 1, \cdots, 1)^{T}. \qquad (8)$$

The equation 7 has to be reshaped in the form

$$\boldsymbol{s}^{T}\cdot\boldsymbol{r} = \boldsymbol{s}^{T}\cdot\boldsymbol{I}\, y. \qquad (9)$$

Calculating the fuzzy data vector $\boldsymbol{s}$ for each time instant t, the equation 9 could be described as an equation with dynamical variable

$$\boldsymbol{s}^{T}(t)\cdot\boldsymbol{r} = \boldsymbol{s}^{T}(t)\cdot\boldsymbol{I}\, y(t). \qquad (10)$$

If the measurment are made in the equidistant time instants $t = t_1, t = t_2, \cdots, t = t_k, \cdots, t = t_N$ the linear matrix equation can be formed in the following way

$$\begin{bmatrix} \boldsymbol{s}^{T}(t_1) \\ \vdots \\ \boldsymbol{s}^{T}(t_k) \\ \vdots \\ \boldsymbol{s}^{T}(t_N) \end{bmatrix} \cdot \boldsymbol{r} = \begin{bmatrix} \boldsymbol{s}^{T}(t_1)\cdot\boldsymbol{I}\, y(t_1) \\ \vdots \\ \boldsymbol{s}^{T}(t_k)\cdot\boldsymbol{I}\, y(t_k) \\ \vdots \\ \boldsymbol{s}^{T}(t_N)\cdot\boldsymbol{I}\, y(t_N) \end{bmatrix}. \qquad (11)$$

The equation 11 can be also described in the matrix form

$$\boldsymbol{\Psi}\cdot\boldsymbol{r} = \boldsymbol{\Upsilon}. \qquad (12)$$

The solution of the overdeterminated sys-

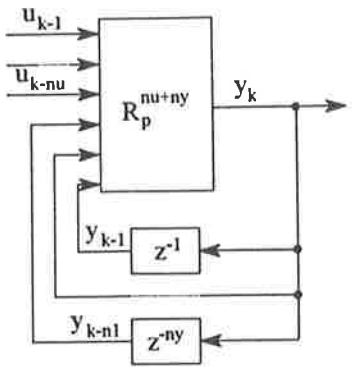


Fig. 1. The general form of the fuzzy relational model

tem described in equation 12 could be obtained as an optimal solution of vector $\boldsymbol{r}$ using the least square method by minimizing the criterion function in the least-squares sence.

The optimal estimated value of vector $\boldsymbol{r}$ can be expressed in the equation

$$\boldsymbol{r} = (\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\boldsymbol{\Upsilon} \tag{13}$$

where $\boldsymbol{\Psi}$ stands for fuzzified data matrix with dimension $N \times (n \cdot m)$ where N represents the number of time instants, and $\boldsymbol{\Upsilon}$ denotes the data matrix with dimension $N \times 1$.

In the case of more sofisticated dynamics the proper choice of input variables has to be made according to the quality of the model. The choice of the structure of the fuzzy relational matrix model is the most important and difficult part of identification algorithm and has to be made interactively using the simulation.

In general the complex dynamics can be modelled using the structure of the model which is presented on Fig. 1. The complex dynamics require higher number of input variables which results nomore in fuzzy relational matrix. In this case the model can be described in fuzzy relational tensor for three input variables and the fuzzy relational $\boldsymbol{R}_p^n$ form model in general. The model can be represented as a matrix in n dimensional space.

3. CANCELLATION CONTROLLER BASED ON FUZZY RELATIONAL MATRIX MODEL

The cancellation controller is ussually designed for servo- or model- following control. The main idea of this approach is to design a controller which prescribes the desired behaviour of the closed-loop system given by the model transfer function $G_m(s)$. The closed-loop transfer function should be equal to the desired model transfer function $G_m(s)$

$$G_m(z) = \frac{Y(z)}{W(z)} = \frac{G_r(z)G_p(z)}{1 + G_r(z)G_p(z)} \tag{14}$$

where $G_p(z)$ and $G_r(z)$ are process and controller transfer functions, respectively.

The controller transfer function follows immediatly from equation 14

$$G_r(z) = \frac{1}{G_p(z)}\frac{G_m(z)}{1 - G_m(z)}. \tag{15}$$

The cancellation controller based on fuzzy relational model could be presented in the following form

$$\frac{Up(z)}{E(z)} = \frac{G_m(z)}{1 - G_m(z)}, \tag{16}$$

$$u_k = \frac{\boldsymbol{s}^T \cdot \mathbf{r}_c}{\boldsymbol{s}^T \cdot \boldsymbol{I}} \tag{17}$$

where the second equation represents the cancellation part which is given by the vector $\boldsymbol{s}$ and the fuzzy relational vector $\boldsymbol{r}_c$ which is given by the matrix $\boldsymbol{R}_c$.

The close-loop system with the cancellation controller based on fuzzy relational matrix is presented on Fig. 2. The proposed algorithm has been tested on the simple nonlinear process where the gain and the time constant of the process vary with the operating point.

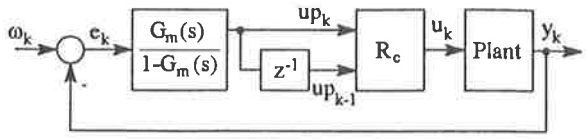


Fig. 2. The closed-loop system with cancellation controller based on fuzzy relational matrix

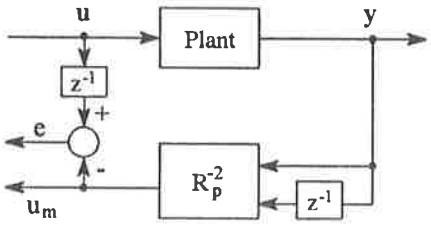


Fig. 3. The input error model

4. FUZZY CANCELLATION CONTROL OF NONLINEAR PLANT

The nonlinear process which has been chosen can be described with difference equation

$$y_{k+1} = (0.8930 + 0.0371 y_k) y_k + \frac{u_k}{30}. \quad (18)$$

The fuzzy controller cancellation matrix $\mathbf{R}_c$ should be defined as the inverse fuzzy model of the process. The inverse fuzzy model is obtained using input error model as shown on Fig. 3. The fuzzy relational matrix inverse is obtained by the identification algorithm as previously shown. In the case of modelling the inverse process of the first order the inputs into the identification algorithm are $y(k)$ and $y(k-1)$ and the output is $u(k-1)$.

The obtained inverse process fuzzy relational model of the nonlinear system expressed in equation 18 can be described in the following form

$$\boldsymbol{R}_p^{-2} = \begin{bmatrix} -10.87 & -58.76 & -0.0017 \\ 52.34 & -0.0009 & -54.38 \\ 0.0006 & 60.80 & 1.96 \end{bmatrix}. \quad (19)$$

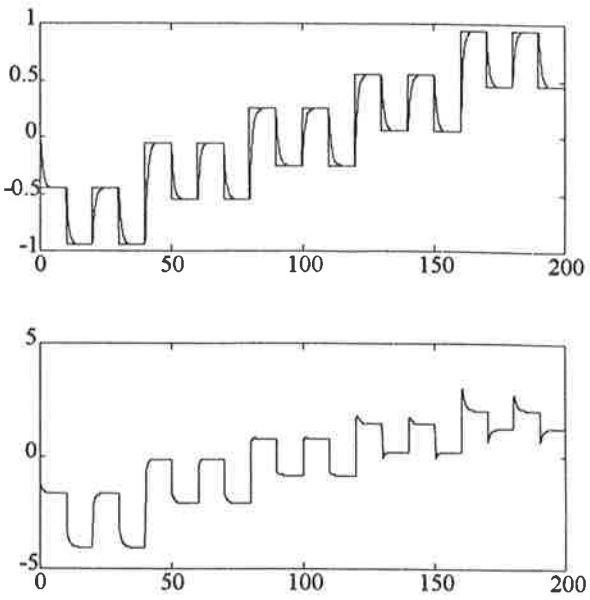


Fig. 4. The reference and output signal and control signal in the case of the fuzzy cancellation control

The fuzzy inverse relational model of the process is equal to the cancellation relational matrix $\boldsymbol{R}_c = \boldsymbol{R}_p^{-2}$ and the prescribed model transfer function is equal $G_m(z) = \frac{0.0908}{z-0.9092}$. The results of control are shown on Fig. 4. The output signal dynamics is equal to the prescribed dynamics in the whole operating area. The method is compered with robust PI and with GPC algorithm. Both algorithms are designed for the operating area around zero. The results obtained using the discrete robust PI controller with transfer function $G_{PI}(z) = \frac{0.45z-0.31}{z-1}$ are presented on Fig. 5. In the case of GPC algorithm with parameters $T_1 = 0, T_2 = 10, N_u = 2$ and $\lambda = 0$ the results are shown on Fig. 6. The transfer function of the GPC controller is $G_{GPC}(z) = \frac{4.59z+4.14}{z-1}$. It can be seen that the results obtained using the proposed cancellation fuzzy algorithm exhibit better performance in comparison to the classical robust algorithms. Proposed algorithm could be seen as an useful tool in the case of nonlinear dynamics of first or second order.

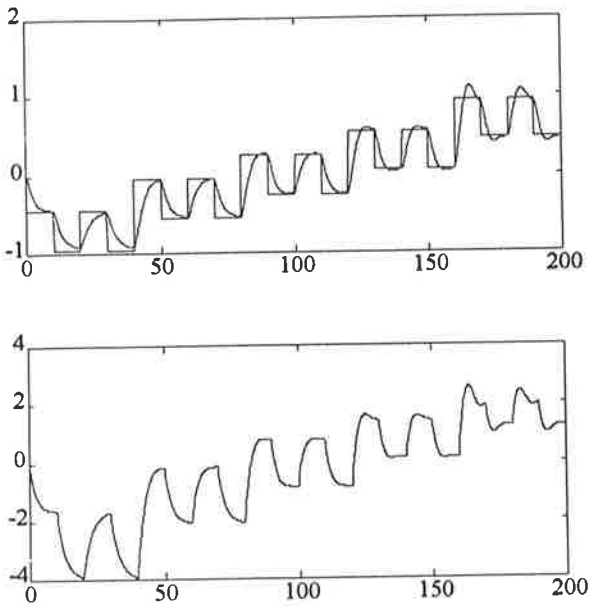


Fig. 5. The reference and output signal and control signal in the case of robust PI controller

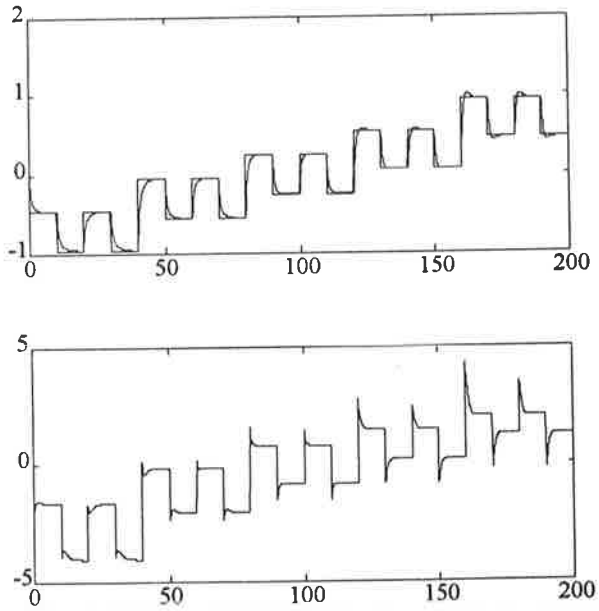


Fig. 6. The reference and output signal and control signal in the case of GPC controller

5. CONCLUSION

In the paper the cancellation controller based on the inverse process fuzzy model is presented. Using the identification algorithm based on the fuzzy relational matrix the nature of nonlinear process can be satisfactorily described. The results obtained using the cancellation controller based on inverse process fuzzy relational model show some features in the case of nonlinear systems of first or second order. The main disadvantage of proposed algorithm is complicated structure of the model and time consuming identification in the case of higher order process dynamics.

REFERENCES

[1]R.M.Tong, Analyses of fuzzy control algorithms using the relation matrix, Int. J. Man-machine studies, vol. 8, Page 679-687, 1976

[2]E.Czogala, W.Pedrycz, On identification in fuzzy systems and its applications in control problems, Fuzzy sets and systems, North Holland Publishing Company, Vol.6, No.1, Page 73-83, 1981

[3]W.Pedricz, An identification algorithm in fuzzy relational systems, Fuzzy sets and systems, North Holland Publishing Company, Vol.15, Page 153-167, 1984

[4]B.M.Pfeiffer, Identifikation von Fuzzy-Regeln aus Lerndaten, Beitrag zum 3. Workshop "Fuzzy-Control" des VDE/GMA-Unterausschusses, Dortmund, 1993

[5]W.T.Miller, R.S.Sutton, P.J.Werbos, Neural Networks for Control, The MIT Press, 1991

[6]B.Müller, J.Reinhardt, Neural Networks, Springer-Verlag, 1991

[7]K.Warwick, G.W.Irwin, K.J.Hunt, Neural networks for control systems, IEEE Control Engineering Series 46, Peter Peregrinus Ltd., 1992

[8]R.Isermann, K.H.Lachmann, D.Matko, Adaptive Control Systems, Prentice Hall, 1992

COMPARISON OF INTERPOLATIONS IN FUZZY CONTROL

B. KOVALERCHUK, H. YUSUPOV, and N. KOVALERCHUK

Uzinform, Tashkent WED University
8 Druzhby Narodov St. Tashkent, 700043, Uzbekistan, CIS

Abstract. An alternative interpolation is offered for fuzzy control. It decreases: (i) runtime, (ii) complicated defuzzification, (iii) tuning of antecedent and consequent membership functions by "blind" optimization and (iv) necessity of ill-founded operations such as min and max for connectives. The deviation between Mamdani and offered (Second) Interpolation is no more than 5,05% of the fuzzy set support for linear interpolation (non symmetrical two-dimensional case). For symmetrical two-dimensional case both linear and non linear second interpolations give the same deviations no more than 5,05%. For one-dimensional case they give deviation no more than 2,5% of the support. For non-normalized membership functions the deviation from linear interpolation is about 13-17% of the support.

Key Words. Fuzzy control, interpolation, error analysis; linearization techniques; controllers

1. INTRODUCTION

There are a lot of problems in fuzzy control as general design methodology, heuristic tuning rules, training data, where, when and how to use this technique (Horacek *et al.*, 1993; Arzen *et al.*, 1993; Klawonn, 1993).

The Second Interpolation approach (Kovalerchuk *et al.*, 1993; Galichet *et al.*, 1993; Raymond *et al.*, 1993; Meyer-Gramann, 1993) is developed in the paper in contrast with Mamdani, Sugeno and some others (First Interpolation) to improve fuzzy control.

The purpose of the First Interpolation is to find fuzzy control system φ as interpolation of a real input/output function f. Here, the input is a state space and/or variations of its components. The output is a control set. The purpose of the Second Interpolation is to find piecewise function λ as interpolation of fuzzy control system φ (Fig.1).

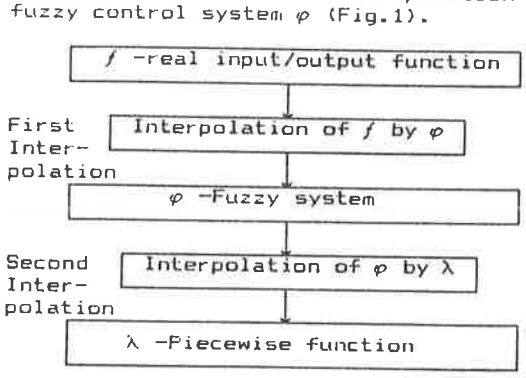


Fig.1. Correspondences of inrerpolations

Let the First Interpolation φ be sufficiently close to the real function f and the Second Interpolation λ gives the same precision of φ interpolation. Then the Second Interpolation λ can be used instead of the real function φ in control.

2. NOTATIONS AND DEFINITIONS

Due to different terms in the field below several variants from (Horacek *et al*, 1993; Klawonn, 1993; Galichet *et al.* 1993, Meyer-Gramann, 1993) and others are given. $X=\{x\}$ is an input. Usually X is a state space and/or variations of its variables.
$U=\{u\}$ is a control set (output, controller's output), i.e. U is the set of control or action variables.
$f(x)=u$ ($\varphi(x)=u$; $\lambda(x)=u$) is an input/output map (I/O map, I/O function).
Example for x. $x=(\varepsilon,\Delta\varepsilon)$ is an input value.
ε is an error. $E=\{\varepsilon\}$ is an universe to the variable *error*.
$\Delta\varepsilon$ is a *variation of the error* (*change of error*).
$\Delta E=\{\Delta e\}$ is an universe to the variable *change of error*. $\{A_i\}$ is the set of linguistic terms (marks) for errors E, usually as NL (Negative Large), NM (Negative Medium), NS (Negative Small), ZR (Approximately Zero), PS (Positive Small), PM (Positive Medium), PL (positive Large).
$\{B_j\}$ is a set of linguistic terms (marks) for errors ΔE.
$\mu_i(\varepsilon)$; $\mu_j(\Delta\varepsilon)$ are input membership functions of fuzzy sets for linguistic terms A_i, B_j for inputs ε and $\Delta\varepsilon$.
a_i is the peak value (modal point) of the membership function $\mu_i(\varepsilon)$.

Normalized Intersection (overlapping=1) of membership functions μ_i and μ_{i+1}, takes place if $\mu_i(a_{i+1})=0$, $\mu_{i+1}(a_i)=0$ and $\mu_i(x)+\mu_{i+1}(x)=1$ (see fig.2).

b_j is the peak value (modal point) of the membership function $\mu_j(\Delta\varepsilon)$;

$u_{ij}=u_k$ is the peak value (modal point) of the output membership function $\mu_k(u_{ij})$

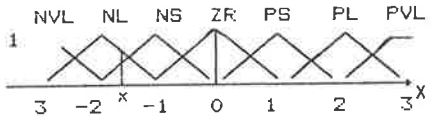


Fig.2. Normalized intersected membership functions (overlapping =1)

$\varphi(x)=u$ is a Fuzzy Rule Based Controller or Fuzzy Controller (Horacek *et al.*,1993). Fuzzy controller includes:

1) Fuzzification as a conversion crisp input(measurement) -> fuzzy input
2) Rule evaluation as a conversion fuzzy input -> fuzzy output
3) Defuzzification as a conversion fuzzy output -> crisp output (controller action).

COG is the Center of Gravity (procedure for defuzzification).

COA is the Center of Area (procedure for defuzzification).

$\{U_k\}$ is the set of linguistic terms for variable *controller output*.

R^j is the Linguistic Rule:
IF ε is A_i AND $\Delta\varepsilon_j$ is B_j THEN u_{ij} is U_k

The Control Algorithm is a set of linguistic rules $R^1, R^2, \ldots R^k$ of the form R^j connected by ELSE.

(a_i, b_j, u_{ij}) is a peak tuning point (known point, prototype).

$PM=\{(a_i, b_j, u_{ij})\}$ is a Rule Matrix, in which all the correlations with the fuzzy sets are implicitly defined. The rows (E), columns (ΔE) and entries (U) of RM represent the positions in the universes that correspond to the peak values of the membership functions.

$\varphi(x)$ is the First Interpolation of $f(x)$ I/O map.

$\varphi(x)=\varphi(x, PM, COG, \{\mu_i(\varepsilon)\}, \{\mu_j(\Delta\varepsilon)\}, \{\mu_k(u)\})$ is Mamdani fuzzy controller

$\lambda(x)$ is the Second Interpolation of I/O map $f(x)$.

$\lambda(x)=\lambda(x, PM)$. $\{\mu_i(\varepsilon)\}$, $\{\mu_j(\Delta\varepsilon)\}$, $\{\mu_k(u)\})$ and the COG are not the arguments of λ. But $\{\mu_i(\varepsilon)\}, \{\mu_j(\Delta\varepsilon)\}, \{\mu_k(u)\})$ must meet requirements of normalized intersection (overlapping is equal to 1) for best interpolation as it is shown below.

3. BASE FOR FIRST INTERPOLATION.

The first interpolation has empirical and mathematical justifications, which are given below.

Empirical arguments. If people control a plant effectively for a long term they often can formulate their strategy in the form of fuzzy linguistic rules. This idea is supported by a lot of successful applications of fuzzy control.

Mathematical arguments. For any AND-operations, OR-operations, defuzzification procedure, and basic membership functions with a compact support, the resulting fuzzy controls are universal approximators. It means that they are capable of approximating any real continuous control function on a compact set to arbitrary accuracy (Nguyen *et al.*,1993). But some of these results are not constructive.

Bauer *et al.* (1993) and Lee *et al.*,1993) offered some constructive algorithms for the for one-dimensional case. But in multi-dimensional case for some class of functions "the possibilities for fine tuning are quite limited" (Bauer *et al.* 1993). So a lot of "blind" empirical tunings were offered.

4. TUNING PROCEDURES AND TRAINING DATA

4.1. "Blind" Empirical Tuning

Dynamic behavior of control system depends on the shape of membership functions. The system can present a limit cycle, a quasi-periodicity or a chaos. Tuning of control system by optimization technique must preserve these features and does not generate meaningless linguistic terms.

The "blind" optimization of interpolation includes searching ideas as Neural Nets and Genetic Algorithms if they are used without sufficient modification for fuzzy control.

Disadvantages of "blind" tuning of fuzzy control systems for First Interpolation were considered by Pfeiffer *et al.* (1993) and Pedrych *et al.* (1993). "The optimization simultaneous of both I/O interfaces and linguistic subsystem with no integrity constraints can generate meaningless linguistic terms" (Pedrych *et al.*,1993, p.1190). "It is not efficient to apply any "blind" numerical optimization procedure not using linguistic information to design a fuzzy control system instead of directly optimizing the conventional control characteristic" (Pfeiffer *et al.*, 1993, p.1407).

The Second Interpolation allows to improve a tuning procedure for the First Interpolation or to substitute it.

4.2. Tuning Procedure

Different tuning ideas have been offered to reduce the number of rules and to create new ones in synthesizing a fuzzy system. They concern I/O interface, membership functions and linguistic system. In particular an idea of Fuzzy Self-Organizing Controllers is being developed by Gutierres *et al.* (1993).

The Second Interpolation gives a way to tune fuzzy control systems less heuristically. It concerns the number of membership functions, their shape and characteristic points.

4.3. Training Data

The collection of valid training data can be difficult. At the moment the number of clusters presented in each input and in each output must be determined a priori. An extension would be to incorporate a hierarchical clustering algorithm which added clusters as needed, thus automatically generating the optimum number of membership functions for each input and output.

The Second Interpolation approach allows to develop this idea. Peak points of membership functions are natural training data. The optimum number of training data can be found starting from normalized set of membership functions due to their efficiency for precise interpolation.

5. SECOND INTERPOLATION

The Second Interpolation is based on the selection of:
(i) membership functions:
(ii) peak tuning points (prototypes):
(iii) piece-wise linear interpolation between prototypes:
(iv) precision estimations of the interpolation.

This Interpolation is made without such "indirect" elements of the First Interpolation as complicated defuzzification and tuning of both antecedent and consequent membership functions on the base of "blind" optimization approaches.

The "blind" approaches can be used after the second interpolation for further optimization. In particular it can be made for control systems under constraints. For example control system with a piecewice linear I/O function and linear constraints can be too complex for linear programming method.

Strict linear interpolation is given by Meyer-Gramann (1993) for the controller with practically the same membership functions as used by Kovalerchuk *et al.* (1993), Galichet *et al.* (1993) and Raymond *at al.* (1993), but for other t-norm and t-conorm.

"Algebraic product" is used instead of the minimum (t-norm) and "bounded sum" instead of the maximum (t-conorm) in Mamdani controller. Below just Mamdani control systems are considered.

Galichet *et al.* (1993) and Pfeiffer *et al.* (1993) represented conditions of strict linear interpolation for Mamdani controller. Main of them are output's not overlapping, rectangular membership functions and normalized triangular input's membership functions.

Here due to not overlapping there are a lot of intermediate points x with

$$\mu_{output}(f(x))=0.$$

The last value is a natural index of reliability of the interpolation for given x. Hence the interpolation is unreliable for such kind of intermediate points. So below more reliable interpolations with overlapping output membership functions are considered

6. GENERAL STATEMENT

Below some heuristic, common sense suppositions of fuzzy control are explicated to improve it. Improvement of fuzzy control systems can be made through:
1) runtime of calculations;
2) precision of calculations;
3) time and procedure for tuning of the system;
4) empirical justification of the system.

For these purpose the general statement (G) is formulated:
(G) General statement. Mamdani Fuzzy control is a kind of quasi-linear interpolation of known points (prototypes) of input/output map. Mamdani input/output map (function) $u=\varphi(x)$ can be sufficiently exact represented by piecewise-linear function of x. This result was shown (Kovalerchuk *et al.*, 1993) for one-dimensional case. Last one means that the premise of "if-then" rules is a single linguistic value. Here inputs and outputs are triangular membership functions with overlapping =1. See fig.2.

The maximum distance between outputs of the linear and Mamdani controllers is proportional to the distance between the nearest peak points of membership functions of output (Galichet *et al*, 1993). Numerical estimations of this distance are given in (Kovalerchuk et al.,1993).

The distance is no more than 2.5% of the fuzzy sets support for one-dimensional case and as we show below it is no more than 5.05% for symmetrical two-dimensional case. Below a modification G1 of General Statement G is formulated on the base of Raymond's *et al.* (1993) paper.

G1-General statement 1. Mamdani fuzzy control is a quasi-square interpolation of known points (prototypes) of input/output map for two-dimensional case. Here Mamdani function $u=\varphi(x)$ can be sufficiently exact represented by a set of non-linear functions of x:

$$u=\varphi(x)=\varphi(x_1,x_2)=k_1x+k_2x+k_3x_1x_2+k_4$$

This $\varphi(x)$ is called a double linear function. The two-dimensional case means that premises of "if-then" rules are AND-connections of two linguistic values. The distance is 5.05% of the support for symmetrical case and 12,5% o support for non-symmetrical case for nonlinear interpolation The result holds for the same as in (Kovalerchuk *et al.*,1993) inputs and outputs (triangular membership functions with overlapping =1).

7. ONE-DIMENSIONAL CASE

Below three representations of linguistic terms (NL,NM,NS,ZR,PS,PM,PL) are considered:

(R1) non-intersected intervals. Linguistic terms are represented by usual intervals without fuzzy sets.

(R2) partly-intersected fuzzy sets

(R3) mormalized-intersected fuzzy sets (overlapping =1). Maximum of μ of one fuzzy set is a minimum of the next. See fig.2.

Rough piecewise interpolation is obtained for (R1) for intermediate points. In the

case (R2) the interpolation is more precise. In the case (R3) the interpolation for intermediate points is the best. The deviations from Mamdani scheme are no more than 2,5% of the of support of considered fuzzy sets. It proves the statement on quasi-linear interpolation for each nearest peak points. So an acceptable piecewise linear function $v=\lambda(\alpha)$ exists.

8. IMPROVEMENT OF FUZZY CONTROL RUNTIME

Decreasing of runtime is an important problem. According to Tilli (1993) applying fuzzy control often requires a fast execution (<1ms) of the inference algorithm and high precision (>10 bit). The standard hardware cannot execute the standard Mamdani MAX-MIN inference with the COG (Center of Gravity) defuzzification quick enough (Tilli,1993). Special hardware solutions typical have resolutions in the range 4 to 6 Bit.
An alternative inference and defuzzification methods have made a progress in these problems on standard processors (Tilli ,1993). But these results are based just on faster realization of t-norms t-conorms and usual defuzzification methods as COG, COA without interpolation.

The second interpolation approach decreases runtime significantly in comparison with Tilli(1993) results because of simple piecewise interpolation can be computed faster.

9. PRECISION OF PIECEWISE LINEAR INTERPOLATION FOR SYMMETRICAL CASE

9.1. Linear I/O Functions for Triangles

An input plane is divided into triangles. For each of them linear interpolation is offered. Then they are compared with Mamdani I/O function and non linear interpolation of Raymond *et a.* (1993). In last one the input plane is divided into squares. It is one of the reason of differences in precision of two versions. Triangulation allows to improve precision of interpolation.

Let us given output values

$$u_{10} \geq u_{00} \geq u_{11} \geq \geq u_{01}$$

for input points (0,0); (0,1); (1,0); (1,1), then linear interpolation for (x_1,x_2) from the triangle with vertices

$$<(0,0);(1,0);(1,1>$$

will be

$$(u_{10}-u_{00})x_1+(u_{11}-u_{10})x_2+u_{00}=u \qquad (1)$$

(Fig.3) and for (x_1,x_2) from the triangle with vertices $<(0,0);(0,1);(1,1>$ will be

$$(u_{11}-u_{01})x_1+(u_{01}-u_{00})x_2+u_{00}=u \qquad (2)$$

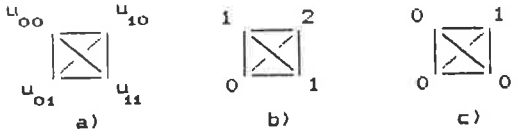


Fig.3. Elementary mesh for interpolation

9.2. Non Linear I/O Functions for Squares

Non-linear interpolation (double linear interpolation) is considered for the points: (0,0); (0,1); (1,0) ; (1,1) and $u_{01} \leq u_{00}=u_{11} \leq u_{10}$

$$(1-x_1)(1-x_2)u_{00}+x_1(1-x_2)u_{10}+$$
$$+(1-x_1)x_2u_{01}+x_1x_2u_{00}=u \qquad (3)$$

Below (3) and accompanying formulas are given in original terms of Raymond *et al.* (1993) for better comparison:

$$(1-\beta_j)(1-\alpha_i)u_{ij}+\alpha_i(1-\beta_j)u_{i+1j}+$$
$$+(1-\alpha_i)\beta u_{ij+1}+\alpha_j\beta_i u_{ij}=u \qquad (4)$$

So there is just one formula for all square(0,0);(0,1);(1,0);(1,1) not dividing them as in (1) and (2) for triangles. As Last dividing of the square allows to preserve and increase the precision of interpolation.

9.3. Mamdani I/O Function

Let us consider a formula for min-max method for elementary mesh (square) for symmetrical case $u_{ij}=u_{i+1j+1}$ (Raymond *et al*, 1993, p.154)

$$u_{MinMax}=\varphi(\varepsilon,\Delta\varepsilon)=C/D, \qquad (5)$$

where

$$C=max[(min(\alpha_i,\beta_j);min(\alpha_{i+1},\beta_{j+1})]u_{ij}+$$
$$+min(\alpha_{i+1},\beta_j)u_{i+1j}+min(\alpha_i,\beta_{j+1})u_{ij+1} \qquad (6)$$

$$D=max[(min(\alpha_i,\beta_j);min(\alpha_{i+1},\beta_{j+1})]+$$
$$+min(\alpha_{i+1},\beta_j)+min(\alpha_i,\beta_{j+1}) \qquad (7)$$

$$\alpha_i=\mu_i(\Delta\varepsilon);\ \beta_j=\mu_j(\varepsilon);$$
$$\alpha_{i+1}=\mu_{i+1}(\Delta\varepsilon);\ \beta_{j+1}=\mu_{j+1}(\varepsilon),$$

here ε is the error and $\Delta\varepsilon$ is the variation of the error $\Delta\varepsilon$, $\mu_i(\Delta\varepsilon);\mu_{i+1}(\Delta\varepsilon);\mu_j(\varepsilon);\mu_{j+1}(\varepsilon)$ are input membership functions of fuzzy sets for linguistic terms A_i,A_{i+1},B_j,B_{j+1} for inputs ε and $\Delta\varepsilon$. Let us consider for normalized triangular membership functions

$$\alpha_{i+1}=1-\alpha_i,\ \beta_{j+1}=1-\beta_j \qquad (8)$$

and estimate u_{MinMax} for antidiagonal, i.e. for

$$\alpha_i+\beta_j=1;\quad \alpha_{i+1}+\beta_{j+1}=1 \qquad (9)$$

Due to (9)

$$\beta_j=1-\alpha_i \text{ and } \beta_{j+1}=1-\alpha_{i+1}=\alpha_i \qquad (10)$$

9.4. Transformation

Using (10) formulas (6),(7) can be rewriten

$$C=max[(min(\alpha_i,1-\alpha_i);min(1-\alpha_i,\alpha_i)]u_{ij}+$$
$$+min(1-\alpha_i,1-\alpha_i)u_{i+1j}+min(\alpha_i,\alpha_i)u_{ij+1} \qquad (11)$$

and

$$min(\alpha_i,1-\alpha_i)u_{ij}+(1-\alpha_i)u_{i+1j}+\alpha_iu_{ij+1} \qquad (12)$$
$$B=min(\alpha_i,1-\alpha_i)+(1-\alpha_i)+\alpha_i=min(\alpha_i,1-\alpha_i)+1$$

Let $\alpha_i \geq 1-\alpha_i$ then

$$C=(1-\alpha_i)u_{ij}+(1-\alpha_i)u_{i+1j}+\alpha_iu_{ij+1}=$$
$$=(1-\alpha_i)(u_{ij}+u_{i+1j})+\alpha_iu_{ij+1} \qquad (13)$$

$$D=2(1-\alpha_i)+\alpha_i=2-\alpha_i \qquad (14)$$

$$u_{MinMax}=C/D=[(1-\alpha_i)(u_{ij}+u_{i+1j})+$$

$$+ \alpha_i u_{ij+1}]/(2-\alpha_i) \qquad (15)$$

Let i=j=0 and

$$u_{ij}=u_{i+1j+1}=1;\ u_{i+1j}=2;u_{ij+1}=0, \qquad (16)$$

See Fig.3 b). Let also $0.5 \le \alpha_i \le 1$ then

$$u_{minmax}=(1-\alpha_i)(1+2)=3(1-\alpha_i)/(2-\alpha_i) \qquad (17)$$

The formula (17) gives values from (16) for vertices of the square as arguments.

9.5. Precision Estimation

For the triangle $<(0,0);(1,0);(1,1)>$ linear interpolation is

$$u_{lin}=(u_{10}-u_{00})x_1+(u_{11}-u_{10})x_2+u_{00} \qquad (18)$$

For antidiagonal ($x_1=-x_2+1$) and condition (16) here

$$u_{lin}=-x_1+x_2+1 \qquad (19)$$

$$u_{lin}=2(1-x_1) \qquad (20)$$

Let us estimate $\min(u_{lin}-u_{MinMax})$ for $0.5 \le \alpha_i < 1$

$$\min(u_{lin}-u_{MinMax})=$$
$$=\min[2(1-\alpha_i)-3(1-\alpha_i)/(2-\alpha_i)]=$$
$$=\min(\alpha_i-3\alpha_i+1)/(2-\alpha_i) \qquad (21)$$

This formula is absolutely the same as in (Raymond *et al*, 1993,p.153). So the minimum is the same -0.101 (see table 1). In the terms of fuzzy sets support this value is equivalent to 5,05% of the support, due to the distance between considered nearest fuzzy sets is a half of support. By the very way the same estimation can be made for other part of antidiagonal.

So it is shown that for symmetrical case the linear interpolation can be used inside of the triangle with errors no more than 0,101, i.e. about 5% of the support of considered fuzzy sets.

Table 1 Comparison of Interpolations

α_i	$u_{lin}-u_{MinMax}$	u_{lin}	u_{MinMax}
0.5	0.0	1.0	1.0
0.6	-0.0571	0.8	0.8571
0.77	-0,1009	0.46	0.5609
0.775	-0.1010	0.45	0.5510
0.776	-0.1010	0,448	0.5490
0.777	-0.1010	0.446	0.5470
0.8	-0.1	0.4	0.50
0.9	-0.07	0.2	0.27
1	0.0	0.0	0.0

9.6. Interpolation for Symmetrical Case

Let us show that linear interpolations (1) and (2) for symmetrical case are the same plane under the condition (16). Let us show it for points out of corresponding triangles. For $u_{01}=(0,1)$ according to (1) and taking into account that $u_{00}=u_{11}=1$; $u_{10}=2$ (see (16))

$$u_{01}=u_{11}-u_{10}+u_{00}=0 \qquad (22)$$

For $u_{10}=(1,0)$ according (1) and $u_{00}=u_{11}=1$; $u_{00}=0$ (see condition (16))

$$u_{10}=u_{11}-u_{01}+u_{00}=2 \qquad (23)$$

So it is sufficient for symmetrical case to consider just one of the formulas (1),(2). It meant, that one linear interpolationexists for simple symmetrical case for the square (Fig.3b) instead of non-linear interpolation of Raymond *et al.* (1993) with the same precision as in (3). Our interpolation can be simpler computed.

If $u_{ij}=q_0+q_i i+q_j j$ for all i,j. where q_0,q_i,q_j are constants. Then it will be linear interpolation instead of piecewise linear interpolation. The proof is analogous.

Let $u_{ij}=u_{i+1j+1}$; $u_{ij+1}-u_{i+1j+1}=u_{i+1j}-u_{ij}$,

there exist two linear functions: a mesh function T_{ij} for each elementary mesh transformation

$$T_{ij}(u_{ij})=t_o+t_1 u_{00}$$
$$T_{ij}(u_{i+1j})=t_o+t_1 u_{10}$$
$$T_{ij}(u_{ij+1})=t_o+t_1 u_{01}$$
$$T_{ij}(u_{i+1j+1})=t_o+t_1 u_{11}$$

and an inverse function T_{ij}^{-1} with coefficients t_i' and t_1'.

If these coefficients are not equal for different elementary meshes it is sufficient to keep coefficients instead of membership functions parameters and to use (1) as common linear interpolation for all elementary meshes. See fig.4.

$$(x_1,x_2) \rightarrow T_{ij} \longrightarrow (1) \rightarrow T_{ij}^{-1} \longrightarrow u$$

input — mesh transformation — values interpolation (1) — linear inverse

Fig.4. Transformation

10. PRECISION OF PIECEWISE LINEAR INTERPOLATION FOR NON-SYMMETRICAL CASE

For non-symmetrical case two simple linear interpolations are more precise than square (double linear) interpolation. Let us show it for the same case as in (Raymond *et al.*, 1993,pp.154-155) for non-symmetrical case with $u_{00}=u_{01}=u_{11}<u_{10}$. It is clear, that for the part of antidiagonal with $\alpha_i \le 0.5$ linear interpolation (2) gives zero-error. For semiantidiagonal with $\alpha_i \ge 0.5$ it will be the same case as was considered above for (1) in the chapter 9.5 with the error -0.101. So two linear interpolations decrease error to zero in one half of antidiagonal and safe it for the other its part in comparison with non linear interpolation. As a result two linear interpolations are better for this non-symmetrical case with $u_{00}-u_{01}<u_{10}-u_{11}$ and $u_{00}=u_{11}$ (see Fig.3c). Their maximal error is 0,101 instead of 0.25 for one non-linear function (Raymond *et al.*,1993).

This result can be generalized for more general non-symmetrical case when all $u_{01},u_{00},u_{11},u_{10}$ are unequal and for example, $u_{01}<u_{00}<u_{11}<u_{10}$ and the similar result can be obtained for other defuzzification schemes and fuzzy operations &,∨.

11. CONCLUSION

The Second Interpolation λ substitutes the real input/output function f with practically the same precision as Mamdani controller φ for wide range of one- and two- dimensional cases.
The deviation λ from φ is no more than 5,05% of the fuzzy set's support for linear interpolation (non symmetrical two-dimensional case) in contrast of 12,5% for non-linear interpolation (double linear interpolation).

Both linear and non-linear variants (symmetrical two- dimensional case) give the same deviations no more than 5,05%. Also for one-dimensional case they both give deviation no more than 2,5% of the sup port. See table 2.

Table 2. Second Interpolation maximal deviations (%)

Second Interpolations for normalized μ				
	Symmetrical		Non-symmetrical	
	linear	non-linear	linear	non-linear
1-dim.	2.5	2.5	-	-
2-dim.	5.05	5.05	5.05	12,5

It is shown that the estimation of deviation for non-normalized membership functions from linear interpolation is about 13-17% of the support.

These Second Interpolations eliminate such elements of the First Interpolation as complicated defuzzification (Center of Gravity), tuning of both antecedent and consequent membership functions by "blind" optimization and ill-founded operations as min and max for connectives.
As a result the Second Interpolation is simpler than conventional Mamdani controller. Due to this simplification the Second Interpolation decreases a runtime.
Among the different Second Interpolations the linear interpolation based on triangles of input plane is more precise than double linear interpolation on squares of input plane. In addition the linear interpolation is more logically founded and can be simpler computed.
Second Interpolation is more founded than First Interpolation (Mamdani Controller). An interpolation is considered as more founded if it uses less suppositions. Suppositions of Mamdani controller include: Center of Gravity (COG) for defuzzification, min-max for connectives, prototypes, membership functions.

Suppositions of Second linear interpolation include: piecewise linearity, prototypes, normalized triangular membership functions. Strictly talking, Mamdani controllers don't require normalized triangular membership functions, but really, in majority of real application it holds. Non-linearity of Mamdani output for normalized triangular membership functions is a result of min-max operations and COG. Other connectives (different t-norms and t-conorms) and defuzzification are able to increase or decrease non-linearity. For other hand peace-wise linearity of Second Interpolation also is just postulated. But this supposition is the simplest and it can be realized by adding of new prototypes.

12. REFERENCES

Arzen K-E. and K.J. Astrom (1993). Expert systems in engineering: state of the Art. In: *First European Congress on Fuzzy and Intelligent Technologies*, Proceedings, (H.-J. Zimmermann, Ed.), Vol.1, pp.308-313, Verlag der Augustinus Buchhandlung, Aachen.

Bauer P.,E.P.Klement,A.Leikermoser and B. Moser (1993).Approximation of real functions by rule bases. In: *Fifth IFSA World Congress*,Proc.,Vol.1,pp. 239-241, Seoul.

Galichet S.,and L.Foulloy (1993). Fuzzy equivalence of classical controllers. In: *First European Congr.on Fuzzy and Intelligent Technologies*, Proc.Vol.3,pp. 1567-1573, Aachen.

Grant B.,and C.de Bruijn (1993).Generation of fuzzy control rules and membership functions using neural nets. In: *First European Congress on Fuzzy and Intelligent Technologies*, Proc., Vol.2, pp.651-656,Aachen.

Gutierres R.,and R.Tanscheit (1993). Performance of a fuzzy self-organizing controller regarding scaling factors, linguistic values and quantization level of the variables. In:*First European Congress on Fuzzy and Intelligent Technologies*, Proc.,Vol.2, pp.745-750,Aachen.

Horacek, P., and Z.Binder (1993). An approach for design and implementation of fuzzy controllers. In: *First European Congress on Fuzzy and Intelligent Technologies*, Proceedings, (H.-J. Zimmermann, Ed.),Vol.1, pp.163-169. Verlag der Augustinus Buchhandlung, Aachen.

Klawonn F.(1993). Mamdani's model in the view of equality relations In: *First European Congress on Fuzzy and Intelligent Technologies*, Proc. ,Vol.1,pp.364-369, Aachen.

Kovalerchuk B.,and H.Yusupov (1993).Fuzzy control as interpolation. In:*Fifth IFSA World Congress*, Proc., Seoul,Vol.2, pp. 1151-1154.

Lee J.,and S.Chae (1993).Completeness of fuzzy controller carrying a mapping f. In: *Proc. FUZZ-IEEE'93*, pp.231-235.

Meyer-Gramann K.D.(1993).Easy implementation of fuzzy controller with a smooth control surface.In:*First European Congress on Fuzzy and Intelligent Technologies*, Proc.,Vol.1, pp.117-123,Aachen.

Nguyen H.T.,and V.Kreinovich (1993). On approximation of controls by fuzzy systems.In:*Fifth IFSA World Congress*, Proc.,Vol.2,1414-1447,Seoul.

Pedrycz W., and J.Valente de Oliveira (1993).Optimization of fuzzy relational models. In: *Fifth IFSA World Congress*, Proc.,Vol.2.pp.1187-1190,Seoul.

Pfeiffer B-M.,and R.Isermann (1993). Criteria for successful application of fuzzy control. In: *First European Congress on Fuzzy and Intelligent Technologies*, Proc.,Vol.3, pp.1403-1409,Aachen

Raymond C.,S.Boverie and J.M.Le Quellec (1993). Practical realization of fuzzy controllers comparison with conventional methods. In: *First European Congress on Fuzzy and Intelligent Technologies*, Proc.,Vol.1,pp.149-155, Aachen.

Tilli T.A.(1993).Practical tools for simulation and optimization fuzzy sys tems with various operatators and defuzzification method.In: *First European Congress on Fuzzy and Intelligent Technologies*,Proc.,Vol.1,pp.256-262,Aachen.

ADAPTIVE TUNING OF FUZZY LOGIC CONTROLLERS

E. K. JUUSO, J. MYLLYNEVA and K. LEIVISKÄ

University of Oulu, Department of Process Engineering, Control Engineering Laboratory
Linnanmaa, 90570 Oulu, Finland

Abstract. A Linguistic Equation Framework developed for adaptive expert systems provides a flexible environment for tuning fuzzy logic controllers. Simulation results, expert knowledge and process experiments can be combined in the development procedure. Controllers represented by compact matrix equations are easily combined with corresponding linguistic process models. The controllers are tuned by adjusting the meanings of the linguistic variables to different working areas. The results are used in real control practice by transferring them to automation systems, or to a fuzzy logic controller *FuzzyCon* where the rules and the membership functions can be changed on-line. FuzzyCon is connected to processes with a data acquisition card, and the data is transferred through DDE-links.

Key Words. Adaptive systems; fuzzy systems; process control; expert systems; process models; simulation; nonlinear systems; knowledge engineering

1. INTRODUCTION

The theory of fuzzy logic provides a method for converting the control knowledge of an operator into a control strategy. Usually, a fuzzy logic controller models the operator rather than the process. This is a quite useful procedure if there really is a lack of a well-posed mathematical model, or if the process is highly nonlinear and sensitive in the operation region. The method also provides an intuitively appealing form of rules which are more readily customizable in natural language terms than conventional controllers.

However, there are also serious problems in designing a fuzzy logic controller. Acquiring the knowledge from the human operator is a tedious and time-consuming task at least if the rule-based procedure is used. Also a trial-and-error based tuning is non-trivial and time consuming, and therefore, far from acceptable. The optimality and stability of the FLC is also hard to prove. Especially for more complicated applications, the designing procedure of FLC must be improved. Actually, it is better to use FLC together with conventional control systems.

In this paper, the emphasis is laid on the methods which could be useful in combining existing simulation models and new ideas of describing qualitative models for the development of combined control system. Expert's experience and control engineering knowledge is used in formulating the models and in defining the estimates of some membership functions. Operator's control actions are used together with fuzzy and linguistic models in tuning the system.

2. LINGUISTIC SIMULATION

Rule-based programming is commonly used in the development of expert systems. However, this paradigm leads to serious problems in practical applications. Maintaining massive rule-based systems is practically impossible. Actually, it is not even possible to reliably test the system in the first place. Therefore, linking the rule-based systems to more efficient modelling methods is essential for practical systems.

2.1. Linguistic rules

The linguistic simulation was originally based on linguistic rules,

$$if \bigcap_{j=1}^{M} Z_j^* \ then \ Y^* \qquad (1)$$

where Z^* and Y^* are linguistic vectors, and there was a clear distinction between input and output variables (Juuso and Leiviskä, 1990). In the present tuning system, the linguistic rules used in previous systems are replaced by linguistic relations and linguistic equations (Juuso and Leiviskä, 1991). In order to get flexibility, the rules are usually used in final applications.

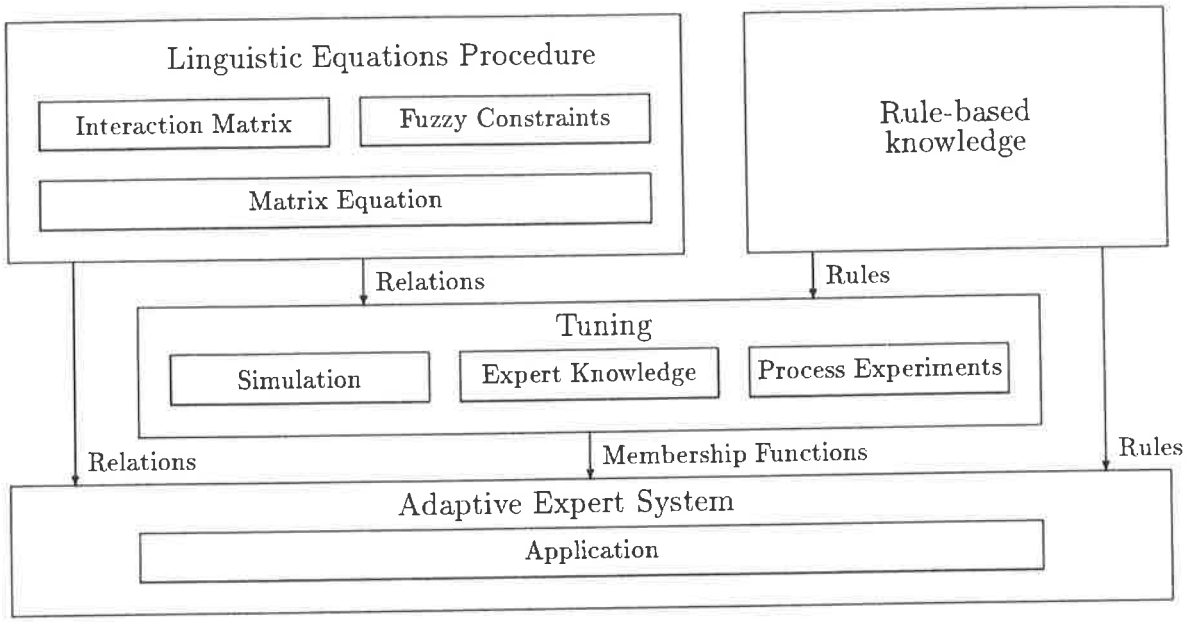


Figure 1: Development of adaptive expert systems.

2.2. Linguistic relations

The linguistic process model is described by groups of linguistic relations: each group can be based on a single fuzzy model, or several fuzzy equations can be aggregated into a single group of linguistic relations (Juuso and Leiviskä, 1991). The variables of the relations are chosen in such a way that the directions of the changes are balanced, e.g. the change-of-control output, Δu, decreases with increasing error, e, and increasing change-of-error, Δe.

A fuzzy PI controller is usually represented by relations $control(x, y, z)$ where x, y and z are the linguistic values for the error, e, the change-of-error, Δe, and the change-of-control output, Δu, respectively. Each relation describes which linguistic values belong together, e.g. $control(normal, normal, normal)$, $control(negative\_small, positive\_small, normal)$. The complete set shown in Fig. 2 consists of 25 linguistic relations if each variable has five linguistic values: *negative_big*, *negative_small*, *zero*, *positive_small*, *positive_big*. Also finer partitions are used for control purposes.

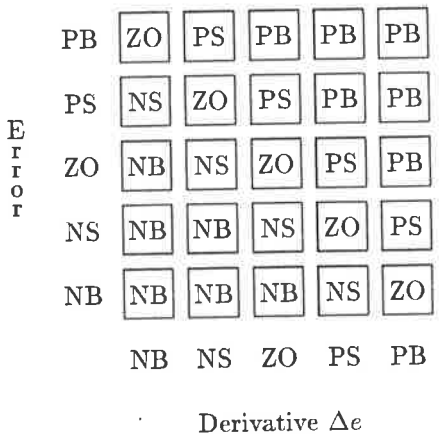


Figure 2: The rule base of a Fuzzy PI Controller.

2.3. Linguistic Equations

A Linguistic Equation approach developed for expert systems provides a flexible environment for combining expertise. The knowledge base of the expert system is represented by linguistic relations which can be changed into *matrix equations*. The reasoning is based on these equations or on the aggregated sets of linguistic relations obtained by solving the equations. The system is adaptive since the meaning of the linguistic values depends on the working point of the process. This presentation is easily generalized for finer fuzzy partitions and transferred between the programming systems (Juuso and Leiviskä, 1993).

A set of linguistic relations can be changed into a compact *equation*

$$\sum_{j=1}^{m} A_{ij} X_j = 0, \qquad (2)$$

where X_j is a linguistic level for the variable j, $j = 1...m$, i.e. the linguistic values *very_low*, *low*, *normal*, *high*, and *very_high* are replaced by numbers -2, -1, 0, 1 and 2. The direction of the interaction is represented by coefficients $A_{ij} \in \{-1, 0, 1\}$. If an interaction is not present, $A_{ij} = 0$.

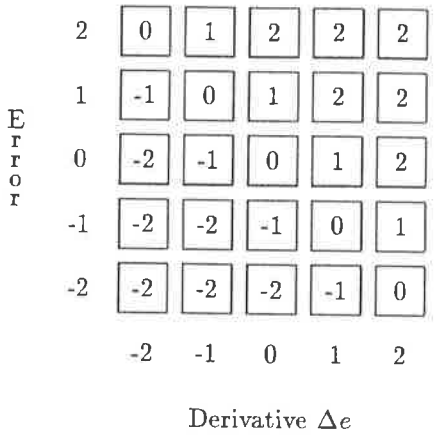


Figure 3: The rule base of a Fuzzy PI Controller in the matrix form.

Fuzzy PI Controller. The rule base shown in Fig. 2 can be represented in a matrix form if the linguistic values, *negative_big*, *negative_small*, *zero*, *positive_small*, *positive_big*, are replaced by numbers -2, -1, 0, 1 and 2 (Fig. 3). All these rules can be obtained from a single linguistic equation

$$\Delta u = e + \Delta e, \tag{3}$$

which is a special case of Equation 2 with the interaction matrix $A = [\ 1 \quad 1 \quad -1\ ]$, and variables $X = [\ e \quad \Delta e \quad \Delta u\ ]^T$.

Fuzzy PD Controller. The table of rules shown in Fig. 2 can used also for fuzzy PD Controller represented by a single linguistic equation

$$u = e + \Delta e, \tag{4}$$

which is a special case of Equation 2 with the interaction matrix $A = [\ 1 \quad 1 \quad -1\ ]$, and variables $X = [\ e \quad \Delta e \quad u\ ]^T$. The PI and PD controllers shown above can also be combined into a single matrix equation, i.e. there are two output variables.

Several equations. Several sets of linguistic relations can be combined by matrix presentation $AX = 0$. In order to solve this problem, a sufficient number of these variables should be known or variated. Because of nearly singular matrices, some of these combinations cannot be used. However, only the integer solutions are required, and exactly the same set of solutions is obtained by any combination.

The result is an aggregated set of those linguistic relations which are relevant if the process constraints described by the complete set of linguistic equations are taken into account. As non-integer alternatives correspond to the solutions in finer fuzzy partition, they can be excluded.

3. CONTROLLER TUNING

Membership functions are tuned by simulation experiments with multilayer *simulation* systems. The simulation system can also be (partly) replaced by *experts* or by *experiments* with real systems (Fig. 1). Both analytical and heuristic knowledge can be used simultaneously. As many rules as possible are replaced by linguistic relations. However, some of them are needed, and the system provides a flexible environment for combining these rules with more efficient modelling methods. In the linguistic equation approach, the relations are developed gradually: only a small part of the problem is taken into account at a time.

3.1 Controller Rules

In the general case, a set of fuzzy inference rules is represented by a single linguistic equation (Juuso, 1993a)

$$u = \sum_{j=1}^{m} A_{ij} X_j, \tag{5}$$

where u is a control action, and X_j a linguistic level for the variable j obtained by the measurements. is also applicable on more detailed fuzzy partitions. This procedure produces always a rule set which is complete, consistent. and continuous. If a noncomplete set is satisfactory, a part of the rules can be rejected already before tuning.

As all these control equations are only special cases of the model above, the system can easily take into account the principles of the model reference control by combining the control equations and the process model into a single set of equations.

For an industrial project, the control system was originally developed on the basis of operator's control actions. Eight variables was used, and the resulting rule base of 22 rules was changed into five sets of rules which corresponded to five linguistic equations. After solving the equation, a more complicated system was handled by 27 rules. The equation system can be used in developing a rule base for finer partitions as well.

In the Linguistic Equation Method, interactions are benefical: they reduce the number of rules necessary for handling the system. It is also possible to identify interactions on the basis of experimental data. In the equation form, large systems are handled quite easily compared to conventional fuzzy methods. For some systems, a really drastic reduction of rules is achieved (Juuso and Leiviskä, 1993).

3.2 Membership Functions

The controller tuning is started by defining the working area for variables, e, and Δe, by fuzzy trapezoidal numbers, several experts are used if available. Fuzzy differential constraints can be used in a similar way as in the DSS applications (Juuso *et al.* 1993). Alternatively, the feasible range can be defined on the basis of experimental data. Actually, the approach is chosen for each variable separately. The feasible range corresponds normally to labels -1, 0 and 1 (Fig. 4).

The final membership functions of the labels are obtained by a polynomial regression model (Fig. 4) which takes into account the sequence of the labels, e.g. positive big is bigger than positive etc. The polynomial model produces more labels close to the working point. The system generates membership functions for finer partition levels.

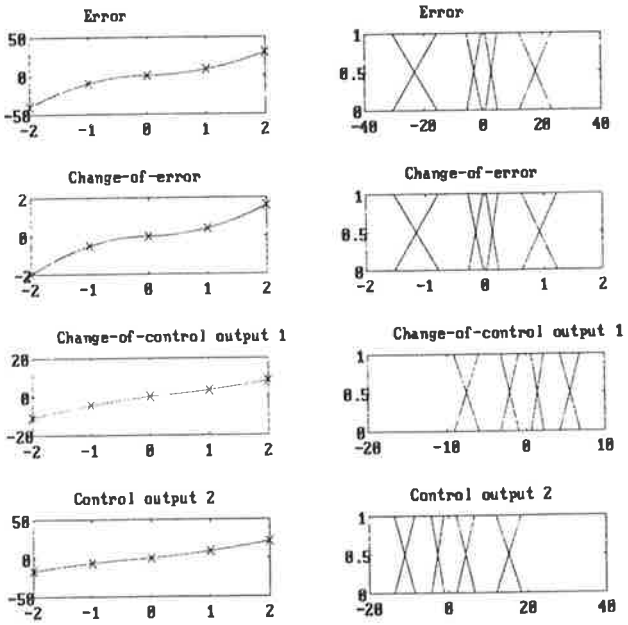


Figure 4: Labels and membership functions for e, Δe, Δu_1, and u_2.

The membership functions of the process state variables, e, and Δe, are used in developing scenarios for experts (or simulations system), and the response produces data for the estimation of membership functions for the labels of the change-of-control-output, Δu (Fig. 4). For some variables, the scenarios and the expert response is replaced by experimental data. In this case, selected input variables are classified by fuzzification routines, i.e. several relations must be taken into account simultaneously. The same methodology is also used when the system is adapted to several working points close to each other.

3.3 Controller Testing

In the tuning system, crosspoint ratio between the membership functions of the neighbouring labels is one for both antecedent and consequent variables. Therefore, the *fuzzification* of the crisp input values on the basis of trapezoidal membership functions is implemented very efficiently. The *knowledge base* and the *inference engine* are implemented in two alternative ways, i.e. traditional rule-based controller and matrix controller. Actually, the rule-based controller is also running in a matrix form similar to one used in FuzzyCon.

The *defuzzification* module is based on the Center-of-Area method (Fig. 5), in the litterature also referred to as Center of Gravity method. In a general case, this method is rather complex and slow. However, our implemetation is quite fast since it is specialized to the membership functions with the crosspoint ratio is one. The resulting control surface (Fig. 6) is very smoothly varying compared to trial-and-error based fuzzy controllers. As the Fig. 5 shows, diagnostical features are a essential part of the system (Juuso 1994).

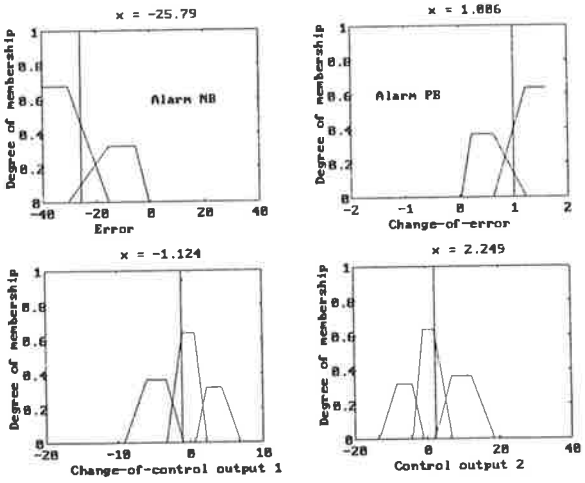


Figure 5: Control Example of a Fuzzy PI Controller with diagnostical features.

3.3 FuzzyCon

For adaptive tuning of fuzzy logic controllers, it is necessary to have a FLC where the rules and the membership functions can be changed on-line. Although a wide variety of fuzzy logic tools available on a PC enviroment were examined, none of them could come up with the requirements of on-line tuning. The knowledge about the systems was limited, and transfering to new computer enviroments would have been difficult.

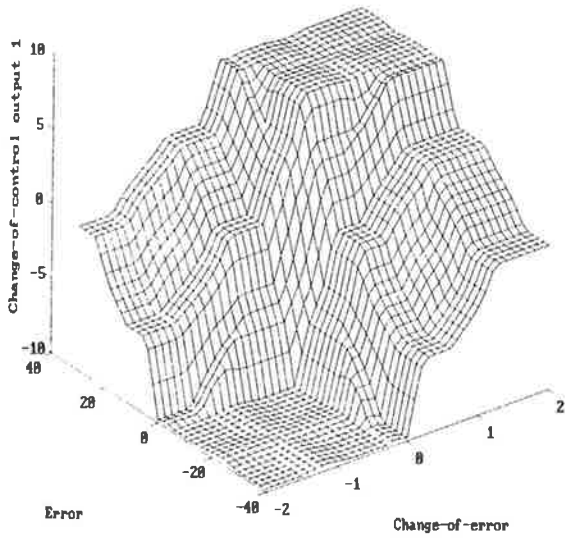


Figure 6: Control Surface of a Fuzzy PI Controller.

Usually, fuzzy logic development tools permit changes in rules and membership functions in simulation mode, but when the code is compiled and the program is running, changes can not be done whithout compiling the code again. The calculation time in adaptive FLC is a little bit longer than in strictly bound FLC (Brubaker 1993), but the speed is enough for controlling processes in process industry.

FuzzyCon allows adaptive tuning and offers good tools for tuning, and is specially designed for fuzzy control in process industry (Juuso *et al.* 1994). As a Windows application created in Visual Basic 3.0, it uses the advantages of Windows: graphical interface, windows and DDE-links. The links are created after applications are started and there is no need to know the memory addresses beforehand. The data acquisition application makes conversions, if needed, to crisp input data and sends them to FuzzyCon. At the moment FuzzyCon can handle ten inputs and five outputs, but the expansion is easy.

The data acquisition card has a Windows support, so the data acquisition program can use DLL-library commands for reading measured data from the card.

Fuzzy logic controller. The knowledge base contains membership functions and rules. The membership functions are trapezoidal or triangular defined by four points (Fig. 4). The maximum number of membership functions is nine for each input and output variable. In the general form, each rule contains several inputs and outputs: at the moment, FuzzyCon can handle ten inputs and five outputs.

Some additional labels are required for the systems consisting of several sets of rules: ANY-label in the premise part of the rule means that the input has no influence on the rule, and a star in some output in the consequent part of the rule means that the rule has no influence on that output. Rules are stored in number form which makes their treatment easier.

Fuzzification is made by calculating the values of membership functions of all the fuzzy sets for input value in question (Viot 1993). Mamdani's min-max method was selected because it provided reasonably good results and was fastest and easiest to calculate (Lee 1990). FuzzyCon does not require similar restrictions for membership functions as the tuning system described above, and therefore, the fast algorthm for the Center-of-Area method cannot be used. The defuzzification module is based on the Center-of-Sums method which is one of the most common defuzzification techniques in control.

User interface. FuzzyCon provides valuable on-line tools for the controller tuning. The user interface is easy to use and several windows can be open at the same time for showing different kind of data. FuzzyCon can also save definitions, membership functions and rules on the file system in matrix forms which are closely related to the matrices used in the tuning system.

For users, the rules are shown in linguistic form and the membership functions in numeric and graphical form. Users can add and edit rules in rule window and edit membership functions in membership function windows, or the rules and membership functions created in some other way can be read to FuzzyCon from the file or transfer with DDE-links. The rules and membership functions can changed on-line by both methods.

FuzzyCon offers several aids to examine control (Juuso *et al.* 1994). The completeness of the rule base can be tested by simulation. During control and simulation there is chance for monitoring the firing of the rules and the forming of degrees of membership for inputs and outputs.

4. ADAPTIVE FUZZY CONTROLLERS

The Control System is adaptive since the meaning of the linguistic values depends on the working point of the process. Only five parameters are needed for reconstructing the set membership functions for any variable on any level of fuzzy partition (Fig. 4). In control applications, this level does not usually change, and it is better to use a set of corner points of the membership functions also shown in Fig. 4.

The adaptivity is pretuned, i.e. the tuning of the membership functions is performed for different working point areas defined by some suitable state variables, e.g. flow velocity, temperature etc. The resulting membership functions, and the corresponding control surface, depend on the working point, i.e. the cube containing the control surface has rubber like dimensions. The selection of alternative tuning areas corresponds to the table shown in Fig. 2, and each variable of the control rules requires an own table.

These tables are used together with linguistic equation models, if available, to obtain the appropriate definitions for the sets of membership functions. This procedure allowes gradual changes in any part of the set of membership functions, and therefore, it is more flexible than Normalization-Denormalization procedure. By this approach, adaptive control can be realized in FuzzyCon or in automation systems as well. The membership function can be regenerated by the database produced by the tuning system.

If the database defining the membership functions is changed online, the adaptation should be restricted to an appropriate range depending on the working point. Otherwise, the normal problems of adaptive systems may arise. Neural nets are planned to use in fine tuning especially when extending the operating area of the adaptive fuzzy controller.

5. CONCLUSIONS

In the linguistic equation approach, the relations are developed gradually: only for a small part of the problem is taken into account at a time. By the matrix method, it is very easy to develope and tune adaptive fuzzy control applications. The linguistification methods have a vital importance in connecting this methodology to the real practice: the relations are tuned by adjusting the meanings of the linguistic variables.

The results of the adaptive tuning method are used in real control practice by transferring them to FuzzyCon which is a fuzzy logic controller where the rules and the membership functions can be changed on-line. FuzzyCon is connected to processes with a data acquisition card, and the data is transferred through DDE-links.

6. REFERENCES

Brubaker, D. 1993. "An accelerated kernel for fuzzy systems." *AI Expert* 8, no. 3:39–44.

Juuso, E. K. 1993. "Linguistic Simulation in Process Control." In *Applied Simulation in Industry. Proceedings of the 35th SIMS Simulation Conference (Kongsberg, Norway, 9 - 11 June)*, T. Iversen, ed., 107–113.

Juuso, E. K. 1994. "Fault Diagnosis Based on Linguistic Equation Framework." To be presented in *SAFEPROCESS'94 (Espoo, Finland, 13-15 June, 1994)*.

Juuso, E. K., J.C. Bennavail, and M.G. Singh 1993. "Hybrid Knowledge-Based System for Managerial Decision Making in Uncertainty Environment." In *Qualitative Reasoning and Decision Technologies, Proceedings of the QUARDET'93 (Barcelona, June 16 - 18)*, N. Piera Carreté and M. G. Singh, eds., CIMNE, Barcelona, 234–243.

Juuso, E. K. and K. Leiviskä. 1990. "Expert Systems Combined with a Multilayer Simulation System." In *MATHEMATICAL and INTELLIGENT models in system simulation*, R. Hanus and P. Kool and S. Tzafestas, eds., Baltzer, Scientific Publishing, 325–330.

Juuso, E. K. and K. Leiviskä. 1991. "Adaptive Expert Systems for Metallurgical Processes." In *Expert Systems in Mineral and Metal Processing, Proceedings of the IFAC Workshop (Espoo, Finland, August 26-28, 1991), IFAC Workshop Series, 1992, Number 2*, S.-L. Jämsä-Jounela and A. J. Niemi, eds., Pergamon, Oxford, UK, 119–124.

Juuso, E. K. and K. Leiviskä. 1993. "Linguistic Equation Approach for Adaptive Expert Systems." In *Proceedings of the EUFIT'93 (Aachen, September 7 - 10)*, H.-J. Zimmermannn, ed., Augustinus, Aachen, Vol. 3, 1550–1556.

Juuso, E. K., J. Myllyneva, and K. Leiviskä 1! "Fuzzy Logic Controller and Adaptive Tuning." To be presented in *ESM'94 (Barcelona, Spain, 1-3 June, 1994)*.

Lee, C. C. 1990. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I & II." *IEEE Transactions on Systems, Man, and Cybernetics*, 20:404–435.

Viot, G. 1993. "Fuzzy logic in C, creating a fuzzy-based inference engine." *Dr Dobb's Journal*, 18(2):40–49.

STEPS TOWARDS REAL-TIME CONTROL USING KNOWLEDGE BASED SIMULATION OF FLEXIBLE MANUFACTURING SYSTEMS

J. NACSA and G. L. KOVÁCS

Computer and Automation Research Institute of Hungarian Academy of Sciences,
Computer Integrated Manufacturing Research Laboratory, H-1518, Budapest, POB63, Hungary

Abstract. Computer aided simulation can assist both in the design and operation of flexible manufacturing systems (FMS). A proper simulation model of a given FMS could be the best tool to validate it and to evaluate its performance. The control of the system can be solved by separating the control mechanism from the real devices of the FMS with a communication interface. In this case it is possible to use the simulated system for investigations instead of the real system. Such simulation systems can be built up using expert system shells. In this paper an FMS simulation system (SSQA) will be introduced with the hybrid application of expert systems and a traditional simulation software. The system has some evaluation, scheduling and quality control power as well, according to the implemented advisory systems that communicate with the simulation package.

Key Words. Expert systems; Flexible manufacturing; Simulation;

1. INTRODUCTION

The manufacturing process in a manufacturing system is defined by the manufacturing schedule that is created from process plans and actual orders. Process plans describe the manufacturing of every type of parts in operation order, the alternative machines, tools, etc. to all operations and the time period of each operation. The actual order contains the number of parts required within a given time period. The manufacturing schedule has the information of the process plan (the time periods and dedicated machines and tools instead of alternatives) plus the starting times of the operations.

A good scheduler can take into account the duration of transportation and set-ups as well if they may not be neglected. In this point of view the control of an FMS means to produce all necessary information for each equipment of the system in time and to activate them by an information transfer.

In this term a scheduler with real-time capabilities is the key of computerised control of an FMS. That is the main reason why we prefer the application of real-time software even for simulation where it was not necessary. The big problem of this approach is the interfacing of the real controlled devices to the scheduler. In the most cases this part of control system is mainly application specific. We think that using standard communication interfaces in the shop-floor level (MMS-OSI networks) this problem will be easier. (Nacsa 1994).

Control of an FMS is a special case of processcontrol, where most events are not continuos but discrete giving good chances of using digital computers for real-time control. In a typical process control application there are more but simpler I/Os, so the complexity of an FMS is similar.

2. DECISION MAKING IN SIMULATION OF FMS

A simulation model is a perfect tool to evaluate the performance of FMSs as different schedules can be examined with minimal costs, as well as the effects of different system problems, as break-down situations, etc. can be checked (Law 1992). During the simulated manufacturing process, which is based on the process plan, the starting and finishing time of each operation can be recorded, and later on a schedule can be built up based on this information.

The power of schedules produced by simple simulation is generally not comparable with schedules resulted from sophisticated, specialised scheduling algorithms, but they are produced definitely faster and are reliable. They have advantages in the case of the often needed re-scheduling. When a breakdown occurs, a new schedule created through simulation offers very soon a way to continue the production if it's possible.

During creation of manufacturing schedules with simulation there are several decisions - even if only

simple algorithms are used - which can hardly be processed with the generally available if-then-else structures of traditional simulation languages. For example:

- if there are more than one workpieces in a parallel buffer (temporary storage), the question is which workpiece can leave the buffer first, or
- if more than one machine tool is suitable for a certain operation, which one should be chosen, or
- if more workpieces are waiting for a certain resources which one to choose,
- etc.

Another problem that these decisions often have many on-line parameters, so the coding of such a decision in a simulation language is rather complicated. For example:

- managing assembly operations defined by process-plans.
- a process plan has alternative paths.
- etc.

Combining simulation systems and knowledge processing methods can be solved the problem of decision making during simulation. This combination will result in the so called knowledge based (KB) simulation, where KB advisors are somehow connected to the simulation.

There are many simulation tools where user written functions can be added to handle such problems. Some knowledge based systems have also simulation possibilities. Third way is to establish an on-line communication between an independent simulator and a knowledge based system. We chose the third one because of two reasons. This solution lets us change later the simulator to an existing system to provide real-time control. In the same time we build up the system that the application specific parts are mainly in the simulator so the object structures, the user-interface, many rules and the interface itself can be user in other applications.

Answering to the questions of the simulator requires some special capabilities from the advisor.

- Run time information about the facilities, buffers, orders, deadlines, etc.
- The look-ahead to the 'near' future (next operations of workpieces and theirs effects)
- A decision method for selecting scheduling algorithm.
- The constraints of the manufacturing.
- etc.

3. EXPERIMENTAL KB SIMULATION SYSTEMS FOR FMS

3.1. Application of CS-PROLOG

The first Knowledge Based FMS Simulation system developed in the CIMLab of CARI (Kovács 1992) was written in a special rule based simulation language, CS-PROLOG (Communicating Sequential Prolog) (Futó 1987), which is a PROLOG language extended with simulation facilities. When the system began to grow different problems of the PC based system appeared making the development harder. The speed performance of the PROLOG program was decreasing radically.

3.2. Application of hybrid systems - SSQA

Then a new system (SSQA - Simulation-Scheduler-Quality Assurance) was defined to reflect the idea of connecting a traditional simulation system to expert systems (deep coupled hybrid system). At the same time it was realised that some quality control power can relatively easily be incorporated into the system.

SSQA consists of a traditional simulation system coupled with three expert systems. The four main modules are: Simulation-Animation System (SAS), Preparation Expert System (PES), Advisor Expert System (AES) and Evaluation Expert System (EES). AES is deep coupled to SAS, while PES and EES are shallow coupled (Fig. 1).

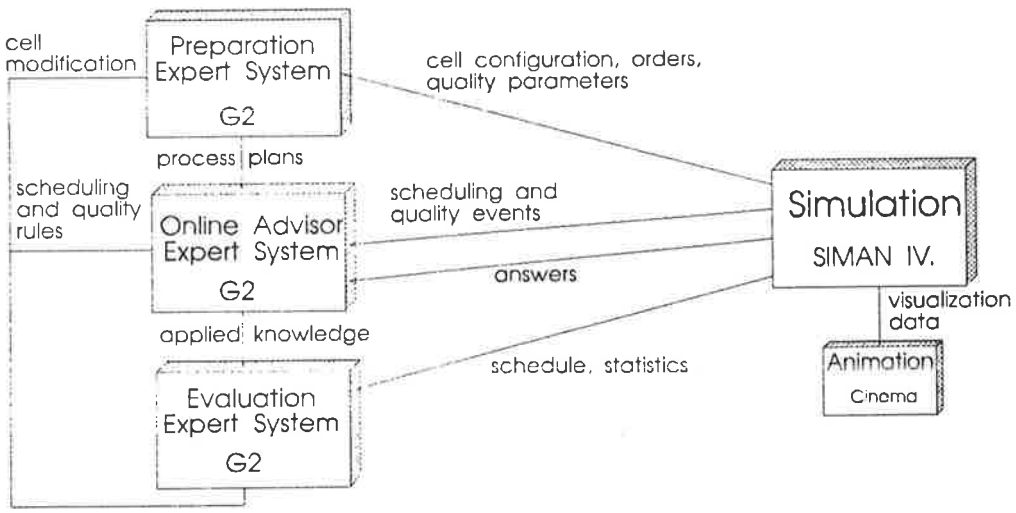


Fig. 1 The main logical modules of SSQA hybrid system

We defined a Preparation and an Evaluation Expert Systems because many information processings can be done before and after the simulation. It depends on the given application and its special problem how much knowledge processing is used in each expert system module.

- The Preparation Expert System (PES) collects all input data for the simulation and completes the simulation model. In SSQA we start from a given simulation model and let this module to manage the orders, the process plans and every experimental specific part of the simulation, e.g. getting a new schedule after a breakdown event means that PES initiates the given situation of the workshop after the crash.

- The Simulation-Animation System (SAS) executes the simulation model generated by the PES. Both scheduling and quality control functions need many decisions, which are made in the Advisor ES. If the simulation needs help from the AES then the simulation halt, sends its question to the AES and waits for the reply. During the simulation a graphical animation - like an animation movie - helps to understand and follow the simulated manufacturing process on the computer's screen.

- The Advisor Expert System (AES) is the slave of the simulation. It waits for the questions of the simulation on certain decision points. Receiving the question the AES starts its inference process and sends back the concluded answer. The knowledge base of the AES consists of scheduling and quality control rules. Workpiece and resource priority rules belong to the scheduling part of the knowledge base, and measurement evaluation rules to the quality control.

- The Evaluation Expert System (EES) evaluates the results of the simulation that are the utilisation statistics of all equipment in the FMS and the manufacturing schedule.
  The EES has a statistical evaluation power, too. It is applied when not only one, but several simulation runs are processed in a row and all results are evaluated together. The number of such simulation runs (10-100 or more) depends on the size of the FMS, on the number of different parts to produce, on their batch sizes and on the available scheduling and quality control algorithms implemented in the advisory system (AES). The calculation of this number is a complicated task, and recently blind guess methods are used, because we do not yet have a good method for it.
  Depending on the knowledge base of the EES it can decide whether the schedule and the cell configuration are acceptable or not, and in this later case modification can be suggested. These suggestions may be the activation of other rules by the Advisor ES or the modification of the original configuration of the manufacturing cell by the Preparation ES.

3.3 Application of G2 to build advisory systems

In the first version of SSQA (Kovács 1993) a CS-PROLOG based expert system called ALL-EX were used as the advisor and evaluation ES, and the simulation-animation part of the system was a SIMAN/Cinema module.

The testing and evaluating of the prototype version showed that ALL-EX did not have enough power to serve as an on-line expert system for the simulation. Communication, speed and memory problems were analysed. So more accepted and commonly used AI tools were examined (MULISP, CLIPS, NEXPERT, G2). The real-time expert system G2 was chosen because of its high speed, communication features and built-in capacities (procedures, rule classing, user-friendly support of development). Comparing the costs of the potential simulated FMSs the price of G2 was acceptable.

At the beginning SIMAN/Cinema were running on a PC, while a SUN SPARCstation was used for G2. To run the two systems together an interface between SIMAN and G2 (Nacsa 1994) was developed. Both the network interface software and the application of SIMAN on a real size problem have high memory requirements that are rather difficult to provide under the DOS. So in the recent version the SIMAN/Cinema is running on a SUN, too. This way three tasks are running parallel: the G2 knowledge base, the SIMAN simulation and a communication server.

All the knowledge of the advisors implemented in ALL-EX was transformed into the appropriate formats of G2 without major problems.

The main difference was caused by the G2's object-oriented view of a system. All elements of the simulated system are now represented by G2 objects. There exists a hierarchy between the objects that makes the attribute inheritance possible. In the following some of the classes we have defined are listed, together with their attributes (inherited attributes are not mentioned):

- workpiece (state, present-station, next-station, current-station, type, color, process-plan, option)
- process-plan (first-element, type, color, is-required, sum-of-is-in, sum-of-is-req, sum-of-is-prod)
- operation (machine, setup time, operation time, alternative operation, next operation, quality pointer)
- quality parameter(expected means, tolerance range, type of measurement, repairable error, etc.)
- cell-unit (state)
  - transport-device (state, distance, speed, speed-level, radian)
  - cell-equipment (state, place-list)
    - store (wp-list, place-list)
    - machine (place-list)

4. FIRST APPLICATIONS OF THE SYSTEMS

The experimental system was developed first to simulate a real system - the pilot FMS of the Technical University of Budapest. Fig. 2. shows the layout of the system.

There are four cells (assembly /1/, storage + AGV /7/, measurement /6/, machining /2,3,4,5/) in the system. The machining cell consists of a CNC machining centre /3/, a CNC lathe /4/ and two robots /2,5/. The system has an input buffer with 30 storage places and each machine tool and station has one input/output buffer, one automatic pallet changer and one working space.

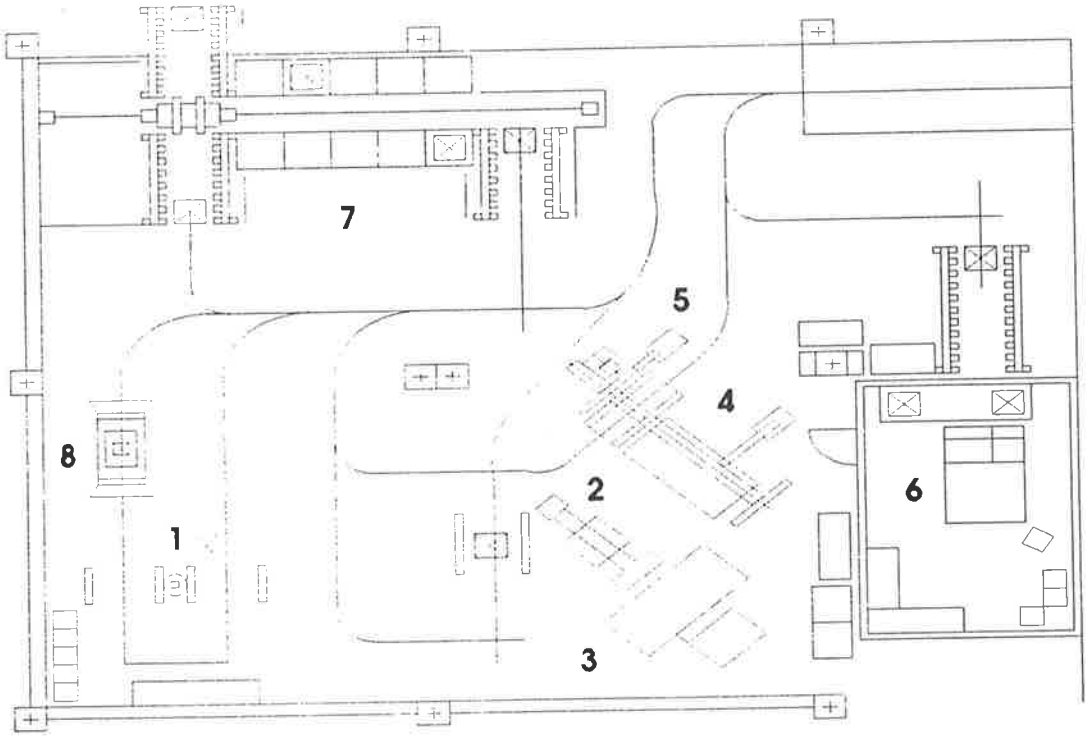


Fig. 2 Pilot FMS in the Technical University of Budapest

Other applications are also under development:
- the simulation and control of the MAP Training Center in CARI of HAS.
- the simulation and evaluation of a workshop in the ZALA Furniture Ltd., Hungary
- the simulation of the FMS of Goldstar Cable, Heavy Industries in South-Korea.

5. CONCLUSIONS

A hybrid KB simulation program was developed with scheduling and quality assurance features for FMS. The simulation and the decision making parts (scheduling, quality assurance) were separated to give better performance. The cost of it was the development of an interface.

The implementation in SIMAN and G2 was fast enough and G2 was excellent to make the different experimental program runs. Data preparation is - however - a tedious and time-consuming activity.

We are convinced that the application of these up-to-date means will lead our CIMLab to have a useful Knowledge Based Simulation-Scheduling-Quality Control program package to support the evaluation and better performance of working FMS and to be used in the design (planning) of new FMS implementations. Finally it has to be mentioned that using the discussed hybrid program structure a potential system control is supported. In this case the real FMS environment should be used instead of the simulation. So one main direction of our recent R&D work is real-time FMS control.

6. REFERENCES

Futó, I. (1987). AI and Simulation on PROLOG Basis. In: Int. Symp. on AI, Expert Systems and Languages in Modeling and Simulation, Barcelona, Spain

Kovács, G.L., I. Mezgár, S. Kopácsi (1992). A PROLOG Based Manufacturing Cell Design System. In: *Proceedings of The Practical Application of Prolog Conference*, London, UK.

Kovács, G.L., and S. Kopácsi (1993). A Knowledge Based Simulation System for Flexible Manufacturing. In: *Proceedings of the 1993 IEEE International Conf. on Robotics and Automation*, Vol. 1. pp. 858-863.,

Law, M. and M. G. McComas (1992). How to Select Simulation Software for Manufacturing Applications. *Industrial Engineering*, July,

Nacsa, J., G.L. Kovács (1994) Communication problems of expert systems in manufacturing environments. *Symposium on Artificial Intelligence in Real-Time Control*, Valencia, Spain, October 3-5.,1994 (accepted)

INTELLIGENT ACTUATION AND MEASUREMENT SYSTEM-BASED MODELLING : THE PRIAM WAY OF WORKING

D. GALARA**, B. IUNG*, G. MOREL* and F. RUSSO ***

* *CRAN-EACN Université de Nancy I BP 239 54506 Vandoeuvre (France)*
** *EDF Direction des Études et Recherches, 6 quai Watier 78400 Chatou (France)*
*** *ENEL 1 Via Volta 20093 Cologno Monzese MI (Italy)*

Abstract. In most of the production systems, the main control functions are nearly similar. In order to save this common knowledge and know-how and to reuse it during another application design, it is necessary to practise a **new rupture way of re-engineering** in which the specification phase is completely independent of all implementation. Indeed to respect the genericity need of such an approach, the technological constraints only have to be taken into account at the time of the function distribution inside the automation systems or directly down to the field devices.

Key words. Intelligent Actuation and Measurement, Functional Requirement Diagrams, Functional Diagrams, Functional Companion Standard, Distributed Process Control, Automation Engineering.

1. INTRODUCTION

In the context of the retrofitting of the industrial sites based on another process control **specification** and **structuration**, the main innovating principle of both European projects which are linked and entitled ESPRIT II - D.I.A.S. (Distributed Intelligent Actuators and Sensors) n° 2172 and ESPRIT III - P.R.I.A.M. (Prenormative Requirements for Intelligent Actuation and Measurement) n° 6188, is to propose a **new rupture way of reengineering** (Hammer, 1993).

Indeed the DIAS project has developed since 1989, the Intelligent Actuator and Transmitter concept which advocated the intelligence distribution down to field devices within a C.M.M.S. (Control, Maintenance and technical Management System) architecture based on the integration of these three conventional islands of automation in order to improve basically the design, operation and conditional maintenance. The experimentation resulting of a semi-formal reference Knowledge Oriented Design approach allowed to demonstrate the concept feasibility and its industrial interest (Iung, 91).
The DIAS basic rupture way is to promote the intelligence distribution in opposition with the current hierarchical process control in order to implement an IAT Information driven instead of an IAT Data driven. The IAT concept materialised by an organic notion rather than a functional one, is necessary but no more sufficient to satisfy the process user needs.

According to these results, the PRIAM project aims to develop the IAM (Intelligent Actuation and Measurement) concept as integrated sub-system of whole of the process system.

The PRIAM way of working is based on the prototyping of the CMMS reference approach by both the user and the vendor in order to qualify and to quantify the IAM interchangeability and interoperability gap.
At this stage, the second rupture way is to promote an IAM application driven implementation independent as opposed to an IAM technology driven implementation dependent.

In additional **prenormative** view, the on-going European Intelligent Actuation and Measurement User Group (ESPRIT project n° 8244) extends this way of thinking and working to an independent and open forum which is composed of end users, IAT suppliers, engineering companies, scientific and which is linked with national and internal standard committees.

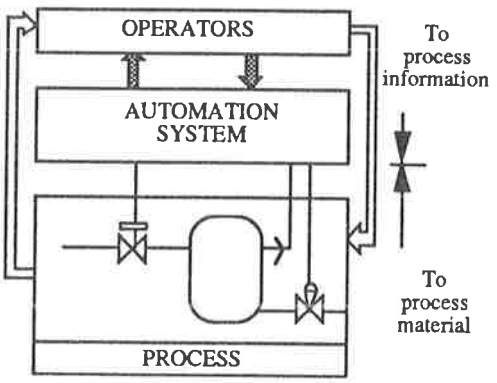


Fig. 1. The plant : Process, Automation System and Operators

2. IAT CONTEXT

Any plant (see Fig.1) is composed of the **process**, processing raw material to transform it into products, the **automation system** processing information from/to the process and the operators, the **operators** operating the process through the automation system.

From this structure emerges two interface types : on the one hand a **physical interface** which manages the exchange between process and automation system through the **actuators** and the **sensors** and on the other hand the **user interface** representing the semantic link between the operators and the automation system. The physical interface is composed of :

- the data coming from the sensors and used by the processing,
- the data coming from the actuator instrumentation and used by the processing,
- the information provided by the processing to the actuator command.

On another conceptual level, the user interface contains :

- the requests coming from the operators for the processing.
- the reports coming from the processing for the operators.

According to the content of the elementary data, the physical interface appears as the informational hinge of the plant. Indeed from a reliability and availability point of view, any communication fault through the physical interface implies bad effects on the whole of the production system. Therefore the actuators and sensors considered in theory as trivial elements, make in fact a real Achilles' heel of the plant. A first up-grading of this conventional equipment consisted in the definition of its technological dysfunctioning by the complementary specification of the waited and acceptable services of control. This has led on the **automation object** concept (Morel, 1993) to the development of the **behaviour filter** notion which corresponds to the using of expected behaviour model of each physical element between the control block and the input/output lines to separate application and technological points of view (see Fig. 2).

The automation object notion allowed on the one hand a higher component **reusability** since 80% of the applications are composed of the same basic equipment and on the other hand a higher interoperability by the standardisation of the interface syntax.

From this first behavioural modelling result, the ESPRIT-DIAS project had as objective to extend the control view towards the **maintenance** and **technical management** islands by the implementation of the C.M.M.S. integration concept (Galara, 1993) which is a sub-concept of C.I.M.E. (Computer Integrated Manufacturing Engineering). Therefore the basic automation object has evolved to a **intelligent** automation object : the **Intelligent Actuators and Sensors** representing a new way in the control part structuration (see Fig. 2).

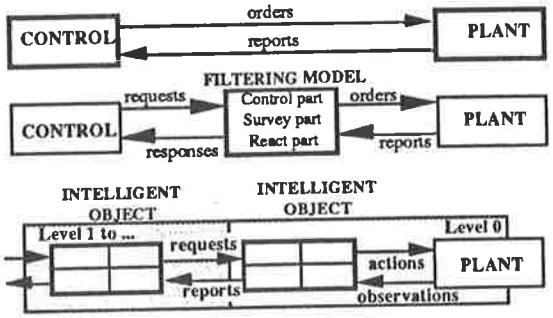


Fig. 2. Control part structuration evolution

Thanks to this intelligence distribution down to field devices (**Intelligent Field Devices**), the physical interface is not only composed of typical analog signal or current software data but also of pertinent, consistent and reliable **information**. The innovating IFD resulting of a KOD approach (Iung, 93) is necessary but not sufficient to completely satisfy the user requirements and particularly the automation system-operators interface.

3. IAM CONTEXT

In fact, the IAT are only equipment quantitatively distributed in the plant which have to be integrated qualitatively inside a real subsystem : the **Intelligent Actuation and Measurement** (Capetta, 1993).
Based on this concept, the ESPRIT-PRIAM project, logical continuation of DIAS, proposes some generic definitions in order to have a same mental image between the users and the vendors :

- Measurement is considered as a set of operations for the purpose of determining the value of a quantity.
- Actuation is also considered as set of operations for the purpose of determining the value of a quantity.

With these two definitions, there is apparently a paradox because measurement and actuation is defined with the same methodology. To avoid such a paradox, it is necessary to go deeper in the definition of "determining the value of quantity" :

- for measurement, the meaning ought to be : to be able "**to observe**" the value of a quantity,
- for actuation, the meaning ought to be : to be able "**to modify**" the value of a quantity.

Measurement and actuation are means of achieving intentions : to observe and to modify a value of a quantity. So, Intelligent Measurement and Actuation integrating IAT equipment have to be considered as subsystems at the two plant interfaces defining an **overlap** (see Fig. 3) taking into account :

- a part of the process : the **IAM machinery** which transforms the observation into a processable signal or the signal into a modification of the value of a flow of material,
- a part of the automation system : the **IAM processing capability** which transforms the signal into pieces of information or a piece of information into a signal.

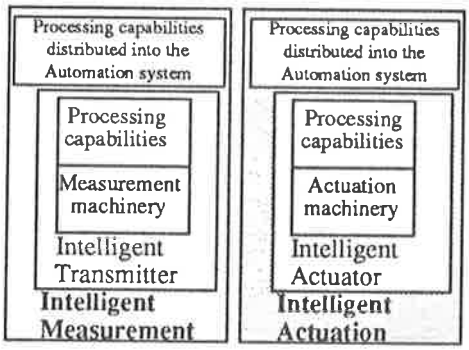


Fig. 3. Intelligent Actuators and Transmitters inside Intelligent Actuation and Measurement

The definition of this IAM is based on a specific life cycle which starting point is a **user need approach** in order to satisfy the user interface : moving from application-driven (the needs) up to technology-driven (the systems).
This functional and systemic approach, established by engineering companies, is entirely **independent** of all implementation of the processing and leads in a first step to the description of the Functional Requirement Diagrams (see Fig. 4) : independent of any implementation into any Automation System.

Implementation independent does not mean completely generic. Indeed the FRD is linked on the one hand to its **environment**, on the other hand to the process which will be controlled and observed by the available actuators and sensors (actions-observations representing the machinery) and finally to the information needs materialised by the control, maintenance and technical management **requests** and **reports**. These requests-reports are normally expressed by a set of **agents** representing the operators considered as End-users and suppliers of IAM devices.

In relation to its global interface, the FRD behaviour is defined by the **rules** which are necessary to implement from the Requests in order to provide the Reports according to the Actions and Observations. This behaviour or processing has to take into account the expected **performances** and the functioning **safety** constraints and contains the **logical, sequential** and **analog** operations.

The processing will later be distributed inside the equipment based on conventional or digital technologies. The distribution established for suppliers and integrators, corresponds to a **technology-driven** dependent of the implementation and represents the Functional Diagrams.

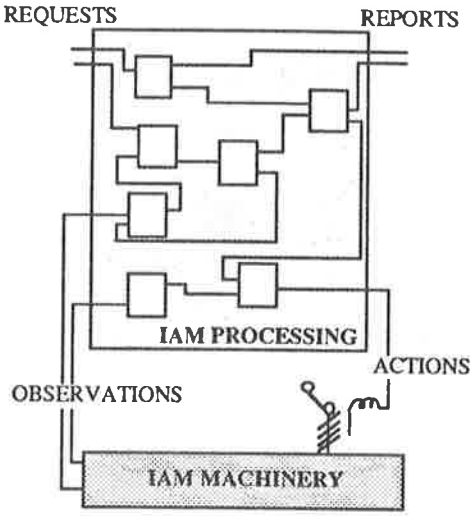


Fig. 4. IAM Functional Requirement Diagrams

The **rupture** between the two FRD and FD islands (see Fig. 5) is situated between the world of Engineering and the world of integrators and suppliers and is made to move from FRD, application driven and implementation independent, up to FD, technology driven and implementation dependent (Morel, 1994).

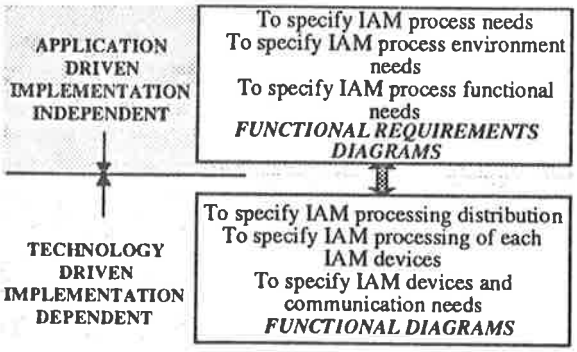


Fig. 5. Functional Requirement Diagrams and Functional Diagrams

4. - FUNCTIONAL REQUIREMENT DIAGRAMS OF IAM

Outside the implementation world, the IAM being immersed to a specific context, the new systemic way of working implies before

4. FIRST APPLICATIONS OF THE SYSTEMS

The experimental system was developed first to simulate a real system - the pilot FMS of the Technical University of Budapest. Fig. 2. shows the layout of the system.

There are four cells (assembly /1/, storage + AGV /7/, measurement /6/, machining /2,3,4,5/) in the system. The machining cell consists of a CNC machining centre /3/, a CNC lathe /4/ and two robots /2,5/. The system has an input buffer with 30 storage places and each machine tool and station has one input/output buffer, one automatic pallet changer and one working space.

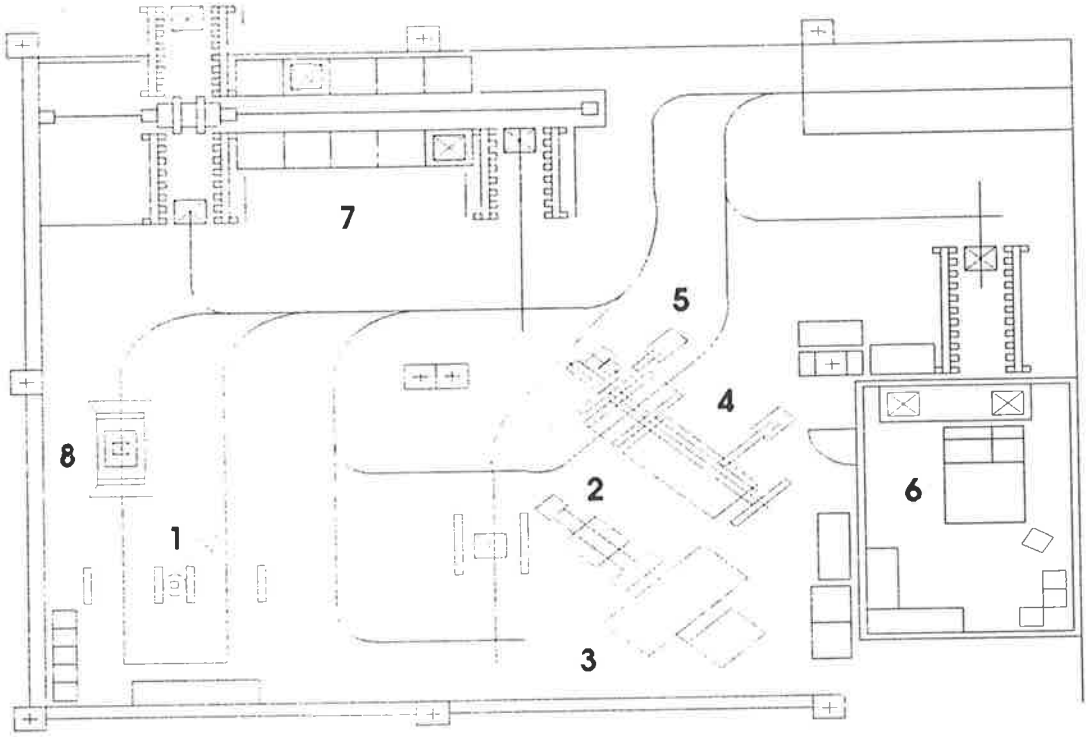


Fig. 2 Pilot FMS in the Technical University of Budapest

Other applications are also under development:
- the simulation and control of the MAP Training Center in CARI of HAS.
- the simulation and evaluation of a workshop in the ZALA Furniture Ltd., Hungary
- the simulation of the FMS of Goldstar Cable, Heavy Industries in South-Korea.

5. CONCLUSIONS

A hybrid KB simulation program was developed with scheduling and quality assurance features for FMS. The simulation and the decision making parts (scheduling, quality assurance) were separated to give better performance. The cost of it was the development of an interface.

The implementation in SIMAN and G2 was fast enough and G2 was excellent to make the different experimental program runs. Data preparation is - however - a tedious and time-consuming activity.

We are convinced that the application of these up-to-date means will lead our CIMLab to have a useful Knowledge Based Simulation-Scheduling-Quality Control program package to support the evaluation and better performance of working FMS and to be used in the design (planning) of new FMS implementations. Finally it has to be mentioned that using the discussed hybrid program structure a potential system control is supported. In this case the real FMS environment should be used instead of the simulation. So one main direction of our recent R&D work is real-time FMS control.

6. REFERENCES

Futó, I. (1987). AI and Simulation on PROLOG Basis. In: Int. Symp. on AI, Expert Systems and Languages in Modeling and Simulation, Barcelona, Spain

Kovács, G.L., I. Mezgár, S. Kopácsi (1992). A PROLOG Based Manufacturing Cell Design System. In: *Proceedings of The Practical Application of Prolog Conference*, London, UK.

Kovács, G.L., and S. Kopácsi (1993). A Knowledge Based Simulation System for Flexible Manufacturing. In: *Proceedings of the 1993 IEEE International Conf. on Robotics and Automation*, Vol. 1. pp. 858-863.,

Law, M. and M. G. McComas (1992). How to Select Simulation Software for Manufacturing Applications. *Industrial Engineering*, July,

Nacsa, J., G.L. Kovács (1994) Communication problems of expert systems in manufacturing environments. *Symposium on Artificial Intelligence in Real-Time Control*, Valencia, Spain, October 3-5.,1994 (accepted)

INTELLIGENT ACTUATION AND MEASUREMENT SYSTEM-BASED MODELLING : THE PRIAM WAY OF WORKING

D. GALARA**, B. IUNG*, G. MOREL* and F. RUSSO ***

* *CRAN-EACN Université de Nancy I BP 239 54506 Vandoeuvre (France)*
** *EDF Direction des Études et Recherches, 6 quai Watier 78400 Chatou (France)*
*** *ENEL 1 Via Volta 20093 Cologno Monzese MI (Italy)*

Abstract. In most of the production systems, the main control functions are nearly similar. In order to save this common knowledge and know-how and to reuse it during another application design, it is necessary to practise a **new rupture way of re-engineering** in which the specification phase is completely independent of all implementation. Indeed to respect the genericity need of such an approach, the technological constraints only have to be taken into account at the time of the function distribution inside the automation systems or directly down to the field devices.

Key words. Intelligent Actuation and Measurement, Functional Requirement Diagrams, Functional Diagrams, Functional Companion Standard, Distributed Process Control, Automation Engineering.

1. INTRODUCTION

In the context of the retrofitting of the industrial sites based on another process control **specification** and **structuration**, the main innovating principle of both European projects which are linked and entitled ESPRIT II - D.I.A.S. (Distributed Intelligent Actuators and Sensors) n° 2172 and ESPRIT III - P.R.I.A.M. (Prenormative Requirements for Intelligent Actuation and Measurement) n° 6188, is to propose a **new rupture way of reengineering** (Hammer, 1993).

Indeed the DIAS project has developed since 1989, the Intelligent Actuator and Transmitter concept which advocated the intelligence distribution down to field devices within a C.M.M.S. (Control, Maintenance and technical Management System) architecture based on the integration of these three conventional islands of automation in order to improve basically the design, operation and conditional maintenance. The experimentation resulting of a semi-formal reference Knowledge Oriented Design approach allowed to demonstrate the concept feasibility and its industrial interest (Iung, 91).
The DIAS basic rupture way is to promote the intelligence distribution in opposition with the current hierarchical process control in order to implement an IAT Information driven instead of an IAT Data driven. The IAT concept materialised by an organic notion rather than a functional one, is necessary but no more sufficient to satisfy the process user needs.

According to these results, the PRIAM project aims to develop the IAM (Intelligent Actuation and Measurement) concept as integrated sub-system of whole of the process system.

The PRIAM way of working is based on the prototyping of the CMMS reference approach by both the user and the vendor in order to qualify and to quantify the IAM interchangeability and interoperability gap.
At this stage, the second rupture way is to promote an IAM application driven implementation independent as opposed to an IAM technology driven implementation dependent.

In additional **prenormative** view, the on-going European Intelligent Actuation and Measurement User Group (ESPRIT project n° 8244) extends this way of thinking and working to an independent and open forum which is composed of end users, IAT suppliers, engineering companies, scientific and which is linked with national and internal standard committees.

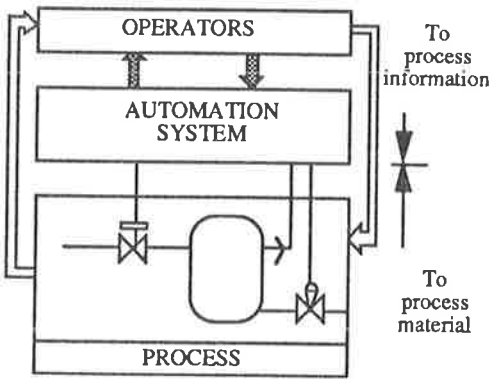


Fig. 1. The plant : Process, Automation System and Operators

2. IAT CONTEXT

Any plant (see Fig.1) is composed of the **process**, processing raw material to transform it into products, the **automation system** processing information from/to the process and the operators, the **operators** operating the process through the automation system.

From this structure emerges two interface types : on the one hand a **physical interface** which manages the exchange between process and automation system through the **actuators** and the **sensors** and on the other hand the **user interface** representing the semantic link between the operators and the automation system. The physical interface is composed of :

- the data coming from the sensors and used by the processing,
- the data coming from the actuator instrumentation and used by the processing,
- the information provided by the processing to the actuator command.

On another conceptual level, the user interface contains :

- the requests coming from the operators for the processing.
- the reports coming from the processing for the operators.

According to the content of the elementary data, the physical interface appears as the informational hinge of the plant. Indeed from a reliability and availability point of view, any communication fault through the physical interface implies bad effects on the whole of the production system. Therefore the actuators and sensors considered in theory as trivial elements, make in fact a real Achilles' heel of the plant. A first up-grading of this conventional equipment consisted in the definition of its technological dysfunctioning by the complementary specification of the waited and acceptable services of control. This has led on the **automation object** concept (Morel, 1993) to the development of the **behaviour filter** notion which corresponds to the using of expected behaviour model of each physical element between the control block and the input/output lines to separate application and technological points of view (see Fig. 2).

The automation object notion allowed on the one hand a higher component **reusability** since 80% of the applications are composed of the same basic equipment and on the other hand a higher **interoperability** by the standardisation of the interface syntax.

From this first behavioural modelling result, the ESPRIT-DIAS project had as objective to extend the control view towards the **maintenance** and **technical management** islands by the implementation of the C.M.M.S. integration concept (Galara, 1993) which is a sub-concept of C.I.M.E. (Computer Integrated Manufacturing Engineering). Therefore the basic automation object has evolved to a **intelligent** automation object : the **Intelligent Actuators and Sensors** representing a new way in the control part structuration (see Fig. 2).

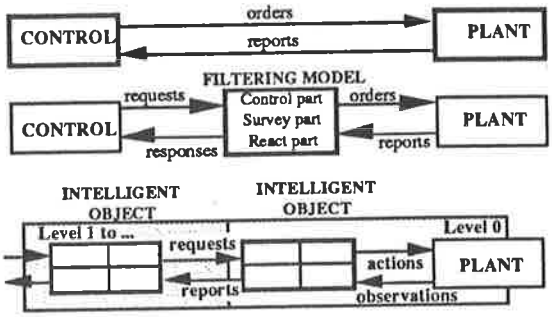


Fig. 2. Control part structuration evolution

Thanks to this intelligence distribution down to field devices (Intelligent Field Devices), the physical interface is not only composed of typical analog signal or current software data but also of pertinent, consistent and reliable **information**. The innovating IFD resulting of a KOD approach (Iung, 93) is necessary but not sufficient to completely satisfy the user requirements and particularly the automation system-operators interface.

3. IAM CONTEXT

In fact, the IAT are only equipment quantitatively distributed in the plant which have to be integrated qualitatively inside a real subsystem : the **Intelligent Actuation and Measurement** (Capetta, 1993).
Based on this concept, the ESPRIT-PRIAM project, logical continuation of DIAS, proposes some generic definitions in order to have a same mental image between the users and the vendors :

- Measurement is considered as a set of operations for the purpose of determining the value of a quantity.
- Actuation is also considered as set of operations for the purpose of determining the value of a quantity.

With these two definitions, there is apparently a paradox because measurement and actuation is defined with the same methodology. To avoid such a paradox, it is necessary to go deeper in the definition of "determining the value of quantity" :

- for measurement, the meaning ought to be : to be able "**to observe**" the value of a quantity,
- for actuation, the meaning ought to be : to be able "**to modify**" the value of a quantity.

Measurement and actuation are means of achieving intentions : to observe and to modify a value of a quantity. So, Intelligent Measurement and Actuation integrating IAT equipment have to be considered as subsystems at the two plant interfaces defining an **overlap** (see Fig. 3) taking into account :

- a part of the process : the **IAM machinery** which transforms the observation into a processable signal or the signal into a modification of the value of a flow of material,
- a part of the automation system : the **IAM processing capability** which transforms the signal into pieces of information or a piece of information into a signal.

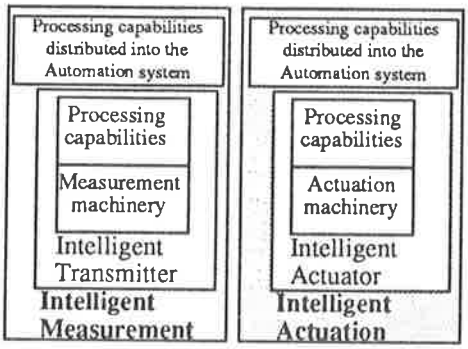


Fig. 3. Intelligent Actuators and Transmitters inside Intelligent Actuation and Measurement

The definition of this IAM is based on a specific life cycle which starting point is a **user need approach** in order to satisfy the user interface : moving from application-driven (the needs) up to technology-driven (the systems).
This functional and systemic approach, established by engineering companies, is entirely **independent** of all implementation of the processing and leads in a first step to the description of the Functional Requirement Diagrams (see Fig. 4) : independent of any implementation into any Automation System.

Implementation independent does not mean completely generic. Indeed the FRD is linked on the one hand to its **environment**, on the other hand to the process which will be controlled and observed by the available actuators and sensors (actions-observations representing the machinery) and finally to the information needs materialised by the control, maintenance and technical management **requests** and **reports**. These requests-reports are normally expressed by a set of **agents** representing the operators considered as End-users and suppliers of IAM devices.

In relation to its global interface, the FRD behaviour is defined by the **rules** which are necessary to implement from the Requests in order to provide the Reports according to the Actions and Observations. This behaviour or processing has to take into account the expected **performances** and the functioning **safety** constraints and contains the **logical, sequential** and **analog** operations.

The processing will later be distributed inside the equipment based on conventional or digital technologies. The distribution established for suppliers and integrators, corresponds to a **technology-driven** dependent of the implementation and represents the Functional Diagrams.

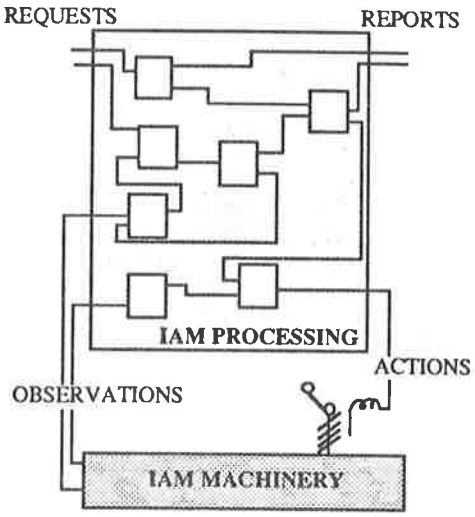


Fig. 4. IAM Functional Requirement Diagrams

The **rupture** between the two FRD and FD islands (see Fig. 5) is situated between the world of Engineering and the world of integrators and suppliers and is made to move from FRD, application driven and implementation independent, up to FD, technology driven and implementation dependent (Morel, 1994).

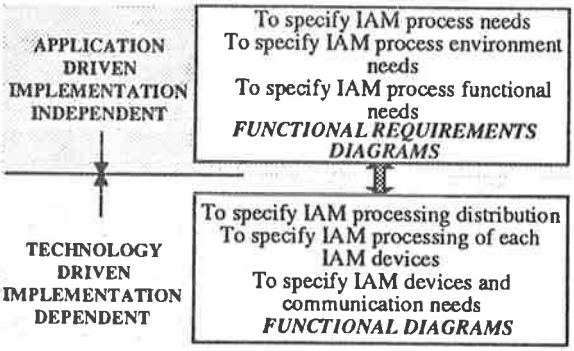


Fig. 5. Functional Requirement Diagrams and Functional Diagrams

4. - FUNCTIONAL REQUIREMENT DIAGRAMS OF IAM

Outside the implementation world, the IAM being immersed to a specific context, the new systemic way of working implies before

defining the FRD, to identify the **integration constraints** which will be taken into account. These constraints concern more precisely the **process needs** in terms of IAM machinery characteristics (type of fluid, ...) and the **environment needs** in terms of set of mecano-climatic characteristics (harsh environment, ...). From the environmental context and according to the objective assigned to the IAM system, the approach consists **in structuring** in a chronological way the user specification described in natural language to identify the IAM process functions with control, maintenance and technical management points of view. The structuration pertinence towards autonomous process functions has to ensure the function perennity and to facilitate its reusability. According to its genericity degree, a function can be **standardised** or specific and stored in library usable for other structurations. Each function is an independent module considered as a "lego" : the system resulting of the "lego" assembling assimilable to an electronic card composed of standard or specific components.
The gathering of the whole IAM functions leads to the description of the physical (actions-observations) and user (requests-reports) interfaces.

At a more detailed level and respecting the same concepts each function which composes an IAM FRD, has to be formalised in a language, **function bloc job-oriented** (graphical and textual) always implementation independent. This allows to introduce **functional behaviour** of the IAM and the **performances** of the IAM (response time, availability, safety). In a reusability point of view, the language is composed of several Elementary Functional Blocks or Macro Functional Blocks (see Fig. 6) which is defined by the assembling of EFB and associated to a smaller degree of genericity as the EFB. The EFB corresponds to a particular operation which cannot be modified (AND, OR, +, -, COSINE, ...).

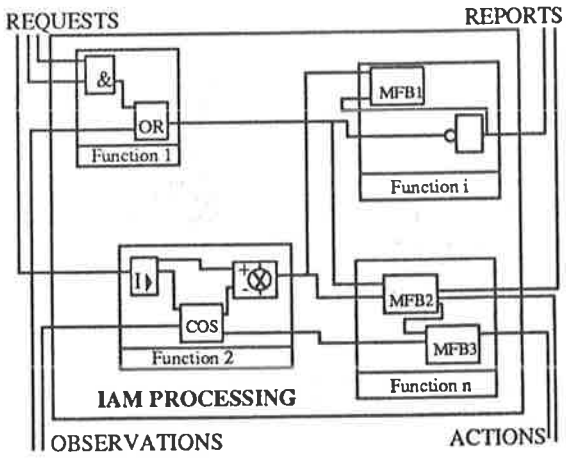


Fig. 6. Detailed FRD : Functions, EFB and MBF

Each EFB and MFD is functionally prototyped, validated and certified by the association of a dynamic to the behaviour (C language, GRAFCET, ...) then stored in a library (Morel, 1993). The simulation are made by the use of **test scenarios** in open loop (interaction with the operator) or in closed loop coupling together the element to a simulation of a part or the whole of the machinery.

By recursivity, the dynamic of the EFB and MFB also allows to validate the process functions. The tests realised at this step could be used as reference for the processing validation after the distribution.
Therefore the FRD approach is based on a top-down description (from function to MFB or EFB) or on a bottom-up description (from EFB to MFB and function) using a same function block language ensuring on the one hand a specification homogenisation for the integrality of the processing and interfaces and on the other hand the taking into account of the whole of the FRD up-gradings (library flexibility).

5. FUNCTIONAL DIAGRAMS OF IAM

This homogenisation has to be conserved during the **automatic translation** for the **processing distribution** into supplier systems implying to **harmonise** the FRD and FD function block language. The way from FRD to FD has to be an **added value** of the none distributed functions : to avoid the incompatibility between the two phases.
Before the distribution **technology dependent**, it is necessary to define the FD supports (IAT, Maintenance System, Process Control System) in relation to a distributed control architecture (Iung, 1993) which appears the most suitable to the FRD needs, to its performances and to the mandatory safety constraints.

After this, the FRD are splited and implemented into the different FD systems consecutively to a distribution based on specific rules which require processing and information **additions** in order to take into account the building constraints of the distributed process control (redundancy, initialisations, communication ...).

Indeed from a communication point of view, this operation introduces semantic and syntactic **links** between the different systems materialised by an information exchange. These application links are the Functional Companion Standards (See Fig. 7.) representing the gathering of each function interface inside each system but also a **common basis** of function determining a minimum degree of intelligence. The FCS are independent of the type of communication system that will support these exchanges of application data.

Moreover the FCS must be standardised to guarantee the interchangeability and

interoperability of the systems to achieve IAM **interworkability** (Morel, 1993). The interoperability allows to connect any system in a unique IAM objective whereas the interchangeability allows to replace a system by another without behaviour change.
So whatever the distribution is, the FCS syntax and semantic (function and information) and the mechanism of the FCS assembling has to ensure the standardised definition of the whole of the system interfaces.

The physical interconnection of the different subsystems can be realised by a **fieldbus** (see Fig. 7) implying after the definition of the Communication Companion Standards (adaptation of the FCS to the communication constraints) to describe a specific network interface for each support allowing to make the link between CCS and specific bus data.

From this global implementation architecture and using the same dynamic principles as for the FRD, the FD and the communication support can also be validated in terms of general behaviour, distributed behaviour, performances, synchronism, ... formal validation thanks to synchronous languages (André, 1993) added to the block language.

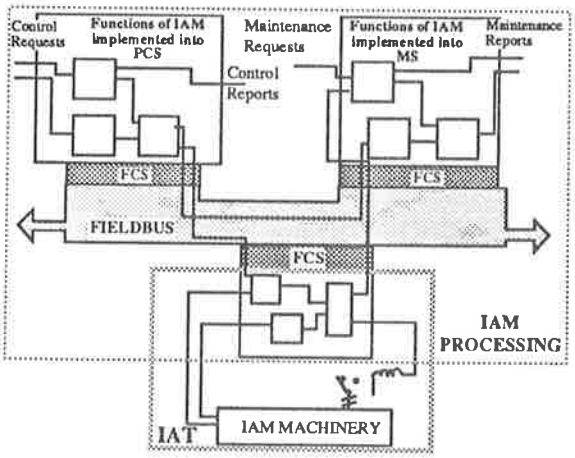


Fig. 7. Distribution and FCS

6. CONCLUSION

In order to develop this way of working and thinking from a scientific to a pragmatic point of view, it is necessary first to prototype a **tool** supporting the whole of the methodology : FRD, FD, distribution, ... (link with the PRIAM tool), second to work to the **homogenisation** between FRD and FD languages to specify and to implement the behaviour of each IAM into the systems from the FRD specifications (link with the IEC 1131) and finally to propose a **standardisation** on the FCS in terms of functions and information (link with the EIAMUG project).

7. ACKNOWLEDGEMENTS

This paper has been prepared by the authors :

D. GALARA : **DIAS** Expert Team Coordinator, **PRIAM** Expert Team Coordinator, **EIAMUG** end-user expert.

B. IUNG : **DIAS** sub-contractor, **PRIAM** Engineering Team Technical Deputy, **EIAMUG** scientific expert.

G. MOREL : **DIAS** Technical Consultant, **PRIAM** Expert Team Technical Deputy, **EIAMUG** scientific expert.

F. RUSSO : **DIAS** end-user expert, **PRIAM** Standardisation Team Coordinator, **EIAMUG** Chairman.

on behalf of the PRIAM consortium.

8. ACRONYMS

CCS	Communication Companion Standard
CMMS	Control Maintenance and technical Management System
DIAS	Distributed Intelligent Actuators and Sensors
EFB	Elementary Functional Block
EIAMUG	European Intelligent Actuation and Measurement User Group
ESPRIT	European Strategic Program for Research and development in Information Technology
FCS	Functional Companion Standard
FD	Functional Diagrams
FRD	Functional Requirement Diagrams
IAM	Intelligent Actuation and Measurement
IAT	Intelligent Actuators and Transmitters
IFD	Intelligent Field Devices
KOD	Knowledge Oriented Design
MFB	Macro Functional Block
MS	Maintenance System
PCS	Process Control System
PRIAM	Prenormative Requirements for Intelligent Actuation and Measurement

9. REFERENCES

André Charles, M.A. Péraldi (1993). Synchronous programming: Introduction and Application to industrial process control. In: *Proceedings of the 7th COMPEURO'93-IEEE conference*, ISBN 0-8186-4030-8, pp 461-470, Paris, May 24-27 1993.

Capetta L., D. Galara, J. Graefer, G.P. Lovischek, L. Mondeil and M. Sanguinetti (1993). From current actuators and transmitters towards IAM - PRIAM approach. In: *Proceedings of Automation 1993-BIAS* , pp1171-1186, Milan, November 23-25 1993.

Galara D. , J.M. Favennec, B. Iung and G. Morel (1993). Distributed Intelligent Actuators and Sensors. In: *Proceedings of the 7th COMPEURO'93-IEEE conference*, ISBN 0-8186-4030-8, pp 79-85, Paris, May 24-27 1993.

Hammer M., J. Champy (1993). Reengineering the corporation : a Manifesto for Business Revolution. *Harper Collins Publishers*, Inc, New York, 1993.

Iung B., P. Lhoste, G. Morel and M. Roesch (1991). Functional Modelling of an Intelligent Actuator : applicable to an ON/OFF or modulating electrical valve. In: *Proceedings of Workshop ESPRIT-CIM*; Athens 20-21 June 1991.

Iung B., J.F. Petin and G. Morel (1993). Development and Integration of intelligent actuator in a manufacturing system. In : *Proceedings of International Conference on Advanced Mechatronics ICAM'93-JSME*, pp 191-196, Tokyo, August 2-4, 1993.

Morel G., B. Iung, D. Galara and F. Russo (1994). Prototyping a sub-concept of computer integrated manufacturing engineering : the integrated control, maintenance and technical management system. In: *Proceedings of ISRAM'94*, Wailea-Maui, USA, August 18 1994.

Morel G., P. Lhoste, B. Iung, J.F. Petin, F. Corbier and O. Douchin (1993). Discrete Event Automation Engineering : outline for the PRIAM project. In: *Proceedings of Automation 1993 - BIAS*, pp1105-1117, Milan, November 23-25 1993.

backwards in time (in terms of the pattern) to see if the sensor data matches the early parts of the reference pattern. This allows us to detect data patterns that gradually builds up to fit the reference pattern over time. Figure 3 illustrates this procedure.

Defuzzifier Transition

The defuzzifier transition is used to defuzzify recommendations for actions deduced from a fuzzy rule base. Fuzzy assertions are converted into crisp control output values through defuzzifer transitions. Defuzzifiers must have at least one fuzzy-recommendation and one fuzzy-action place connected to it, in order to operate properly. A special case occurs when $x_i(t)$ is a negative number. In order for the CFPN to provide a concise method for representing control actions such as "*set valve to not low*", as well as reconciling the incorporation of a measure of disbelief, negative certainty factors were used to model this behavior. We propose that when a defuzzifier transition receives a recommendation with a negative certainty, the formula $1 - \mu_F$ should be used in place of the normal membership function μ_F defined for an input fuzzy-recommendation. This provides us with a logical and consistent method for extending the fuzzy logic paradigm to include -1 to 1 logic. Figure 1 illustrates this procedure.

5 Example

This example illustrates how one might develop an application from input sensor values to control output, by combining various logic elements of the CFPN approach to form a fuzzy rule-base. Referring to figure4, A simple linear mapping of the temperature values to a certainty factor is done by each of these two fuzzifier transitions. The **N-fuzzifier** transition attempts to match the pressure sensor reading to a reference pattern which varies as (0,1,2,3,4,5). Each of the **recommendation** places, *power-low* and *power-high* contains a membership function which determines what is considered a low and high power setting respectively. The **DISPLAY-PANEL** shows the temperature and pressure sensor readings, along with their corresponding certainty values for each fuzzy assertion. The last graph in this example shows the control output (*power setting*) produced by this fuzzy rule-base.

6 Conclusions

In this paper, the theoretcial background, as well as a description of Continuous Fuzzy Petri Nets was given. A time based pattern matching algorithm for fuzzification is also given. In addition, negative certainty values in fuzzy logic has been introduced as well as a method for handling negative certainty values in the defuzzification process. The Continuous Fuzzy Petri Net has been implemented using the G2 real-time expert system development package. ESSO Canada Ltd. has adopted the CFPN approach and has integrated it successfully with their refinery process monitoring system in Sarnia, Ontario, Canada. The Continuous Fuzzy Petri Nets approach is a new direction in Petri Net development. It combines the flexibility of fuzzy logic and the graphical nature of Petri Nets to form a tool useful for monitoring, diagnosis, decision support and intelligent control.

References

[Årzén 93] Årzén, K-E. (1993). *Grafcet for intelligent real-time systems*, In *Proceedings of 12th IFAC World Congress*, Vol.4, pp.497-500, Sydney, Australia.

[Al-Jaar *et al.* 90] Al-Jaar, R.Y. and Desrochers, A.A. (1990). *Petri Nets in Automation and Manufacturing*, (G.N.Saradis Ed.), In *Advances in Automation and Robotics: Knowledge-Based Systems for Intelligent Automation*, Vol.2, pp.153-225, Greenwich, Connecticut, JAI Press Inc.

[Brand *et al.* 88] Brand, K.P. and Kopainsky, P. (1988). *Principles and engineering of process control with petri Nets*, In *IEEE Transactions on Automatic Control*, Vol.33 No.2, pp.138-149.

[Brauer *et al.* 87] Brauer, W., Reisig, W. and Rozenberg, G. (eds), *Advances in Petri Nets 1986*, (Parts I and II), *Petri Nets: Central Models and Their Properties* and *Petri Nets: Applications and Relationships to Other Models of Concurrency. Proceedings of an Advanced Course*, Bad Honnef, Germany, Sept 8-19, 186. Lecture Notes in Computer Science, Vol.254 and 255, Springer-Verlag, New York.

[Card. *et al.* 90] Cardoso, J., Valette, R. and Dubois, D. (1990). *Petri nets with un-*

certain markings, (G.Rozenberg Ed.), In *Advances in Petri Nets*, Lecture Notes in Computer Science, Vol.483, Springer-Verlag, pp.65-78.

[Chen *et al.* 90] Chen, S.M., Ke, J.S. and Chang, J.F. (1990). *Knowledge Representation Using Fuzzy Petri Nets*, In *IEEE Transactions on Knowledge and Data Engineering*, Vol.2, No.3.

[Gensym 92] Gensym Corp. (1992). *G2 Reference Manual*, G2 Version 3.0, Gensym Corp., 125 CambridgePark Drive, Cambridge MA 02140, U.S.A.

[Jensen 81] Jensen, K. (1981). *Coloured Petri Nets and the invariant method*, In *Theoretical Computer Science*, Vol.14, pp.317-336, Springer-Verlag.

[Looney 88] Looney, C.G. (1988). *Fuzzy Petri Nets for Rule-Based Decision making*, In *IEEE Trans. on Systems, Man, and Cybernetics*, Vol.18, No.1, pp.178-183.

[Natkin 80] Natkin, S.O. (1980). *Les reseaux de Petri stochastiques et leur application a l'evaualtion des systems informatiques*, These de Docteur-Ingenieur, Conservatoire National des Arts et Metiers (CNAM), Paris.

[Petri 62] Petri, C.A. (1962). *Kommunikation mit Automaten* , Ph.D. dissertation, Uniniversity of Bonn, Bonn, West Germany.

[Rama. 73] Ramachandani, C. (1973). *Analysis of asynchronous concurrent systems by timed Petri Nets*, Ph.D. dissertation, Dept. of Electrical Engineering, MIT, Cambridge, MA.

[Reisig 87] Reisig, W. (1987). *Petri Nets in Software Engineering*, (W.Brauer, W.Reisig and G.Rozenberg, Eds.), In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Springer-Verlag, New York, pp.63-96.

[Zadeh 65] Zadeh, L.A. (1965). *Fuzzy Sets*, In *Information and Control*, V.8, pp.338-353.

[Zadeh 75] Zadeh, L.A. (1975). *The concept of a linguistic variable and its applications in approximate reasoning - Part 1,2,3*, In information Sciences, Vol.8, pp.199-249,pp.301-357 and Vol.9, pp.43-80.

[Zhou *et al.* 90] Zhou, M.C., DiCesare, F. and Rudolph, D. (1990). *Control of a flexible manufacturing system using petri nets*, In *1990 IFAC Congress*, Vol.9, pp.43-48, Tallinn, USSR.

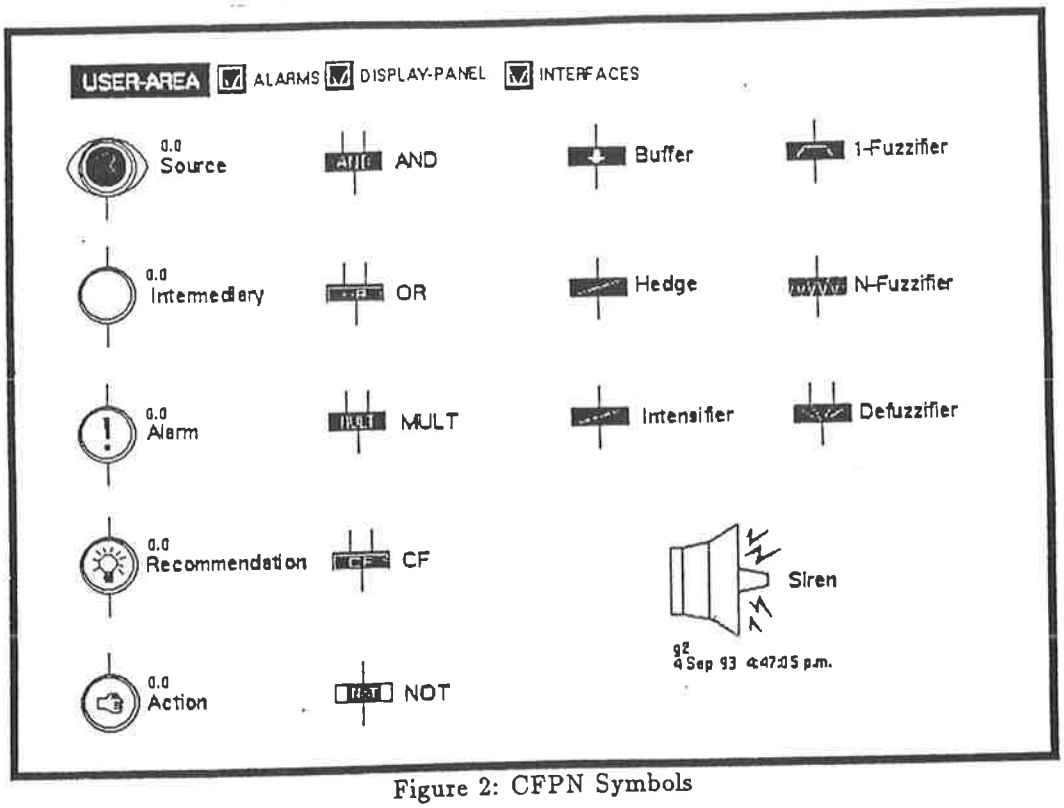


Figure 2: CFPN Symbols

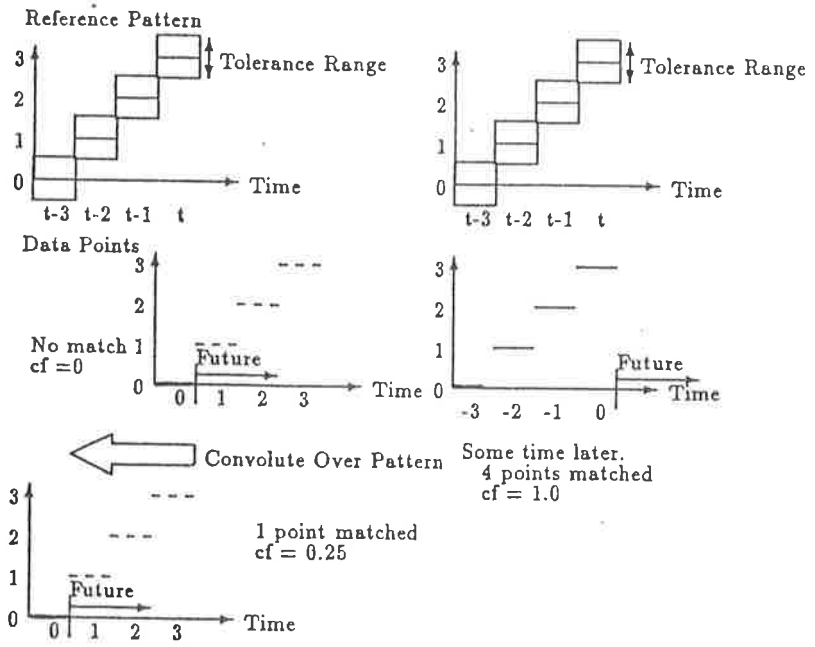


Figure 3: N-Fuzzifier Pattern Matching Procedure

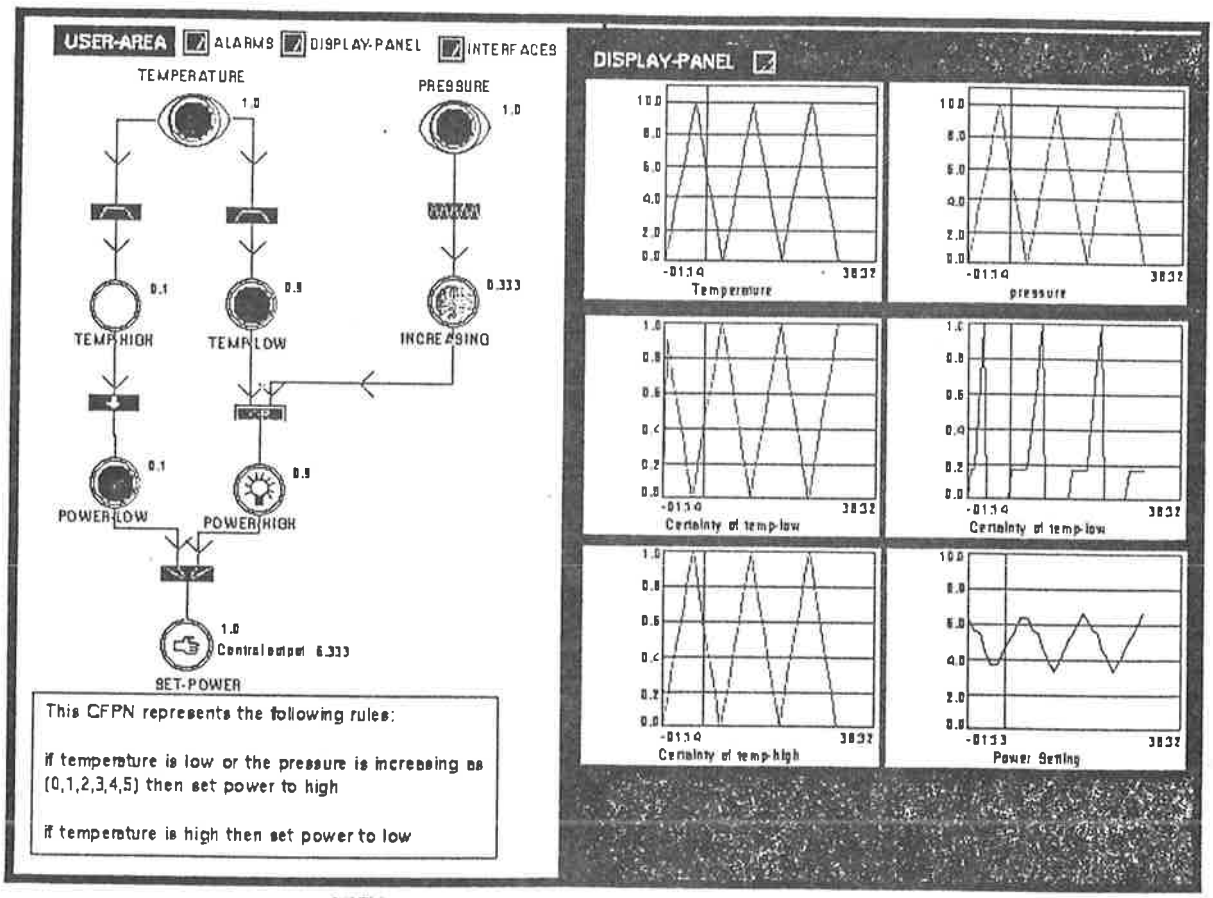


Figure 4: A Simple Control Example

PARAMETERIZED HIGH-LEVEL GRAFCET FOR STRUCTURING REAL-TIME KBS APPLICATIONS

Karl-Erik Årzén

Department of Automatic Control,
Lund Institute of Technology,
Box 118, S-221 00 Lund, Sweden,
Email: karlerik@control.lth.se

Abstract: Grafcet is suggested as a way of structuring on-line rule-based systems supervising sequential processes or implementing sequential reasoning procedures. Grafchart, a G2-based Grafcet toolbox has been developed. It has been applied in an on-line refinery application. High-Level Grafchart allows parameterization and sequence nets with multiple tokens.

1. INTRODUCTION

Supervisory control applications such as set-point control, monitoring, fault detection, diagnosis, scheduling, planning, and production optimization receive increasing attention from both the academic control community and the industry. One reason for this is the increased demands on performance, flexibility, and safety caused by increased quality awareness, environmental regulations, and customer-driven production.

There is a large industrial interest in applying AI techniques, in particular expert systems or knowledge-based systems (KBSs), to supervisory control problems, [Årzén, 1991a]. In the majority of the applications the systems are used as operator support systems. The motivation for using AI techniques is the need to utilize heuristic knowledge and/or models of a more qualitative or functional nature than what is given by purely mathematical models.

Rules are commonly used in knowledge-based systems to express domain knowledge. A problem with rule-based system is representation of, and reasoning about, sequential processes. For sequential processes it is often only a part of the total number of rules that is applicable at any given time. As the sequential process evolves there is a need to change the rules that apply, i.e., change the context of the rule-based system. The traditional solution to this is to use a context clause as an additional conjunction condition in the rule antecedents. The condition of the rules may only become fulfilled if the system is in the correct context. Certain rules are devoted to the changing the context, i.e. advancing the rule-based system into the next step of the sequence. A problem with this solution is that the natural sequential nature of the problem is hidden away among the rule antecedents and consequents making the total system difficult to develop and maintain.

The problems with sequential processes show up in two different situations. The processes in the process industry are typically of a combined continuous and sequential nature. All processes run in different operating modes. In the simplest case these can consist of start-up, operation, and production. In the different modes the process and its components may function differently. This means, e.g., that a rule-based monitoring and diagnosis system must contain different rules for the different operating modes of the process. As the process changes its operating mode the rule-based system must change its set of active rules.

The second situation concerns the case when the problem that the rule-based systems should solve itself can be decomposed into sequential steps. Assume that we want to implement an on-line production optimization system. The system should at regular time intervals perform measurement and production analysis, calculate new optimized parameter settings, and execute the parameter changes. Although the process may operate in the same mode all the time, this problem is still of sequential nature.

During the last years Grafcet has emerged as a standard for representing and executing sequence control logic at local control level in PLC systems and process control systems. Originally a French standard, Grafcet has been accepted as an international standard (IEC 848), now under the name Sequential Function Charts (SFC). SFC is also an essential part of the recently released standard for programming languages for PLC systems (IEC 1131-3).

The solution to the problem of sequential rule-based systems proposed in this paper is to use Grafcet to structure supervisory rule-based applications in the same way at it it currently used at the local control level. Grafchart [Årzén, 1991b, Årzén, 1993, ?] is a toolbox that combines the network formalism of Grafcet with real-time knowledge-based systems. Grafchart is implemented in G2 from Gensym Corp. G2 is an object-oriented graphical programming environment primarily aimed at intelligent supervisory control applications. Grafchart has been developed at Lund Institute of Technology and is commercially available.

Grafcet is based on Petri Nets [Reisig, 1985]. It can in fact be viewed as a special class of Petri Nets. Parallel to the development of Grafcet, High-Level Petri Nets [Jensen and Rozenberg, 1991] have been developed from ordinary Petri Nets. In Grafcet, an active step is indicated by a token that is placed at the step. The token contains no information in itself, it is imply a boolean indicator. In High-Level Petri nets the tokens are represented by abstract data types, i.e. they carry information.

High-level Grafchart is an extension of Grafchart that is based on High-Level Petri nets and ideas from object-oriented programming. It is based on the fact that all elements in Grafchart, e.g., steps, transitions, tokens, etc., are objects which are defined in class definitions, can have attributes, and can be specialized.

Section 2 describes Grafchart. An industrial oil refinery application of Grafchart is briefly overviewed in Section 3. Section 4 describes some of the features of High-Level Grafchart.

2. GRAFCHART

In Grafcet, [David and Alla, 1992], a sequence control problem is represented in terms of steps and transitions. A step represents a state, phase, or mode. A step can be active or inactive. The former is indicated by a token that is placed at the step. Associated with the step are actions that are executed when the step is active. In Grafchart instead a set of rules is associated with the step. A transition interconnecting two steps contains a transition condition. A transition is active when

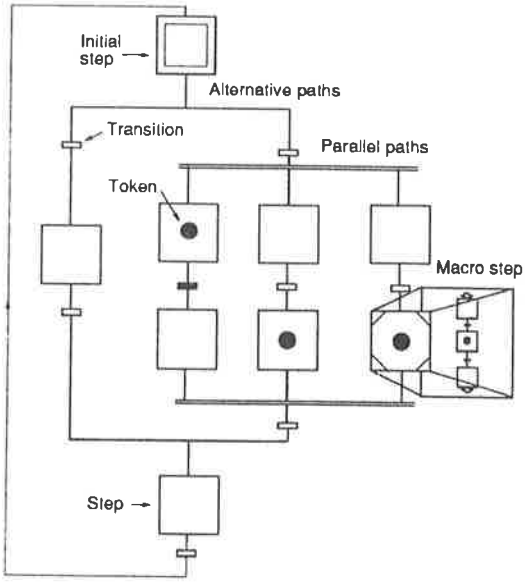


Figure 1. Grafcet example

the preceding step is active. When the transition condition becomes active the step preceding the transition is deactivated and the step succeeding the transition is activated. Grafcet also supports alternative branches, parallel branches, and macro steps, i.e. hierarchical steps that themselves contain steps and transitions. The graphical syntax of Grafcet is summarized in Fig. 1.

The implementation of the toolbox is based on the G2 concept of *activatable subworkspaces*. A workspace is a virtual, rectangular window upon which various G2 items such as rules, procedures, objects, displays, and interaction buttons can be placed. A workspace can also be attached to an object. In this case the workspace is called a subworkspace of that object. When a subworkspace is deactivated all the items on the workspace are inactive and "invisible" to the G2 inference engine. This means, e.g., that rules placed on a deactivated subworkspace cannot be invoked.

Steps: A step is represented by an object that has an activatable subworkspace. Initial steps are represented by a special initial step object, see Fig. 1. Initial steps are automatically activated when execution starts. An active step is indicated by a dynamically created token object that is placed on the step.

The rules that should be active when the step is active are placed on the subworkspace of the step. The subworkspace is only active when the step is active, i.e., the rules are only executed when the step is active. The rules can be invoked by forward or backward chaining or by associating a scan interval with the rules. Rules can also be specified to be invoked only once when the step becomes active.

The full G2 rule syntax can be used in Grafchart. However, in order to further simplify for the user

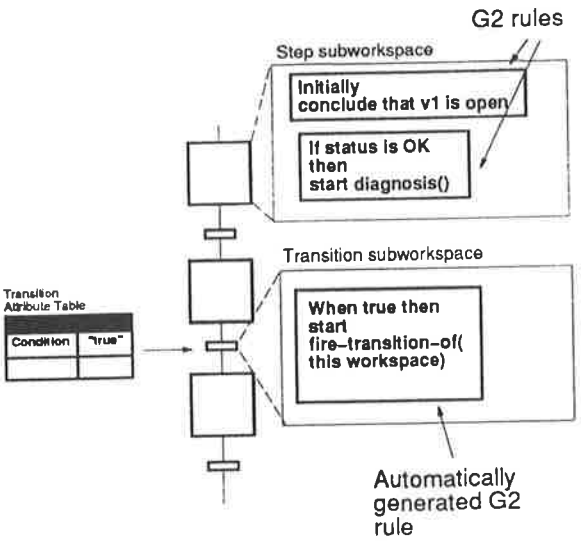


Figure 2. Step and transitions with their subworkspaces

the somewhat annoying natural language syntax of G2 expressions has been replaced by a Pascal-like dot-notation. That means that instead of referring to an object attribute using the standard G2 syntax `the attribute1 of object1` the shorter `object1.attribute1` is used.

Transitions: A transition is represented by an object with an activatable subworkspace. Each transition contains an attribute named "condition". Here, the user enters the logical condition for when the transition should fire expressed as a text string. This string may contain the previously mentioned dot-notation. During initialization the condition attribute is used to automatically generate the rule that executes the transition firing. This rule is located on the subworkspace of the transition. In this way the rule will only be tested when the transition is active, i.e., when the steps preceding the transition is active. The situation is shown in Fig. 2.

Macro steps: Macro steps are used to represent steps that have an internal structure of (sub)steps, transitions and macro steps. The internal structure is placed on the subworkspace of the macro step, see Fig 1. Special enter-step and exit-step objects are used to indicate the first and the last substep of a macro step. A macro step must have exactly one enter-step and at least one exit-step. When the transition preceding a macro step becomes true the enter-step upon the subworkspace of the macro step and all the transitions following that enter-step are activated. The transitions following a macro step will not become active until the execution of the macro step has reached an exit-step.

In addition to steps, transitions, and other macro steps, a macro step can itself also contain rules.

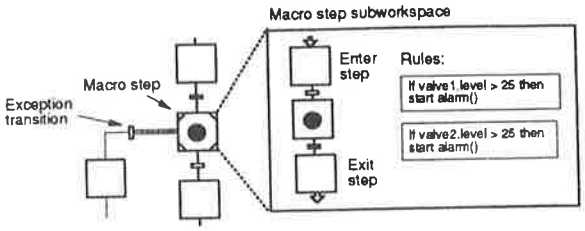


Figure 3. A macro step

These rules will be active all the time while the macro step is active, see Fig. 3.

Exception Transitions: An exception transition is a special type of transition that only may be connected to a macro step. An ordinary transition connected after a macro step will not become active until the execution has reached an exit step. An exception transition is active all the time that the macro step is active. If the exception transition condition becomes true while the corresponding macro step is executing the execution will be aborted and the step following the exception transition will become active. This also applies recursively, i.e., if the macro step contains other macro steps that are currently active, these will also be aborted. An exception transition is shown in Fig. 3. Exception transitions are an addition to Grafcet that was first proposed in [Årzén, 1991b]. It has proved very useful for implementing error handling.

3. AN INDUSTRIAL EXAMPLE

Grafchart has been used to implement a knowledge-based system that generates on-line advice for operators regarding the distribution of hydrogen resources at the Star Enterprise Delaware City Refinery [Petti and Dhurjati, 1992]. The system uses KBS techniques coupled with numerical optimization. The application is an example of the second situation where sequential processes are important. The specific problem that is solved is to meet the needs of the hydrogen consuming units in the refinery while minimizing the hydrogen that is wasted. A catalytic reformer unit and a continuous catalytic reformer unit produce hydrogen as by-products. A hydrocracker unit consumes high purity hydrogen and vents low purity hydrogen. Hydrogen from these units is used to satisfy the needs of the hydrogen consuming hydrotreaters, sulphur recovery, methanol, and naphtalene units. Any additional hydrogen needs must be met by a hydrogen production unit. The solution to the problem consists of three sequential steps [Petti and Dhurjati, 1992, Petti, 1992].

- Attempt to recover any suction venting associated with the compressors.
- Determine the best operating policy based on

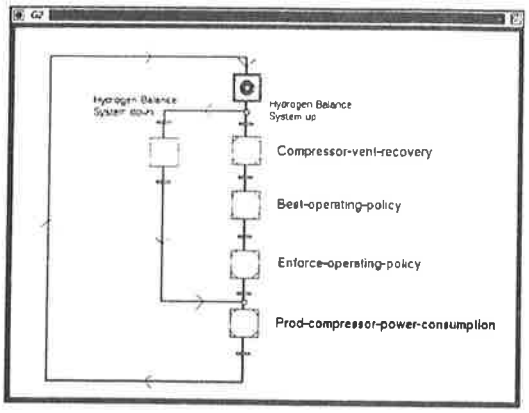


Figure 4. Hydrogen application Grafchart

the current operating state.

- Generate advice for the operators on adjustments necessary to enforce the optimal operating philosophy.

Each of these steps can be broken down into substeps. The first and the third steps are solved by heuristic rules. The second step employs linear programming to solve a numerical optimization problem formulated by the KBS. In the case a numerical solution cannot be achieved heuristic rules are used as a backup.

The sequential steps of the problem are represented as Grafchart macro steps according to Fig. 4. The entire system contains more than 18 macro steps, and over 80 ordinary steps. The reasoning path is traced by changing the colour of a step that has been active. The trace colours are reset each time a new cycle of the main Grafchart in Fig. 4 is started. The main reasoning cycle is executed once every 2 minutes. The program formulates suggested flow rates at two splits in the hydrogen network and also recommends compressor settings (or loadings) for all of the compressors in the network.

The application is currently running on a Sun Sparcstation that is linked to a Foxboro IA process control system. The experiences of the system [Petti and Dhurjati, 1992] are that

> *"the use of the Grafcet toolbox is very helpful from both a development and operational standpoint. The various functions within the knowledge base are built in a straightforward fashion by associating rules, formulas, and procedures with Grafcet objects. This makes revision and maintenance of the system possible. Additionally, review of the reasoning path through the network is a clear indication of how the KBS arrived at its advice."*

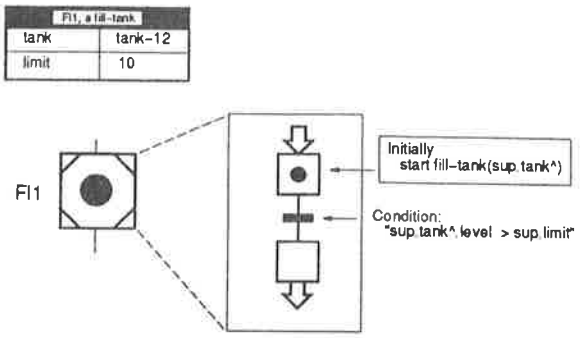


Figure 5. Parameterization

4. HIGH-LEVEL GRAFCHART

High-Level Grafchart is an extension of Grafchart that is based on ideas from High-Level Petri nets and object-oriented programming. The implementation of Grafchart is object-oriented. Steps, transitions, tokens, macro steps, and even entire Grafcet networks are represented by objects that are defined in a class hierarchy. High-Level Grafchart is based on the possibility to specialize these object classes and thereby adding attributes and refining their behaviour.

4.1 *Parameterization*

Ordinary Grafchart has no means for parameterization of steps, transitions or macro steps. The rules within a step are specific, i.e., they contains references to global variables and objects. This makes it difficult to reuse steps from one application to another. In High-Level Grafchart this is resolved by utilizing the fact that step, transitions and macro steps are objects defined in class definitions. The user can specialize these classes to subclasses in which additional attributes have been added. These attributes act as parameters which can be referenced from rules within the step using simple dot notation. By instantiating the step subclasses with different values of the parameters the step can be reused.

Consider the example shown in Fig. 5. The `fill-tank` class is a specialization of a macro step with two new attributes: `tank` and `limit`. `Fl1` is an instance of `fill-tank`. A fill-tank macro step contains the logic for the control and monitoring of the filling of a tank. The `tank` attribute contains the name of the tank that should be filled, i.e., the value of this attribute acts as a name reference. The `limit` attribute contains the limit up to which the tank should be filled. The macro step contains an enter-step that contains a rule that initiates the filling. This rule refers to the value of the tank attribute using the `sup.tank` notation (sup is short for superior). It refers to the tank referenced by the tank attribute using the Pascal-style notation `sup.tank^`. Similarly the transition condition upon the workspace refers to the level

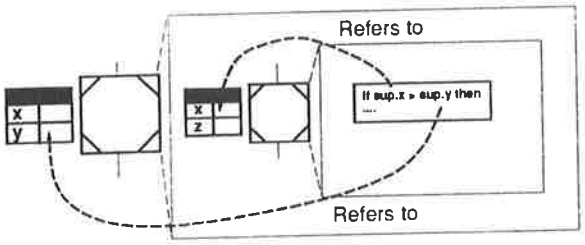


Figure 6. Lexical scoping

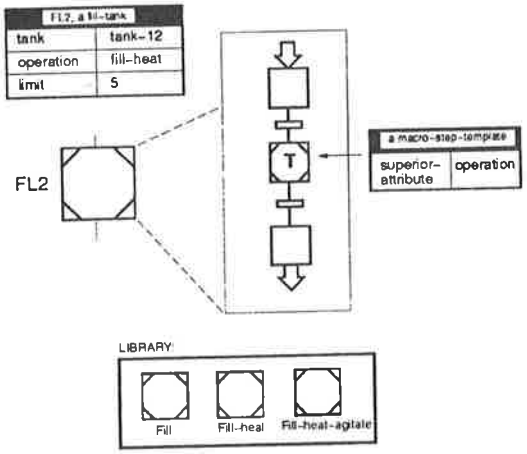


Figure 7. Procedure parameterization

of the tank referenced by the tank attribute and to the value of the limit attribute.

The `sup.attribute` notation is translated and replaced by a corresponding G2 expression during initialization. Macro steps can be hierarchically nested to any depth. Lexical scoping is used according to Fig. 6

4.2 *Procedure parameterization*

With procedure parameterization it is possible to have entire steps or macro steps as parameters to, e.g., another macro step or an entire Grafcet network. In the macro step the step is represented by a template step object that contains a reference to an attribute of the Grafcet network object. During initialization or execution of the network, the template step object is replaced by an instance of the class denoted by the attribute value, i.e., by some step or macro step class.

Consider the small example shown in Fig. 7. FL2 is an instance of a macro step subclass with three attributes. It contains a special macro step template object that contains one pre-defined attribute, `superior-attribute`. The value of this attribute is operation, the name of one of the attributes of FL2. During initialization, the macro step template will be substituted by a copy of the macro step referenced by the operation attribute. In this case the value of `operation` is `fill-heat`. Assume that there exists a library that contains three different macro steps: `fill`, `fill-heat`, and `fill-heat-agitate` which contains different operations that may be performed on a tank. The three macro steps can have totally different internal structure. With the procedure parameterization described, FL2 can be parameterized both with respect to which tank object it should operate on, what the level limit should be, and which operation that should be performed on the tank.

Procedure parameterization is similar to the possibility to have procedures as arguments to procedures in ordinary programming languages. Templates are available for steps, transitions, and macro steps.

4.3 *Token objects*

It is also possible to specialize tokens. In this case the tokens of a Grafcet network correspond to objects that move around in the network. Multiple tokens of the same or of different classes may coexist in the same network. The transition conditions can refer to the token object attributes using simple dot notation. The attributes of the token objects can be compared to the arguments and the local variables of a procedure in an ordinary programming language. The rules in a step can refer to and modify the token object attributes. They can also create and delete token objects. With this extension a behaviour corresponding to that of Petri nets with individual tokens is achieved.

The rules and transition conditions can refer to the attributes of a token object. This is done with the `inv.attribute` notation (inv is short for invocation). Token objects can be used in different ways. If the rules and the transition conditions only refer to the token attributes the different token objects will correspond to different independent procedure invocations. This corresponds to reentrant procedures in a ordinary programming language.

It is also possible to use token objects to model different types of ordered elements that are not allowed to pass each other, e.g., messages in a FIFO queue or machine parts moving on a manufacturing conveyor belt.

Finally, it is possible to have entire Grafcet networks as attributes of token objects. This gives a hierarchical structure where the tokens in a Grafcet network themselves may contain Grafcet networks. It is possible to start and stop the execution of a Grafcet network of a token object by step rules. It is also possible to test whether a certain step of a Grafcet network of a token object is activated or not in a transition condition.

5. CONCLUSIONS

Hardware and software integration are important in order for knowledge-based techniques to be ac-

3.2 Basic Structural Specification

A basic specification has been designed for each *work-station*. For this purpose, some fundamental properties of control systems like *controllability* and *observability* have been kept in mind. Control parameters are mostly input data and observation variables are output information.
The data are structured according to the Input/Output approach: the work-station receives a set of input data, simulates the task made at this position and gives as results a set of output data.
These latter constitute at their turn the set of input data of the next work-station through the relation. We can find several types of data (parameters or constants): some relating to the *process* and the others which are specific to the *product*.
As for the relations, we can also find the same distinction between process relations and the product ones (the tube). So the relationship between two sequential work-stations is summarized by the equality below:

$$ProductOutputParameter[Work-station(i)] = ProductInputParameter[Work-station(i+1)]$$

3.3 The Quantitative-Qualitative Model

3.3.1 The Quantitative Model —When numerical relations are available, it is konwn that they are the best representation. Most of them are simple algebraic relations. The others are (generally first order) differential equations. The specification of those relations are not immediate. They need sometimes a long time to be established. So they are specified at multiple stages. Some numerical variables that have to be controlled are the *rotation speed of the tube* **vrot** and its *temperature* tg. In fact, the quality of the coloured phosphor layer (Green, Blue or Red) will largely depend on these variables.

3.3.2 The Qualitative Model —The introduction of qualitative relations is based on the two main following reasons:

- the use of qualitative models can be viewed as a first step in obtaining more detailed knowledge. Reliable mathematical models need more time to be established. So qualitative reasoning is suitable for incomplete knowlegde specification.
- some aspects of product quality are not readily expressed in physical quantities, e.g. *dust quality* (amount of undesirable dust on the tube) or amount of wall wetting. Also, a lot of physical relations -e.g. turbulence - cannot be translated into practical mathematical models. In both cases, the only way to deal with them is by introducing qualitative relations.

The qualitative (Q-) formalism which is used here is based on the *Sign Algebra* and from the *process expert knowledge representation*. This formalism is the basis of most qualitative approaches like those introduced by Kuipers (Kuipers 86), by Forbus (Forbus 84) or by de Kleer & Brown (DeKleer & Brown 84).

We summarize here below the qualitative specification that has been used in this model:

- Qualitative behavioural model: (see (Rak-et-al.92))
  - sign algebra set S = {-,0,+}, $S' = S \cup \{?\}$
  - disjoint value $(s,t) = s \vee t$ (logical-OR)
  - $S_d = \{(-,0),(0,+),(-,+)\}$
  - universe of calculus: $S'' = S' \cup S_d$
  - basic mappings (`Qplusp()`, `Plusp()`, `Distp()`)
  - sign operators $\oplus$ and $\ominus$ for set S, extended to set S''
  - compound mappings (`Plus()`, `Dist()`, `Tune()` and all possible combinations of them)

The meaning of the qualitative variables is as follows: a symbol represent the deviation of the variable from its nominal value. So, a variable taking the value (+) means that its actual value is higher than its nominal one.

3.3.3 The Hybrid Model —The hybrid relations are specified **to predict** some results of the process simulation. An hybrid parameter is designed to be a couple of two values `h = (val, qi)`, where `val` is a numerical value and `qi` is a **qualitative influence** on the parameter and is a qualitative value as it was specified in the earlier paragraph.
For example, the couple `p = (3,+)` means that the numerical value of p is 3, but due to qualitative influence is in fact **higher than 3** (+). So the qualitative values {-,0,+} means that the *actual* value is *lower than, exactly equal to, higher than* the reference value, respectively. The symbolic part gives additionnal information with respect to the quantitative one. These combinations of quantitative and qualitative relations will allow to predict some results from the process simulation.

4 PROCESS SIMULATION

4.1 Simulating with G2

The TV-tube manufacturing process is simulated with G2 Expert System). The choice of an expert system to simulate the process has been motivated to the ability of a such system to *handle both quantitative and qualitative variables*, (not to mention its graphical facilities of representation) and also because of its *rule-based structure* which allows to reason and then to perform the supervision task.
In G2 expert system, every piece of knowledge is represented by an **item**: workspaces, rules, formulas, functions, procedures are some examples of items. Each item has an **attribute** that is a piece of information associated to it.

In order to give a clear representation, the items are structured in different knowledge-based (KB) workspace (or window), which can have one or more sub-workspaces. The upper workspace is the **root** one which has, as sub-workspaces, the following items: a) the *object definition*, b) the rules, c) the procedures and d) the process workspace

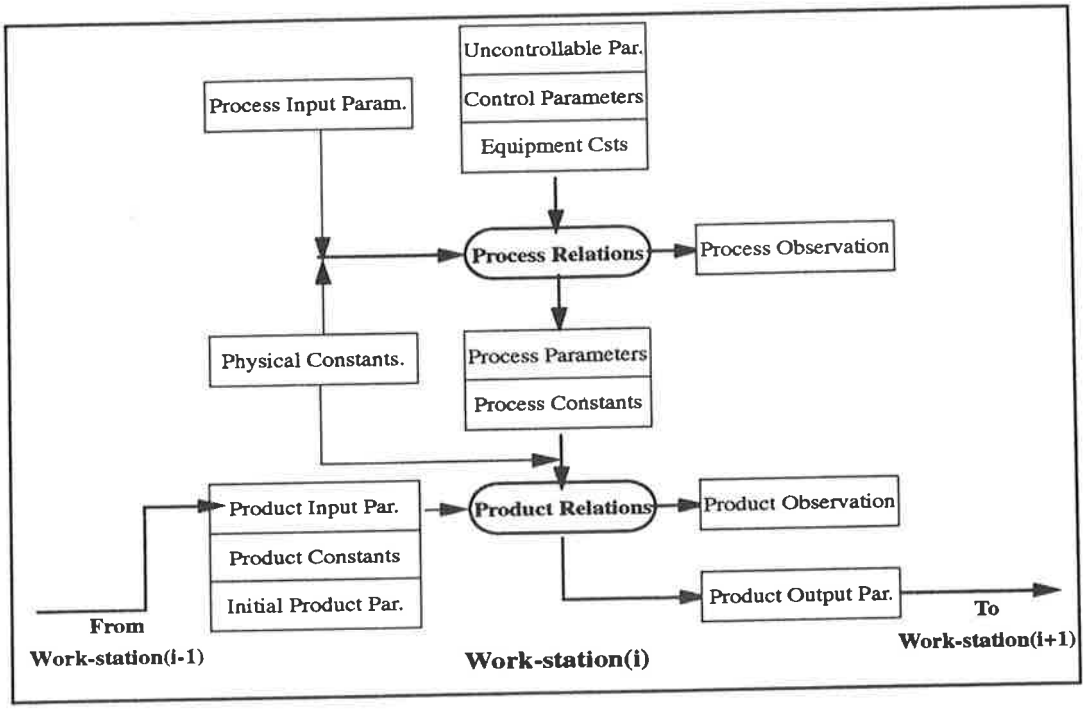


Fig. 7. Basic structure of specification

TABLE 2: Sign addition extended to set S''

$\oplus$	$-$	$(-,0)$	0	$(0,+)$	$+$	$(-,+)$	$?$
$-$	$-$	$-$	$-$	$?$	$?$	$?$	$?$
$(-,0)$	$-$	$(-,0)$	$(-,0)$	$?$	$?$	$?$	$?$
0	$-$	$(-,0)$	0	$(0,+)$	$+$	$(-,+)$	$?$
$(0,+)$	$?$	$?$	$(0,+)$	$(0,+)$	$+$	$?$	$?$
$+$	$?$	$?$	$+$	$+$	$+$	$?$	$?$
$(-,+)$	$?$	$?$	$(-,+)$	$?$	$?$	$?$	$?$
$?$	$?$	$?$	$?$	$?$	$?$	$?$	$?$

TABLE 3: Basic and compound qualitative mappings

Mappings	Initial set(s)	Final set	Expression
Qlusp(s)	S''	S''	$= \begin{cases} (0) & \text{if } (s = (-) \text{ or } s = (-,0) \text{ or } s = (0)), \\ (+) & \text{if } s = (+), \\ (0,+) & \text{otherwise.} \end{cases}$
Plusp(x, Δ)	$R \times R$	S''	$= \begin{cases} (0) & \text{if } x \leq 0, \\ (0,+) & \text{if } 0 < x \leq \Delta, \\ (+) & \text{if } x > \Delta. \end{cases}$
Distp(x, Δ)	$R \times R$	S''	$= \begin{cases} (0) & \text{if } x \leq 0, \\ (?) & \text{if } 0 < x \leq \Delta, \\ (-,+) & \text{if } x > \Delta. \end{cases}$
Plus(x, Δ_1, Δ_2)	$R \times R \times R$	S''	$=$ Plusp($x - \Delta_1, \Delta_2$) $\ominus$ Plusp($-x - \Delta_1, \Delta_2$)
Dist(x, Δ_1, Δ_2)	$R \times R \times R$	S''	$=$ Distp($x - \Delta_1, \Delta_2$) $\oplus$ Distp($-x - \Delta_1, \Delta_2$)
Tune(x, Δ_1, Δ_2)	$R \times R \times R$	S''	$=$ Plusp($x - \Delta_1, \Delta_2$) $\oplus$ Plusp($-x - \Delta_1, \Delta_2$)

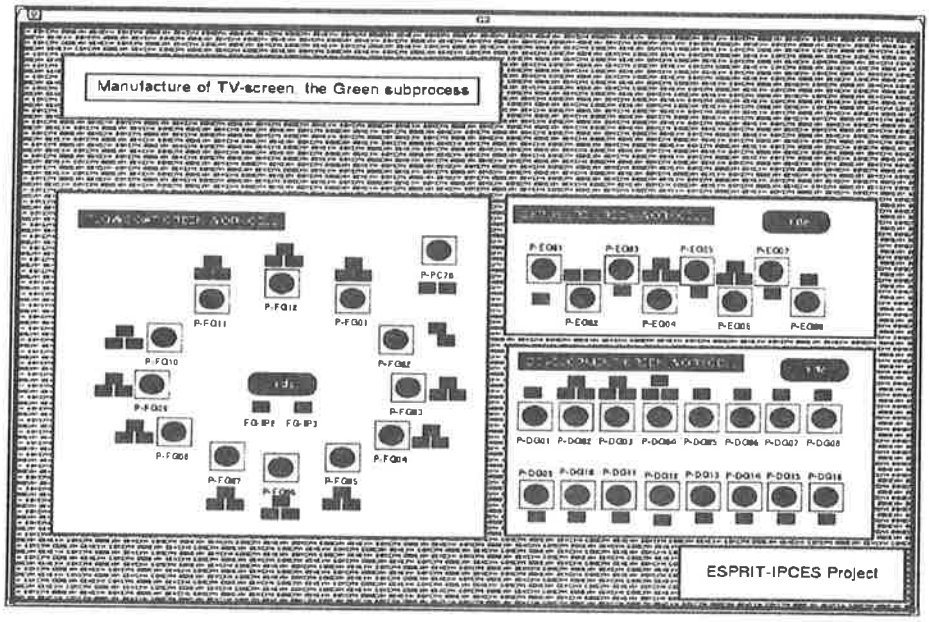


Fig. 8. The Green subprocess in G2

The process has been designed according to its physical decomposition. So, the process workspace has 3 sub-workspaces (i.e. 3 colour subprocesses). Each colour subprocess has 3 work-cells which are the Flowcoat, the Exposure and the Development work-cells, composed by 12, 8 and 16 work-stations, respectively. The Green subprocess is depicted in Figure 8 below.

4.2 Some Simulation Results

We give here as first illustration is **the rotation speed of the tube in the Flowcoat Green work-cell** (= 12 work-stations) (Table 4). These simulation results have been validated with the real process.

TABLE 4: Rotation speed of the screen in Green work-cell

work-station	rotation speed (r.p.m)
wks.1	07.156
wks.2	03.142
wks.3	03.142
wks.4	02.435
wks.5	15.708
wks.6	15.708
wks.7	12.566
wks.8	12.566
wks.9	15.708
wks.10	15.708
wks.11	18.850
wks.12	18.850

The second illustration is **the output window of the FG-04 work-station** (Flowcoat Green work-cell). We can find some *numerical values* like the tube temperature `tg`, the suspension layer thickness `hsus` or the solution layer thickness `hsol`, as well as *qualitative values* like the amount of agglomerates on the tube `qdag = 'eq'` (for equal, i.e. 0) or the amount of wall wetting `qdww = 'gt'` (for greater than, i.e. +). The output results from FG-04 work-station are given in Table 5.

TABLE 5: Results from FG-04 work-station

Process Observations		
vdp[1] = 9.703e-5		
Product Observations		
qdww[1] = 'gt'		
Product Output Parameters		
tg[1] = 32.732	tg[2] = 32.711	tg[3] = 32.688
vsens = 0.045	vpva = 168355e-2	vph = 0.148
hsus = 5.584e-4	hsol = 5.178e-4	hsed = 1.297e-6
vphmax = 0.5	dph = 5.0e-6	qdab = 'gt'
qdag = 'eq'	qdww[1] = 'gt'	qdust = 'eq'

4.3 Simulation Results Analysis

We reproduce here as a basis for analyzing the simulations results, the evaluation criteria defined by Cohen and Howe (Coh-How89) for the stage 5 (analyze experiment results).

- **program performance:** The software program gives more satisfaction with the third prototype (PT3) than the second one (PT2) especially in time efficiency.
- **predictions:** the quantitative results from the simulation seem to be realistic. The qualitative one have to be checked while they were produced to test the simulation software. The hybrid relations specification seems to be an accurate of prediction with its formalism **higher than, exactly equal to, lower than** (with respect to some reference value).
- **time/space efficiency of the program:** Within the expert system, it is possible to simulate the process behaviour in different time bases. But the time efficiency was rather low when the expert system was used alone. The introduction

of a fortran program as foreign module for the expert system has considerabily increased the time efficiency, from 20 minutes (G2 alone) to about one minute (with a Fortran program) to simulate the whole process.

- **easy for intended users:** at this stage of project achievement (first part of development) the software has been only designed for the developper, though it will be analyzed for the software implementation in the factory.

- **program's performance limitations:** only one tube has been simulated which is not the real situation in the factory. This is done only to validate the process model. The simulation of sequential tubes will be addressed afterwards.

5 CONCLUDING REMARKS

We have presented in this paper a knowledge-based approach in modelling a complex industrial process, a manufacture of a TV-tube system. The process is specified according to its hierarchical and structural model (Process / subprocess /work-cell / work-station). The proposed formalism for each smallest component (work-station) which is based on the **hybrid** (quantitative and qualitative) representation seems to be accurate to model such process, while the complete model cannot be obtained immediately but only by successive model improvements. The qualitative specification is based on the *sign algebra* and on **the process expert knowledge representation.** One of the main advantage of the knowledge-based approach is the modularity of the system (which can be easily modified, without a complete knowledge reorganization. In this paper, we have focussed on the modelling phase which is very important with respect to the following one, that is the control task.

As a long term perspective, we can cite the generalization of this approach of modelling to other manufacturing processes having the same structural decomposition. This hybrid quantitative-qualitative specification (per work-station) may also be interesting to model other systems like control systems, because of the basic concept of specification (controllability and observability). In that way, on-going research concerns the specification of other processes having the same structure but focussing on the hybrid (numerical and symbolic) causal relations of the variables (Rakoto 93). This type of specification is called *Hybrid Causal Graph*, and we believe that this concept could lead to a more generic model process simulation and control.

ACKNOWLEDGMENTS

The authors are grateful to every people who contributed to this project, especially to Adelaida Delgado-Dominguez, Jose Lluis De La Rosa and to the Process Expert, Jan S. Kikkert (Philips TCDC). Their full disponibility was very helpful to the achievement of this work.

REFERENCES

J.S. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems Man and Cybernetics.*, 21(3):473–509, May-June 1991.

K.E. Arzén. An architecture for expert system based feedback control. *Automatica*, 25(6):813–827, 1989.

K.E. Arzén. A survey of commercial real-time expert system environments. *proc. 1992 IFAC/IFIP/IMACS Int. Symp. on AIRTC*, Delft, The Netherlands, June 1992.

K.E. Arzén. A model-based control system concept. *Internal report LUTFD2/TFRT-3213-SE*, Lund Instutute of Technology (Sweden), December 1992.

K.J. Astrom, J.J. Anton and K.E. Arzén. Expert control. *Automatica*, 22(3):277–286, 1986.

P.R. Cohen and A.E. Howe. Toward AI research methodology: Three case studies in evaluation. *IEEE Transactions on Systems Man and Cybernetics.*, 19(3):634–646, May-June 1989.

A. Delgado, N. Rakoto-Ravalontsalama, and J.L. de la Rosa. "Esprit 2428 IPCES: Prototype 3 simulation part". Internal Report N. 92108, LAAS-CNRS, Toulouse, France, March 1992.

Gensym Corporation, Cambridge, MA 02140, USA. *G2 Reference Manual (Ver 2.1)*, August 1990.

Gensym Corporation, Cambridge, MA 02140, USA. *G2 Reference Manual (Ver 3.0)*, July 1992.

J. Kamerbeek. Generating simulators from causal process knowledge. *European Simulation Symposium 1993.* Delft, The Netherlands, October 1993.

J. Kamerbeek and J.S. Kikkert. Esprit 2428 IPCES: Process description of prototype 2. Private Communication, Philips-PRL and Philips-TCDC, Eindhoven, The Netherlands, December 1991.

C.R. Mc Lean. Interface concepts for plug-compatible production management systems. *Computers in Industry*, 9(4), December 1987.

N. Rakoto-Ravalontsalama, A. Missier, and J.S. Kikkert. "Qualitative operators and process engineer semantics of uncertainty". In B. Bouchon-Meunier, L. Valverde, and R.R. Yager, editors, *Lecture Notes in Computer Science 682*, IPMU'92 – Advanced Methods in Artificial Intelligence, pages 284–293. Springer Verlag, 1992.

N. Rakoto-Ravalontsalama. Control-oriented hybrid causal graph. Internal Report N.93372, LAAS-CNRS, Toulouse (F), October 1993.

W.W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proc. Wescon*, August 1970. also avalaible in Proc. ICSE 9, Computer Society Press, 1987.

B.C. Williams. A theory of interaction: Unifying qualitative and quantitative algebraic reasoning. *Artificial Intelligence Journal*, 51:39–94, 1991.

A PERSPECTIVE ON THE INTEGRATED ARTIFICIAL INTELLIGENCE/KNOWLEDGE BASED SYSTEMS IN THE PROCESS INDUSTRIES: CHALLENGES AND OPPORTUNITIES

Professor R S Benson
Chief Engineer, ICI Engineering, Billingham, Cleveland TS23 1LD, UK
Visiting Professor: Centre of Process Systems Engineering: Imperial College

ABSTRACT

A perspective of progress in the process industries is developed to provide a backcloth for artificial intelligence/knowledge-based systems. This is related to experience in ICI and other industries. Analysis of the evidence suggests the need for more focus on the benefits to Business Managers and response to the changing technology in the process industry. The paper concludes that, while the potential for AI/KBS is high the 'jury is still out' on the benefits.

Key Words: perspective, artificial intelligence,knowledge based systems, process industries, challenges, opportunities

1 INTRODUCTION

A perspective on artificial intelligence/knowledge based systems (AI/KBS) first requires a view of the process industries. The author has recently provided such a view for the IFAC Conference in Kyoto (Ref 1). Only a summary is presented in this paper.

Historically, the fifties and sixties were periods of rapid growth for the process industries worldwide. This was driven by the growing world economies and the continuing stream of innovations such as pharmaceuticals and plastics. Profits were high and new technology was valued. During that period, many of the first applications of process control, optimisation and computers were developed and exploited in the process industries. Chemical engineering became the central skill and safety became the dominant performance parameter. This period could be described as one that was comfortable for the manufacturers.

During the eighties and into the nineties, the perspective has changed. Most process industries such as chemicals, food, pharmaceuticals are showing distinct signs of maturity. Profits are at the least very low, resources are being trimmed, businesses are being focused and the industry is in a phase of re-trenchment. There are exceptions such as biochemical engineering but these are few and far between. In a climate such as this, phrases such as 'making the assets sweat' rise to the foreground and new technology is often considered risky. Manufacturing engineering, as an approach to studying the whole chain from raw material to final customer, is increasingly an important skill.

Using accepted chemical efficiency measures, the performance of the industry is improving. Yields are increasing, waste products are diminishing and avoidable trips and shutdowns are reducing. However, using the performance measures of other manufacturing industries such as consumer goods, the story is much less exciting. For example, the stockturn of the industry is hardly changing. Stockturn is defined as the total annual sales divided by the value of the stock. It is possibly one of the most sensitive measures of process improvement. High stocks can be indicative of poor control; too many breakdowns; shutdowns that take too long or too many different grades. In general, the industry supplies from stock rather than direct from the production line. This compares starkly with, for example, the motor industry (Ref 3) where phrases such as 'just-in-time', 'Kaizan', 'made to order', 'on time in full' delivery are now common and the impact has been dramatic.

Comparing the process industries with other industries

(Ref 2) could suggest that in many ways the process industry is now approaching a similar position to that of cars and steel about 10 to 15 years ago. At that time, those industries went through major change of which process control was a major factor. Improved process control had immediate impact on the production process, though often in the form of statistical process control and quality measurement. It also played a significant role in the demand for concurrent engineering used to reduce the time to market for new models. Finally, it had a dramatic impact on the control of the engines. Within a modern motor car is a computer control system for the engine which is probably more advanced than any chemical process control system. What is more, it is mass produced, relatively cheap and achieves an extremely high performance specification.

The observation in this analysis is that process control probably has a great deal to offer the process industry. That is in both the manufacturing process and the control of the product quality. As in other industries, necessity will be the mother of 'innovation' and hence the drive for improved process control will become an imperative in the late nineties.

2 POSITION OF ARTIFICIAL INTELLIGENCE/KNOWLEDGE BASED SYSTEMS

This analysis of the industry would suggest an increasing demand for sophisticated process control systems. What is the experience of artificial intelligence and knowledge based systems in the process industries?

The literature would suggest that there have been many applications. Most articles on the subject give examples and work within the process industries. However, detailed examination suggests that the actual on-plant experience may not be as large. The UK government and other governments have been financially encouraging the adoption and exploitation of these systems. Within ICI we have had some experience.

It is in the area of neural nets that ICI has some experience; and the following observations draw on this experience. This suggests that the concept of neural nets and related subjects can and does work in process control. It is best applied to problems with the following characteristics:

a Where there is a non-linear relationship between the cause and effect

b where process data is readily available and complete

c where the signal:noise ratio of the data is greater than 6

d most importantly where linear approximation is insufficient to solve the problem and where linearisation of the problem or engineering simplification cannot lead to a suitable solution

e good data collection facilities exist

This experience is already beginning to restrict the scope. We have further learned that the most suitable have the following characteristics:

a a simple architecture with single layer, as few hidden nodes as possible and dynamic incorporated using simple first order filters

b where engineering knowledge is available, it is incorporated in the data pre-processing stage prior to neural net

c preferably the position should be one of multi-input but single output

All this simply says is that the more experience and knowledge is built in to the application of artificial intelligence, the higher the probability of success. Experience has not been good where the neural net has simply been treated as a black box to solve or recognise a pattern without any guidance. These rules essentially eliminate the idea that neural nets can solve any problem. In fact, as far as control is concerned, this technology would probably be better described as non-linear parameter estimation. The approach has strong similarities to linear regression science which is well reported elsewhere.

Specific applications of neural nets/artificial intelligence to a range of problems within ICI has resulted with varying degrees of success. Some examples are:-

2.1 *Inferential Measurements*

This may well prove to be the most beneficial area. Where the products are measured or sold on some parameter which is difficult to measure on line, then it is very attractive to estimate this parameter from available information. Theory exists, the number of inputs is often few and the value of a correct inferred measurement is high. It is rarely repeatable and the data is often available. In addition, it is often non-

linear and hence satisfies most of the characteristics. Our experience has been good and it is anticipated that more applications will occur in that area.

2.2 *Control Strategy*

There are a few dynamic problems where it is extremely difficult or expensive to develop mathematical models. In those circumstances, neural networks, artificial intelligence offers scope and has been used. While attractive, the opportunities are fairly limited and this is not thought to be of great value.

2.3 *Fault Detection*

Much has been written about the potential for fault detection and the attractiveness is clear. It is an area where data is often interpreted to identify an embedded fault; where the human operator is particularly good and yet where the occurrence is random and often infrequent. In theory, neural nets should be extremely good at this pattern recognition. Our experience in practice has not yet convinced us of the case. It is, however, noticeable that one major machinery monitoring firm is now offering a software package to do just that on large machines and this would suggest that this is an area which will grow.

2.4 *Biotechnology*

This is proving a promising area. The process technology is relatively new and the requirements for sterility make measurement difficult and expensive. In addition, the biological nature of the reaction processes suggest that artificial intelligence/knowledge-based systems should be applicable. Experience is confirming this analysis (Ref 9) with AI/KBS being used for model building, inferential measurement and increasingly for control.

The observation from this experience is that artificial intelligence/knowledge based systems should be most applicable for tasks that are normally carried out by plant operators on a relatively infrequent basis. They are tasks where the connection of facts and pattern recognition is key and where intelligence is required. The task mentioned of inferential measurement, fault detection, and process trends all have potential. Equally, applications into detecting faults in sensors and controlling batch profile are important areas where unknown parameters are significant. They could be areas of potential.

3 EXPERIENCE IN OTHER INDUSTRIES

In drawing a perspective, it is always useful to examine what has happened in other industries. The greatest reported successes for AI/KBS appear to be in the financial world. Areas such as credit evaluation, portfolio management, management decisions etc. Here the premise is that by recognising patterns in historical data one can predict the future. The adoption would suggest that there is some evidence that this is correct. However, the key characteristic of the financial world is the high degree of competitiveness and its global nature. it is a market where the slightest improvement in performance gives significant competitive and hence financial benefit. This would suggest that when the process industries get to that stage of competitiveness, the opportunities for this technology will increase.

If one includes fuzzy logic and adaptive control within the definition of AI/KBS then analogies can be drawn with industries such as cement. These have often been the earliest adopters of this technology. Characteristic of these industries was that their process control was very poor and the measurement of the key parameters was almost impossible. They are industries which moved from poor control to advanced control in almost a single step. This compares with the balance of the process industries where control is good to adequate and incentive is not quite so large.

4 ANALYSIS OF THE EVIDENCE

By observation, introduction of a new technology to the process industry may take up to 20 years. For the initial years, there is a lot of academic talk and technical presentations. Eventually the stage is reached where the deliverables do not match the expectations and the publicity becomes quiet. During the quiet period, those working the subject continue to work and the promised benefits begin to be realised. Towards the end of the 20 years, the technology becomes just a matter of fact built into existing offerings as a standard offering and making a significant contribution. It would appear that AI/KBS could be approaching a period of quietness. This is often, I should add, the most productive period.

There is strong evidence that AI/KBS has been marketed very heavily and very technically. It is important to recognise how terms like 'artificial intelligence' and 'knowledge based systems' could appear to the business managers. who are the ultimate customers of the technology. The author took the opportunity to examine the meanings of the words 'artificial', 'intelligence' and 'knowledge' in a thesaurus.

There are some interesting interpretations. No doubt, people will be happy with artificial intelligence being described as:

simulated sagacity
manmade wisdom
manufactured aptitude
experienced based learning

I would suggest, however, that they would be less happy with other suggested interpretations:

fake news
counterfeit information
spurious information
new-based learning or data-based learning

In other words, I am suggesting that the titles presently adopted could potentially do damage in the rate of adoption within the Process Industries. The titles are derived from the technical background. They are not described in terms of the customer benefits. The following are suggestions as to how the technology could be described in 'customer benefit' terms.

Safety Enhancers
Wastage Reducers
Plant Availability Improvers
Quality Improvers

In every case, the technology has been embedded within a customer benefit phrase. It is interesting that the QCBRIN by Bruell Kerr is described as 'a trained ear' and 'a spot checker'. The phrase that has been used within ICI is the concept of an 'intelligent wheeltapper'. The author is suggesting that the way AI/KBS is marketed from a technical point of view is detrimental to its use.

Given this analysis of the terminology, coupled with the position that process control is still undervalued within industry would suggest that the future is limited in the near term.

The opposite view, however, would suggest that as the competitive pressures on the process industry increase significantly the attraction of using neural networks will increase and also their contribution will be valued. The opportunity is to identify and accept that impending challenge and be ready for it.

5 HOW COULD THE PROCESS INDUSTRIES CHANGE

Earlier in this paper and in other papers (Ref 4, 5, 6) the author has made a case for change in the Process Industries technology and means of delivery. There are the pressures of profit margins, the high capital cost of new investment, environmental, safety. and biological processes. Add to this, the changes in the marketplace, the use of bar codes and till roll analysis to drive order patterns and the incessant pressure to improve delivery and quality while reducing stocks and cost. A conclusion is that the assumption that all chemical plants will operate on large chemical sites is questionable. It is now possible to invent and test a biological molecule in much less time than it takes to design, build and validate the process to manufacture. It is now possible for a customer to change their order pattern hourly on plants that were designed to run for many months at a single production rate. These pressures will demand change. It is suggested that this change will manifest itself in the form of 'responsive processes'. (Ref 6, 7) These will be small, mass produced processes which can be delivered anywhere by road transport. They will operate either on the customer's site or be one of many in parallel on a larger manufacturing site. They will have a number of characteristics:

- small, modular and mass produced
- totally manless at the site although there may be remote monitoring and management
- capable of operating at the customer's demand rates. This will imply automatic start-up and shutdown and repeatable quality operations at any rates specified by the customer.
- wherever possible, manufacturing product from feedstocks which are readily available such as air, water, electricity, ethylene and gas where there are grids, salt and other materials.
- inherently safe

The relevance of this concept to AI/KBS is the implied assumption that the operator would have to be replaced by a control scheme. The control scheme will have to cope with much more than just steady state and it could be suggested that it will demand a high degree of intelligence. In addition to the normal production patterns, there will be abnormal patterns that have to be recognised and acted upon. The control schemes will be mass-produced ('chip') based very similar to those in modern motor cars. The question posed is where will AI/KBS systems fit in such a concept. The quiet period is probably about to come and the opportunity is just

around the corner. Will it be satisfied by AI/KBS or will it, in fact, be satisfied by other technologies?

6 CONCLUSIONS

The author's conclusion is that in the case for AI/KBS systems in the process industries the 'jury' is still out. Within certain defined circumstances, the technology works. It can identify patterns of historical data and it can do tasks where there are no other means. Given the fact that the process industry operates already, these tasks are, however, a small set of the total set because solutions exist for the balance of the issues. What will be the added values that this technology brings that cannot be brought by further exploitation of other known technologies. It will probably be inferential measurement, fault detection and operation of manless plants. To achieve this, will probably require a more customer focused approach to publicity and marketing.

The challenges exist, the opportunities will arise, will the technology be the appropriate one to adopt?

REFERENCES

1 Benson, R S (1994) The Process Industry Requirements of Advanced Control Techniques; Challenges and Opportunities, Keynote address; ADCHEM 94, Kyoto, Japan.

2 Benson, R S (1988) Process Systems Engineering : Past, present and a personal view of the future, Keynote address, PSE '88, and *Computers in Chemical Engineering*, Vol 13, No. 11/12, pp 1193-1198, 1989.

3 Womack, J P; Jones, D T; Roos, D: The Machine that changed the World; Macmillan Publishing Company, 1990; ISBN: 0-89256-350-3

4 Benson, R S; Ponton, J W (1991) Process Miniaturisation - A Route to Total Environmental Acceptability ? *Trans of the Institution of Chemical Engineers*, Vol 71, No A2

5 Marlin, T E; Perkins, J D; Barton, G W ; Brisk, M L (1991), Benefits from Process Control : Results of a joint Industry-University Study, J. Proc. Control, 1, 57-83

6 Control III. Inst. Chem. E., York, September, 1992

7 Niwa, T; Honda, A (1990), Transferable Vessel-type Multi-purpose Batch process, *Chemical Engineering*, Japan, vol. 54, No 5

8 Pantelides, C C; Realff, M J; Shah, N (1992) Short-Term Scheduling of Pipeless Batch Plants, AIChE 1992 Annual Meeting

9 Massimo, C Di; Willis, M J; Montague, G A; Tham, M T; Morris, A J; (1991) Bioprocess model building using artificial neural networks. Bioprocess Engineering 7(1991) 77-82.

ACKNOWLEDGEMENT

The views expressed in the paper are those of the author and should be taken as the official view of either his employer, ICI, or the Centre of Process System Engineering at Imperial College, London, where he is a visiting professor.

The author wishes to acknowledge the many creative thoughts and much stimulation he has received from his colleagues in ICI and the Centre of Process Systems Engineering. In particular, the author wishes to recognise the contribution of Paul Turner of ICI and Newcastle University who is working for the Teaching Company Scheme on Neural Networks.

Improvement of Mold–Level Control using Fuzzy–Logic

N. Kiupel, P.M. Frank and J. Wochnik

Department of Electrical Engineering, University of Duisburg Bismarckstrasse 81 BB, W-47048 Duisburg, Germany, e-mail: hl420ki@duc220.uni-duisburg.de

Abstract. In this paper we report on a feasibility study of an Improvement of conventional mold–level control using fuzzy logic. For an economic steel production nowadays continuous casting is used. Of special interest for continuous casting is the mold, because in this part of the process the liquid steel will be fed to the mold. From the metallurgical point of view it is necessary to keep the mold–level constant. Greater variations of the mold–level cause non–homogeneity in the product. Especially an overflow of the mold as well as an empty mold has to be avoided, because the liquid steel overflows the working area or the bottom parts of the mold respectively. Beside an interrupt in the production, in addition, high maintenance costs will be caused. At the example of a slab–caster with varying mold geometry it will be shown in this article that in case of disturbances an improvement of the conventional control, using fuzzy logic, can be achieved. The basic idea is the use of a parallel control structure with fuzzy logic. This provides the advantage that under normal operation conditions the conventional controller is still in operation, whilst in the case of a disturbance the fuzzy controller *supports* the conventional control. This leads to a much better disturbance behaviour as without the fuzzy controller, that means, that in opposite to conventional control especially an overflow can be avoided.

Keywords. Mold–level control, fuzzy logic, fuzzy logic controller,PI–controller

1. Introduction

For the investigation of the improvement of the conventional control by using fuzzy logic a conventional controlled continuous caster has been used. The control task is to keep the mold–level within a small bound around the setpoint.

However, in this paper the fuzzy logic has been used, there are also other method to deal with this kind of problem, like adaptive control, neural network–based control or other theoretical methods can be used.

To study this process by simulation a complete model of the process has been developed. This dynamical model allows the simulation of the mold–level in correspondance to the setpoint and disturbances as well. The results of the simulation are compared with measurements of the real process. The simulation results are approximately equal to those of the measurements. Therefore the mathematical model can be treated as well validated. With this mathematical model it is now possible to generate disturbances in order to prove the fuzzy controller.

2. Model of the Mold–level Control Circuit

The components of the mold–level control circuit are described in Fig. 1

1. Tundish
2. Submerged nozzle
3. Mold
4. Liquid steel–level
5. Strand
6. Mold–level measurement device
7. Steering unit
8. Stopper lifting device
9. Stopper

The liquid steel is fed to the submerged nozzle (2) via the tundish (1) into the mold (3). The mold–level is the result of the liquid steel fed into

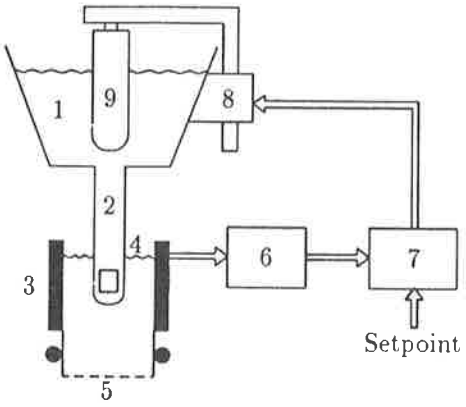


Fig. 1. Components of the mold–level control circuit.

the mold and the strand (5). The mold–level itself is measured by a mold–level measurement device (6), based on a radiometric measurement principle. The measurement value is the input to the steering unit (7). The steering unit contains, beside units for monitoring, the mold–level controller. The input to the controller is the difference between the actual and the desired mold–level (error of the mold–level). The output of the controller is the setpoint for the stopper lifting device (8), which moves the stopper (9). The resulting value of the stopper position is due to the input flow of the liquid steel.

2.1 Some characteristics of the mold–level control circuit

The mold–level control circuit can be characterized by some properties which are summarized as below, (Kiupel 1994).

1. The dominating time constant of the mold–level control circuit is the time constant of the mold–level measurement unit. The value of the time constant is approximately $0.2s$. This is, however, not usual, but in order to measure the actual mold–level there arise some difficulties. A compromise in the accuracy of the measurement because of the environmental conditions has to be made. As mentioned earlier, the mold–level is measured using the radiation measurement principle, which is a contactless measurement principle. This is due to the very strange environmental measurement conditions, e.g. especially the high temperature near the mold and the dusty air around the mold. This and other conditions together restrict the number of possibilities to measure the mold–level. But, however, this method is quite succesful and has the following advantages:
   - Reliability,
     the measurement is taken contactless, and therefore a direct contact of the measurement unit with the mold is avoided.
   - Robustness (in this case robustness means the robustness aginst environmental conditions)
     the measurement unit is placed below the mold and is protected with a shield against demolition.
   - User friendly calibration.
     The calibration of the measurement unit is quite simple

   Of course, this measurement principle has some disadvantages as well:
   - the real mold–level cannot be detected. This is due to cast powder and slag which is not considered.
   - The measurement detection is relatively slow. A compromise between the deviation of the measurement and the desired accuracy has to be made, because of the high frequency measurement disturbance.

   Taking this into account, the radiation measurement principle is a robust method with a reasonable accuracy.
   To evaluate wether the true mold–level has increased the upper level or not it is necessary to have the value of the true mold–level. In this context the true mold–level means the level **before** the mold–level measurement device, whilst the measured mold–level means the mold–level **after** the mold–level measurement device. Of course this is not possible in practice because the value of a measurement is the value **after** the measurement device. It is always assumed, that the output of the measurement device is somehow proportional to the real value. It will be shown later that the true mold–level and the measured mold–level are not exactly proportional. Because these are simulation studies, it is possible to take the true mold–level into account. The difference of the true mold–level and the measurement is the dynamic of the measurement device which has also been modelled to provide a precise model of the process.
2. The mold oscillates in the vertical direction. This is due to process requiremens to reduce the friction between the metal and the mold. This oscillation is also measured in the mold–level.
3. The steel flow as a function of the stopper position is highly nonlinear. The stopper–characteristic is shown in Fig. 2. It can be

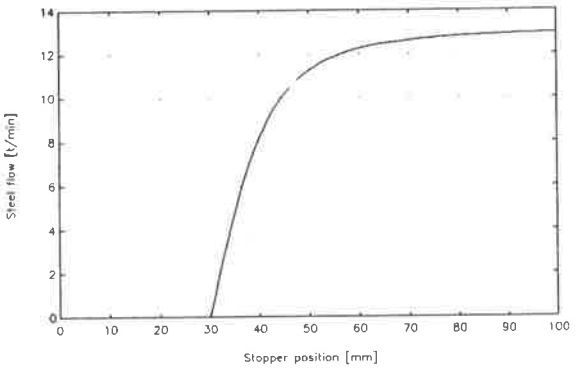


Fig. 2. Flow characteristic of the stopper.

seen, that there is a dead zone and a saturation as well. Furthermore the operating point (around 35 to 40 mm) of the charateristic has a great slope which leads for small variations of the stopper position to great variations in the steel flow.

3. Fuzzy PI–Control of the Mold–level Circuit

The conventional mold–level controller is a simple PI-controller. The task for the controller is to keep the mold–level constant at a certain operating point. To control the liquid steel level the variation of the steel flow via the stopper position has been used. This task should be performed for different speeds as well as for disturbances in the mold–level.

Under normal operation conditions, or if the disturbances are within a small bound, the used PI–controller works satisfactory. The deviations from the nominal mold–level are around $\pm 1.5mm$. This deviations are acceptable and fulfill the requirements for high quality steel production.

Greater disturbances during the slab production are relatively seldom, but however, if they occur and therefore the mold–level deviates very much from the working point the quality of the steel decreases significantly.

Some of the disturbances lead to a production failure und therefore the maintenance costs are very high. Two types of faults will be distinguished:

1. An overflow of the liquid steel over the top of the mold. This kind of disturbance is very dangerous for the operating personal.
2. The mold–level decreases below the bottom of the mold. That means that the production process is interrupted.

One possible cause for the overflow is that oxids of the liquid steel condens on the bottom part of the stopper and around the input flow point of the submerged nozzle. If these condensates suddenly break from the stopper the steel flow increases rapidly and therefore the mold–level as well. These disturbances cannot be avoided with the classical PI–controller. Therefore a modified approach is presented in order to prevent these disturbances

The principle of fuzzy–logic, which was developed by Zadeh (Zadeh 1973), has been widely used. An introduction to the theory of fuzzy systems can be found elsewhere (Kaufmann 1975), (Zimmermann 1991).

Currently, there exists no *systematic* design procedure for the fuzzy controllers as in the case of the classical controllers. Normally, the parameters and the structure of the fuzzy controller are "optimized" as long as the results leading to a better performance, however it has been shown that with the use of a fuzzy controller the performance and the robustness as well can be improved, see e.g. (Kiupel and Frank 1993). On the other hand this design method is useful if no, or a very bad mathematical model exists.

Usually, most of the classical controller design strategies are strongly dependent on the preciseness of the mathematical model. The use of a fuzzy controller is decoupled from the mathematical model, because no *mathematical* model is used. In addition, if the mathematical model is very complex, as for this process, and the model cannot be simplified for the design of a classical controller the fuzzy concept is a much promising concept, because in a fuzzy controller it is possible to accumulate process knowledge and transform it to the rule–base for the fuzzy controller. This is also a model of the process, but no explicit mathematical model which can sometimes be much easier to generate than a mathematical model.

Finally, the implementation of the fuzzy controller for this plant has lead to a complementation of the PI–controller. The first attempt for the solution of this problem was to design a fuzzy controller as a complete substitution of the classical PI–controller (Kiupel and Frank 1993), (King and Mamdani 1977), (Boverie *et al.* 1991).

The inputs for the fuzzy controller are the error of the mold–level and the derivation of the mold–level. In opposite to the conventional control nonlinearities can be incorporated into the knowledge–base.

A better solution to fulfill both, a good steady state performance as well as a good robustness against disturbances was not found.

Using this procedure it is possible to design the fuzzy controller only for the disturbance whilst the PI–controller is still in operation for nominal behaviour. This provides the advantage, that the design of the fuzzy controller is decoupled from the design of the PI–controller. Therefore the fuzzy controller can be optimized for the disturbance control of the process. To achieve the desired performance described above, a fuzzy PD–type controller has been chosen. The D–part of the controller is only activated if the error of the mold–level decreases a certain level. Finally, this structure has lead to good results because of the decoupling of the design for the disturbance and the steady state performance respectively.

The inputs for this type of fuzzy controller are the error of the mold–level and the derivation of the error as well. The derivation is numerically generated by taking the difference of two samples and divide it by the sampling time. The output of the fuzzy controller is weighted by a factor and then added to the output of the conventional controller. The used controller is depicted in Fig. 3. As mentioned earlier, the parameters

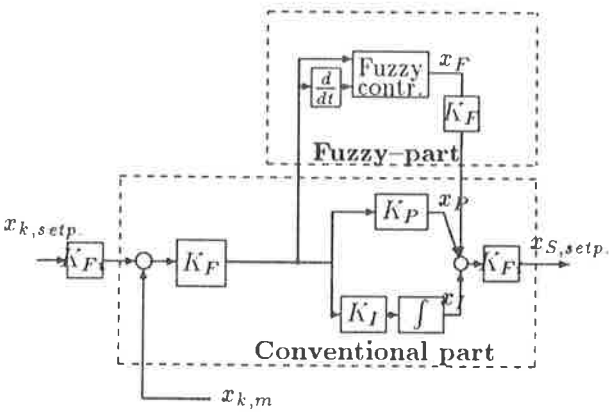


Fig. 3. Extended control scheme

of the fuzzy controller are generated in simulation studies. The disturbance has been simulated as a change of the stopper position. Using this disturbance, the parameters of the fuzzy controller are adapted to this disturbance in order to get a good performance.

The fuzzy sets for the input are shown in Fig. 4 and Fig. 5 respectively. The fuzzy sets for the output are shown in Fig. 6. For the inference the *Min Max* method has been used and for the defuzzification the **center of area** method has been used. Using this fuzzy sets for the input and

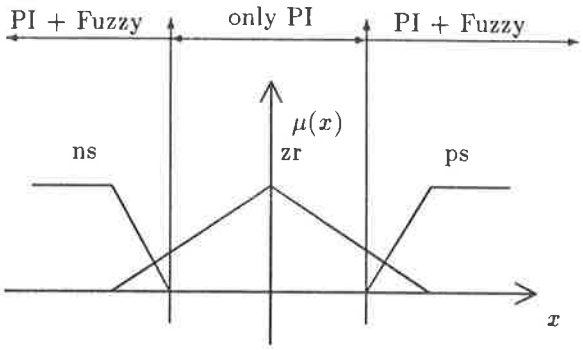


Fig. 4. Fuzzy sets of the error of the mold–level

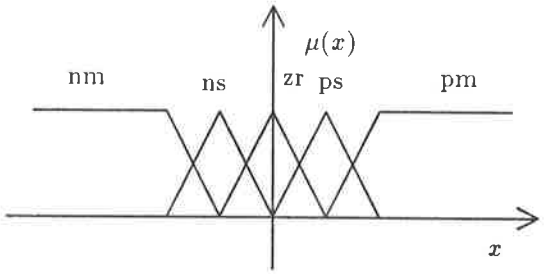


Fig. 5. Fuzzy sets of the derivation of the error of the mold–level

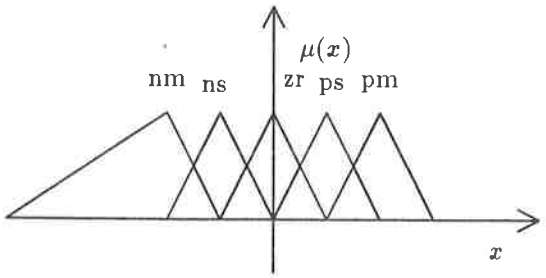


Fig. 6. Fuzzy sets of the output of the controller

the output respectively, the rule base can be determined. The chosen rule base is presented in Fig. 7. The used abbreviations for the fuzzy sets are depicted in Tab. 1. If the mold–level increases the setpoint, the error is negativ and therefore the assignment to the fuzzy sets negativ small (NS) and zero (ZR) of the output has been chosen and vice versa. If the mold–level is within the margin of $\pm 4mm$ this will be assigned exclusively to the fuzzy set zero (ZR), see also Fig. 4. The derivation of the error is according to Fig. 5 divided into

(Ref 2) could suggest that in many ways the process industry is now approaching a similar position to that of cars and steel about 10 to 15 years ago. At that time, those industries went through major change of which process control was a major factor. Improved process control had immediate impact on the production process, though often in the form of statistical process control and quality measurement. It also played a significant role in the demand for concurrent engineering used to reduce the time to market for new models. Finally, it had a dramatic impact on the control of the engines. Within a modern motor car is a computer control system for the engine which is probably more advanced than any chemical process control system. What is more, it is mass produced, relatively cheap and achieves an extremely high performance specification.

The observation in this analysis is that process control probably has a great deal to offer the process industry. That is in both the manufacturing process and the control of the product quality. As in other industries, necessity will be the mother of 'innovation' and hence the drive for improved process control will become an imperative in the late nineties.

2 POSITION OF ARTIFICIAL INTELLIGENCE/KNOWLEDGE BASED SYSTEMS

This analysis of the industry would suggest an increasing demand for sophisticated process control systems. What is the experience of artificial intelligence and knowledge based systems in the process industries?

The literature would suggest that there have been many applications. Most articles on the subject give examples and work within the process industries. However, detailed examination suggests that the actual on-plant experience may not be as large. The UK government and other governments have been financially encouraging the adoption and exploitation of these systems. Within ICI we have had some experience.

It is in the area of neural nets that ICI has some experience; and the following observations draw on this experience. This suggests that the concept of neural nets and related subjects can and does work in process control. It is best applied to problems with the following characteristics:

a Where there is a non-linear relationship between the cause and effect

b where process data is readily available and complete

c where the signal:noise ratio of the data is greater than 6

d most importantly where linear approximation is insufficient to solve the problem and where linearisation of the problem or engineering simplification cannot lead to a suitable solution

e good data collection facilities exist

This experience is already beginning to restrict the scope. We have further learned that the most suitable have the following characteristics:

a a simple architecture with single layer, as few hidden nodes as possible and dynamic incorporated using simple first order filters

b where engineering knowledge is available, it is incorporated in the data pre-processing stage prior to neural net

c preferably the position should be one of multi-input but single output

All this simply says is that the more experience and knowledge is built in to the application of artificial intelligence, the higher the probability of success. Experience has not been good where the neural net has simply been treated as a black box to solve or recognise a pattern without any guidance. These rules essentially eliminate the idea that neural nets can solve any problem. In fact, as far as control is concerned, this technology would probably be better described as non-linear parameter estimation. The approach has strong similarities to linear regression science which is well reported elsewhere.

Specific applications of neural nets/artificial intelligence to a range of problems within ICI has resulted with varying degrees of success. Some examples are:-

2.1 *Inferential Measurements*

This may well prove to be the most beneficial area. Where the products are measured or sold on some parameter which is difficult to measure on line, then it is very attractive to estimate this parameter from available information. Theory exists, the number of inputs is often few and the value of a correct inferred measurement is high. It is rarely repeatable and the data is often available. In addition, it is often non-

linear and hence satisfies most of the characteristics. Our experience has been good and it is anticipated that more applications will occur in that area.

2.2 *Control Strategy*

There are a few dynamic problems where it is extremely difficult or expensive to develop mathematical models. In those circumstances, neural networks, artificial intelligence offers scope and has been used. While attractive, the opportunities are fairly limited and this is not thought to be of great value.

2.3 *Fault Detection*

Much has been written about the potential for fault detection and the attractiveness is clear. It is an area where data is often interpreted to identify an embedded fault; where the human operator is particularly good and yet where the occurrence is random and often infrequent. In theory, neural nets should be extremely good at this pattern recognition. Our experience in practice has not yet convinced us of the case. It is, however, noticeable that one major machinery monitoring firm is now offering a software package to do just that on large machines and this would suggest that this is an area which will grow.

2.4 *Biotechnology*

This is proving a promising area. The process technology is relatively new and the requirements for sterility make measurement difficult and expensive. In addition, the biological nature of the reaction processes suggest that artificial intelligence/knowledge-based systems should be applicable. Experience is confirming this analysis (Ref 9) with AI/KBS being used for model building, inferential measurement and increasingly for control.

The observation from this experience is that artificial intelligence/knowledge based systems should be most applicable for tasks that are normally carried out by plant operators on a relatively infrequent basis. They are tasks where the connection of facts and pattern recognition is key and where intelligence is required. The task mentioned of inferential measurement, fault detection, and process trends all have potential. Equally, applications into detecting faults in sensors and controlling batch profile are important areas where unknown parameters are significant. They could be areas of potential.

3 EXPERIENCE IN OTHER INDUSTRIES

In drawing a perspective, it is always useful to examine what has happened in other industries. The greatest reported successes for AI/KBS appear to be in the financial world. Areas such as credit evaluation, portfolio management, management decisions etc. Here the premise is that by recognising patterns in historical data one can predict the future. The adoption would suggest that there is some evidence that this is correct. However, the key characteristic of the financial world is the high degree of competitiveness and its global nature. it is a market where the slightest improvement in performance gives significant competitive and hence financial benefit. This would suggest that when the process industries get to that stage of competitiveness, the opportunities for this technology will increase.

If one includes fuzzy logic and adaptive control within the definition of AI/KBS then analogies can be drawn with industries such as cement. These have often been the earliest adopters of this technology. Characteristic of these industries was that their process control was very poor and the measurement of the key parameters was almost impossible. They are industries which moved from poor control to advanced control in almost a single step. This compares with the balance of the process industries where control is good to adequate and incentive is not quite so large.

4 ANALYSIS OF THE EVIDENCE

By observation, introduction of a new technology to the process industry may take up to 20 years. For the initial years, there is a lot of academic talk and technical presentations. Eventually the stage is reached where the deliverables do not match the expectations and the publicity becomes quiet. During the quiet period, those working the subject continue to work and the promised benefits begin to be realised. Towards the end of the 20 years, the technology becomes just a matter of fact built into existing offerings as a standard offering and making a significant contribution. It would appear that AI/KBS could be approaching a period of quietness. This is often, I should add, the most productive period.

There is strong evidence that AI/KBS has been marketed very heavily and very technically. It is important to recognise how terms like 'artificial intelligence' and 'knowledge based systems' could appear to the business managers. who are the ultimate customers of the technology. The author took the opportunity to examine the meanings of the words 'artificial', 'intelligence' and 'knowledge' in a thesaurus.

There are some interesting interpretations. No doubt, people will be happy with artificial intelligence being described as:

simulated sagacity
manmade wisdom
manufactured aptitude
experienced based learning

I would suggest, however, that they would be less happy with other suggested interpretations:

fake news
counterfeit information
spurious information
new-based learning or data-based learning

In other words, I am suggesting that the titles presently adopted could potentially do damage in the rate of adoption within the Process Industries. The titles are derived from the technical background. They are not described in terms of the customer benefits. The following are suggestions as to how the technology could be described in 'customer benefit' terms.

Safety Enhancers
Wastage Reducers
Plant Availability Improvers
Quality Improvers

In every case, the technology has been embedded within a customer benefit phrase. It is interesting that the QCBRIN by Bruell Kerr is described as 'a trained ear' and 'a spot checker'. The phrase that has been used within ICI is the concept of an 'intelligent wheeltapper'. The author is suggesting that the way AI/KBS is marketed from a technical point of view is detrimental to its use.

Given this analysis of the terminology, coupled with the position that process control is still undervalued within industry would suggest that the future is limited in the near term.

The opposite view, however, would suggest that as the competitive pressures on the process industry increase significantly the attraction of using neural networks will increase and also their contribution will be valued. The opportunity is to identify and accept that impending challenge and be ready for it.

5 HOW COULD THE PROCESS INDUSTRIES CHANGE

Earlier in this paper and in other papers (Ref 4, 5, 6) the author has made a case for change in the Process Industries technology and means of delivery. There are the pressures of profit margins, the high capital cost of new investment, environmental, safety. and biological processes. Add to this, the changes in the marketplace, the use of bar codes and till roll analysis to drive order patterns and the incessant pressure to improve delivery and quality while reducing stocks and cost. A conclusion is that the assumption that all chemical plants will operate on large chemical sites is questionable. It is now possible to invent and test a biological molecule in much less time than it takes to design, build and validate the process to manufacture. It is now possible for a customer to change their order pattern hourly on plants that were designed to run for many months at a single production rate. These pressures will demand change. It is suggested that this change will manifest itself in the form of 'responsive processes'. (Ref 6, 7) These will be small, mass produced processes which can be delivered anywhere by road transport. They will operate either on the customer's site or be one of many in parallel on a larger manufacturing site. They will have a number of characteristics:

- small, modular and mass produced
- totally manless at the site although there may be remote monitoring and management
- capable of operating at the customer's demand rates. This will imply automatic start-up and shutdown and repeatable quality operations at any rates specified by the customer.
- wherever possible, manufacturing product from feedstocks which are readily available such as air, water, electricity, ethylene and gas where there are grids, salt and other materials.
- inherently safe

The relevance of this concept to AI/KBS is the implied assumption that the operator would have to be replaced by a control scheme. The control scheme will have to cope with much more than just steady state and it could be suggested that it will demand a high degree of intelligence. In addition to the normal production patterns, there will be abnormal patterns that have to be recognised and acted upon. The control schemes will be mass-produced ('chip') based very similar to those in modern motor cars. The question posed is where will AI/KBS systems fit in such a concept. The quiet period is probably about to come and the opportunity is just

around the corner. Will it be satisfied by AI/KBS or will it, in fact, be satisfied by other technologies?

6 CONCLUSIONS

The author's conclusion is that in the case for AI/KBS systems in the process industries the 'jury' is still out. Within certain defined circumstances, the technology works. It can identify patterns of historical data and it can do tasks where there are no other means. Given the fact that the process industry operates already, these tasks are, however, a small set of the total set because solutions exist for the balance of the issues. What will be the added values that this technology brings that cannot be brought by further exploitation of other known technologies. It will probably be inferential measurement, fault detection and operation of manless plants. To achieve this, will probably require a more customer focused approach to publicity and marketing.

The challenges exist, the opportunities will arise, will the technology be the appropriate one to adopt?

REFERENCES

1 Benson, R S (1994) The Process Industry Requirements of Advanced Control Techniques; Challenges and Opportunities, Keynote address; ADCHEM 94, Kyoto, Japan.

2 Benson, R S (1988) Process Systems Engineering : Past, present and a personal view of the future, Keynote address, PSE '88, and *Computers in Chemical Engineering*, Vol 13, No. 11/12, pp 1193-1198, 1989.

3 Womack, J P; Jones, D T; Roos, D: The Machine that changed the World; Macmillan Publishing Company, 1990; ISBN: 0-89256-350-3

4 Benson, R S; Ponton, J W (1991) Process Miniaturisation - A Route to Total Environmental Acceptability ? *Trans of the Institution of Chemical Engineers*, Vol 71, No A2

5 Marlin, T E; Perkins, J D; Barton, G W ; Brisk, M L (1991), Benefits from Process Control : Results of a joint Industry-University Study, J. Proc. Control, 1, 57-83

6 Control III. Inst. Chem. E., York, September, 1992

7 Niwa, T; Honda, A (1990), Transferable Vessel-type Multi-purpose Batch process, *Chemical Engineering*, Japan, vol. 54, No 5

8 Pantelides, C C; Realff, M J; Shah, N (1992) Short-Term Scheduling of Pipeless Batch Plants, AIChE 1992 Annual Meeting

9 Massimo, C Di; Willis, M J; Montague, G A; Tham, M T; Morris, A J; (1991) Bioprocess model building using artificial neural networks. Bioprocess Engineering 7(1991) 77-82.

ACKNOWLEDGEMENT

The views expressed in the paper are those of the author and should be taken as the official view of either his employer, ICI, or the Centre of Process System Engineering at Imperial College, London, where he is a visiting professor.

The author wishes to acknowledge the many creative thoughts and much stimulation he has received from his colleagues in ICI and the Centre of Process Systems Engineering. In particular, the author wishes to recognise the contribution of Paul Turner of ICI and Newcastle University who is working for the Teaching Company Scheme on Neural Networks.

Improvement of Mold–Level Control using Fuzzy–Logic

N. Kiupel, P.M. Frank and J. Wochnik

Department of Electrical Engineering, University of Duisburg Bismarckstrasse 81 BB, W-47048 Duisburg, Germany, e-mail: hl420ki@duc220.uni-duisburg.de

Abstract. In this paper we report on a feasibility study of an Improvement of conventional mold–level control using fuzzy logic. For an economic steel production nowadays continuous casting is used. Of special interest for continuous casting is the mold, because in this part of the process the liquid steel will be fed to the mold. From the metallurgical point of view it is necessary to keep the mold–level constant. Greater variations of the mold–level cause non–homogeneity in the product. Especially an overflow of the mold as well as an empty mold has to be avoided, because the liquid steel overflows the working area or the bottom parts of the mold respectively. Beside an interrupt in the production, in addition, high maintenance costs will be caused. At the example of a slab–caster with varying mold geometry it will be shown in this article that in case of disturbances an improvement of the conventional control, using fuzzy logic, can be achieved. The basic idea is the use of a parallel control structure with fuzzy logic. This provides the advantage that under normal operation conditions the conventional controller is still in operation, whilst in the case of a disturbance the fuzzy controller *supports* the conventional control. This leads to a much better disturbance behaviour as without the fuzzy controller, that means, that in opposite to conventional control especially an overflow can be avoided.

Keywords. Mold–level control, fuzzy logic, fuzzy logic controller,PI–controller

1. Introduction

For the investigation of the improvement of the conventional control by using fuzzy logic a conventional controlled continuous caster has been used. The control task is to keep the mold–level within a small bound around the setpoint.

However, in this paper the fuzzy logic has been used, there are also other method to deal with this kind of problem, like adaptive control, neural network–based control or other theoretical methods can be used.

To study this process by simulation a complete model of the process has been developed. This dynamical model allows the simulation of the mold–level in correspondance to the setpoint and disturbances as well. The results of the simulation are compared with measurements of the real process. The simulation results are approximately equal to those of the measurements. Therefore the mathematical model can be treated as well validated. With this mathematical model it is now possible to generate disturbances in order to prove the fuzzy controller.

2. Model of the Mold–level Control Circuit

The components of the mold–level control circuit are described in Fig. 1

1. Tundish
2. Submerged nozzle
3. Mold
4. Liquid steel–level
5. Strand
6. Mold–level measurement device
7. Steering unit
8. Stopper lifting device
9. Stopper

The liquid steel is fed to the submerged nozzle (2) via the tundish (1) into the mold (3). The mold–level is the result of the liquid steel fed into

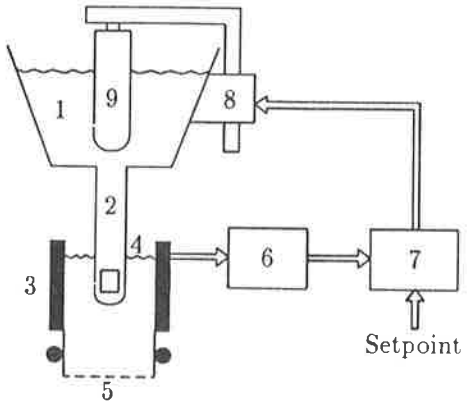


Fig. 1. Components of the mold–level control circuit.

the mold and the strand (5). The mold–level itself is measured by a mold–level measurement device (6), based on a radiometric measurement principle. The measurement value is the input to the steering unit (7). The steering unit contains, beside units for monitoring, the mold–level controller. The input to the controller is the difference between the actual and the desired mold–level (error of the mold–level). The output of the controller is the setpoint for the stopper lifting device (8), which moves the stopper (9). The resulting value of the stopper position is due to the input flow of the liquid steel.

2.1 Some characteristics of the mold–level control circuit

The mold–level control circuit can be characterized by some properties which are summarized as below, (Kiupel 1994).

1. The dominating time constant of the mold–level control circuit is the time constant of the mold–level measurement unit. The value of the time constant is approximately $0.2s$. This is, however, not usual, but in order to measure the actual mold–level there arise some difficulties. A compromise in the accuracy of the measurement because of the environmental conditions has to be made. As mentioned earlier, the mold–level is measured using the radiation measurement principle, which is a contactless measurement principle. This is due to the very strange environmental measurement conditions, e.g. especially the high temperature near the mold and the dusty air around the mold. This and other conditions together restrict the number of possibilities to measure the mold–level. But, however, this method is quite succesful and has the following advantages:
   - Reliability,
     the measurement is taken contactless, and therefore a direct contact of the measurement unit with the mold is avoided.
   - Robustness (in this case robustness means the robustness aginst environmental conditions)
     the measurement unit is placed below the mold and is protected with a shield against demolition.
   - User friendly calibration.
     The calibration of the measurement unit is quite simple

   Of course, this measurement principle has some disadvantages as well:
   - the real mold–level cannot be detected. This is due to cast powder and slag which is not considered.
   - The measurement detection is relatively slow. A compromise between the deviation of the measurement and the desired accuracy has to be made, because of the high frequency measurement disturbance.

   Taking this into account, the radiation measurement principle is a robust method with a reasonable accuracy.
   To evaluate wether the true mold–level has increased the upper level or not it is necessary to have the value of the true mold–level. In this context the true mold–level means the level **before** the mold–level measurement device, whilst the measured mold–level means the mold–level **after** the mold–level measurement device. Of course this is not possible in practice because the value of a measurement is the value **after** the measurement device. It is always assumed, that the output of the measurement device is somehow proportional to the real value. It will be shown later that the true mold–level and the measured mold–level are not exactly proportional. Because these are simulation studies, it is possible to take the true mold–level into account. The difference of the true mold–level and the measurement is the dynamic of the measurement device which has also been modelled to provide a precise model of the process.
2. The mold oscillates in the vertical direction. This is due to process requiremens to reduce the friction between the metal and the mold. This oscillation is also measured in the mold–level.
3. The steel flow as a function of the stopper position is highly nonlinear. The stopper–characteristic is shown in Fig. 2. It can be

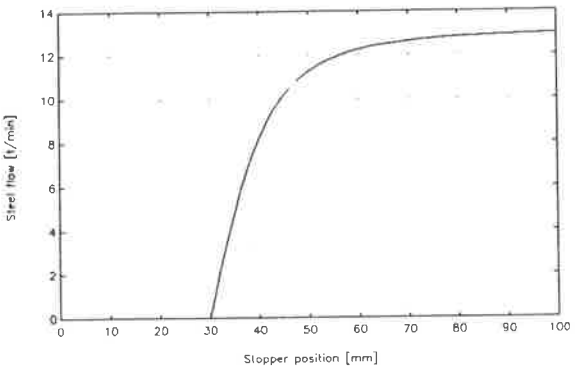


Fig. 2. Flow characteristic of the stopper.

seen, that there is a dead zone and a saturation as well. Furthermore the operating point (around 35 to 40 mm) of the charateristic has a great slope which leads for small variations of the stopper position to great variations in the steel flow.

3. Fuzzy PI–Control of the Mold–level Circuit

The conventional mold–level controller is a simple PI-controller. The task for the controller is to keep the mold–level constant at a certain operating point. To control the liquid steel level the variation of the steel flow via the stopper position has been used. This task should be performed for different speeds as well as for disturbances in the mold–level.

Under normal operation conditions, or if the disturbances are within a small bound, the used PI–controller works satisfactory. The deviations from the nominal mold–level are around $\pm 1.5mm$. This deviations are acceptable and fulfill the requirements for high quality steel production.

Greater disturbances during the slab production are relatively seldom, but however, if they occur and therefore the mold–level deviates very much from the working point the quality of the steel decreases significantly.

Some of the disturbances lead to a production failure und therefore the maintenance costs are very high. Two types of faults will be distinguished:

1. An overflow of the liquid steel over the top of the mold. This kind of disturbance is very dangerous for the operating personal.
2. The mold–level decreases below the bottom of the mold. That means that the production process is interrupted.

One possible cause for the overflow is that oxids of the liquid steel condens on the bottom part of the stopper and around the input flow point of the submerged nozzle. If these condensates suddenly break from the stopper the steel flow increases rapidly and therefore the mold–level as well. These disturbances cannot be avoided with the classical PI–controller. Therefore a modified approach is presented in order to prevent these disturbances

The principle of fuzzy–logic, which was developed by Zadeh (Zadeh 1973), has been widely used. An introduction to the theory of fuzzy systems can be found elsewhere (Kaufmann 1975), (Zimmermann 1991).

Currently, there exists no *systematic* design procedure for the fuzzy controllers as in the case of the classical controllers. Normally, the parameters and the structure of the fuzzy controller are "optimized" as long as the results leading to a better performance, however it has been shown that with the use of a fuzzy controller the performance and the robustness as well can be improved, see e.g. (Kiupel and Frank 1993). On the other hand this design method is useful if no, or a very bad mathematical model exists.

Usually, most of the classical controller design strategies are strongly dependent on the preciseness of the mathematical model. The use of a fuzzy controller is decoupled from the mathematical model, because no *mathematical* model is used. In addition, if the mathematical model is very complex, as for this process, and the model cannot be simplified for the design of a classical controller the fuzzy concept is a much promising concept, because in a fuzzy controller it is possible to accumulate process knowledge and transform it to the rule–base for the fuzzy controller. This is also a model of the process, but no explicit mathematical model which can sometimes be much easier to generate than a mathematical model.

Finally, the implementation of the fuzzy controller for this plant has lead to a complementation of the PI–controller. The first attempt for the solution of this problem was to design a fuzzy controller as a complete substitution of the classical PI–controller (Kiupel and Frank 1993), (King and Mamdani 1977), (Boverie *et al.* 1991).

The inputs for the fuzzy controller are the error of the mold–level and the derivation of the mold–level. In opposite to the conventional control nonlinearities can be incorporated into the knowledge–base.

A better solution to fulfill both, a good steady state performance as well as a good robustness against disturbances was not found.

Using this procedure it is possible to design the fuzzy controller only for the disturbance whilst the PI–controller is still in operation for nominal behaviour. This provides the advantage, that the design of the fuzzy controller is decoupled from the design of the PI–controller. Therefore the fuzzy controller can be optimized for the disturbance control of the process. To achieve the desired performance described above, a fuzzy PD–type controller has been chosen. The D–part of the controller is only activated if the error of the mold–level decreases a certain level. Finally, this structure has lead to good results because of the decoupling of the design for the disturbance and the steady state performance respectively.

The inputs for this type of fuzzy controller are the error of the mold–level and the derivation of the error as well. The derivation is numerically generated by taking the difference of two samples and divide it by the sampling time. The output of the fuzzy controller is weighted by a factor and then added to the output of the conventional controller. The used controller is depicted in Fig. 3. As mentioned earlier, the parameters

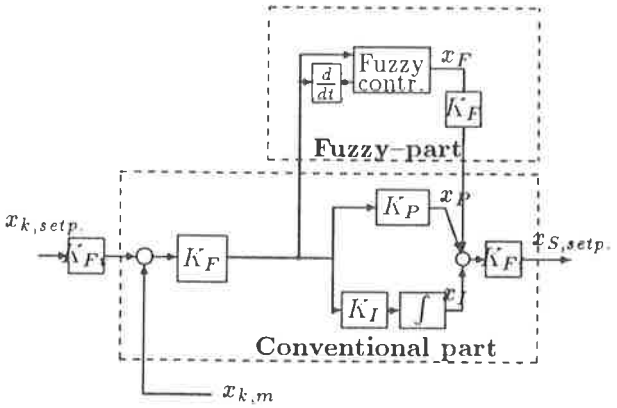


Fig. 3. Extended control scheme

of the fuzzy controller are generated in simulation studies. The disturbance has been simulated as a change of the stopper position. Using this disturbance, the parameters of the fuzzy controller are adapted to this disturbance in order to get a good performance.

The fuzzy sets for the input are shown in Fig. 4 and Fig. 5 respectively. The fuzzy sets for the output are shown in Fig. 6. For the inference the *Min Max* method has been used and for the defuzzification the **center of area** method has been used. Using this fuzzy sets for the input and

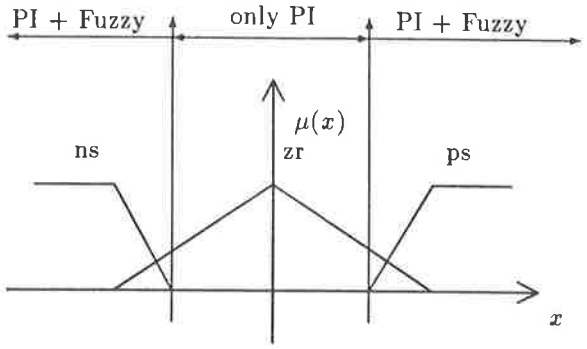


Fig. 4. Fuzzy sets of the error of the mold–level

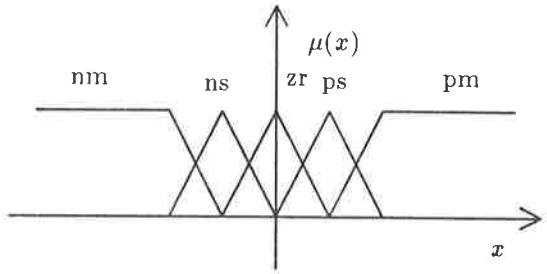


Fig. 5. Fuzzy sets of the derivation of the error of the mold–level

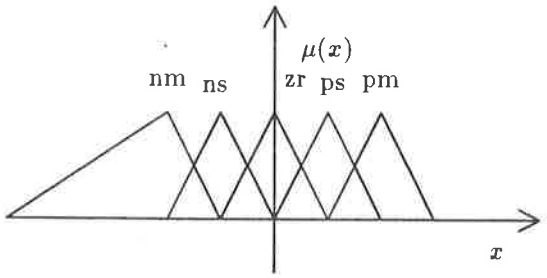


Fig. 6. Fuzzy sets of the output of the controller

the output respectively, the rule base can be determined. The chosen rule base is presented in Fig. 7. The used abbreviations for the fuzzy sets are depicted in Tab. 1. If the mold–level increases the setpoint, the error is negativ and therefore the assignment to the fuzzy sets negativ small (NS) and zero (ZR) of the output has been chosen and vice versa. If the mold–level is within the margin of $\pm 4mm$ this will be assigned exclusively to the fuzzy set zero (ZR), see also Fig. 4. The derivation of the error is according to Fig. 5 divided into

e \ de	NM	NS	ZR	PS	PM
NS	NM	NM	NM	NS	ZR
ZR	ZR	ZR	ZR	ZR	ZR
PS	ZR	ZR	PS	PS	PS

Fig. 7. Rule-base of the used fuzzy controller

abbreviation	output
nm	negative medium
ns	negative small
zr	zero
ps	positive small
pm	positive medium

TABLE 1 Abbreviations for the fuzzy sets

5 fuzzy sets. The fuzzy sets negative small (NS) and negative medium (NM) describe an increasing mold-level, whilst the fuzzy stes positive small (PS) and positive medium (PM) respectively describe a decreasing mold-level. The fuzzy set zero (ZR) is an indication for a constant mold-level.

The fuzzy sets of the output of the fuzzy-PD-controller are assigned in a similar manner. The fuzzy sets positive small (PS) and positive medium (PM) decribe an increase of the stopper position, whilst the fuzzy sets negative small (NS) and negative medium (NM) describe a decrease of the stopper position. And also the fuzzy set zero (ZR) is an indication for no change in the stopper position.

From the rule base, see Fig. 7, it can be seen, that for error values within the tolerance band the output of the fuzzy-PD-controller is always zero, regardless of the value of the derivation of the error. But if the error is negativ and the derivation of the error too then the fuzzy-PD-controller decreases the stopper position in order to avoid an overflow of the mold.

For a positive error and a positive value of the derivation of the error the stopper position will be increased by the fuzzy-PD-controller in order to avoid an interrupt of the production. Because of the nonlinearity of the stopper characteristic the change of the stopper position is different in the positive and negative direction respectively. This is considered using different fuzzy sets for negative medium (NM) and positive medium (PM).

4. Simulation results

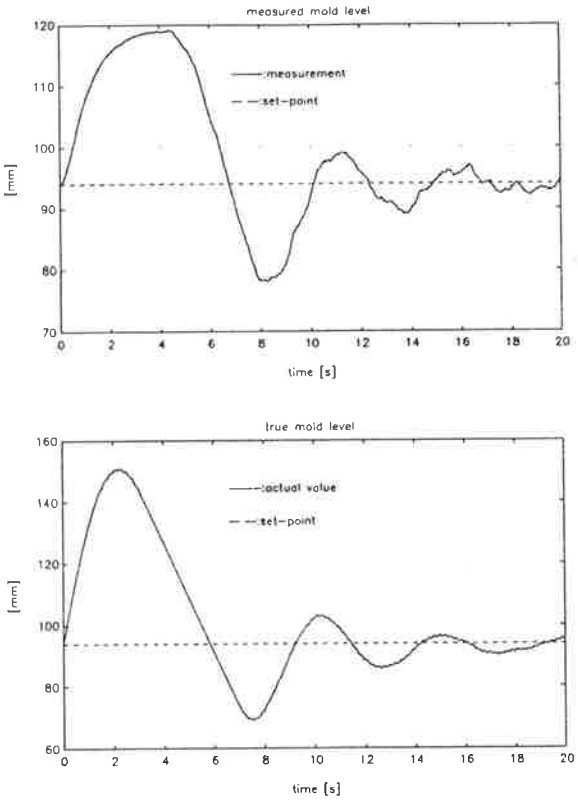


Fig. 8. PI: change of the stopper position from 70 mm to 58 mm

To prove the performance of the fuzzy controller, the results of the fuzzy controlled system are compared with those of the classical controlled system. For this task a disturbance in the stopper position has been considered. The cause for this kind of disturbance occurs always in practice if oxides of steel, which are slowly growing on the bottom of the stopper and at the whole of the steel cuvette, see Fig. 1, suddenly break from the stopper or the cuvette and the steel flow increases rapidly. In the simuation this disturbance is considered as a step function. The following plots are showing the following states of the process:

- the mold-level setpoint and the mold-level measurement value (top)
- the true (physical not measured) mold-level and the setpoint of the mold-level (lower right)

From Fig. 8 and Fig. 9 it can be seen, that the extension of the conventional controller with the fuzzy-PD-controller is superior to the conventional control alone.

As a disturbance a step function of the stopper position from 70 mm to 58 mm at $t = 0$ has been considered. The conventional controller reacts extremely inert. The true mold-level increases up to 150 mm, the upper bound (120 mm) of the mold-

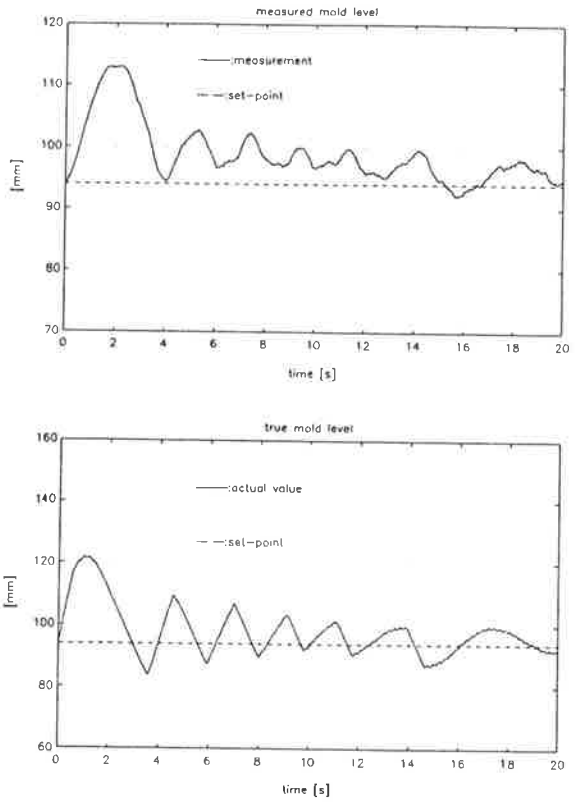


Fig. 9. Fuzzy: change of the stopper position from 70 mm to 58 mm

level measurement device is increased. In practice this means that the top of the mold is increased and therefore the liquid steel flows in the working area.

In opposite to the conventional control, the fuzzy–PD–controlled mold–level reacts much faster, the stopper position is decrased immedeately and the true mold–level moves only up to 125 mm. This is a reduction of 50 %. In addition, to the fast reaction of the disturbance, the transient is much more damped. This is a direct reaction of the D–part of the fuzzy controller. The time the system needs to come to the equilibrium is in both cases approximately equal.

It can be seen that the fuzzy–PD–controller is superior to the conventional controller. In addition, the parallel (complement) fuzzy controller is from the practical point of view much more interesting because the conventional controller keeps still in operation so that the fuzzy controller has just to be *added* to the existing system. In principle, there is no need to interrupt the production process to incorporate the fuzzy controller.

5. Conclusion

A fuzzy controller for a continuous caster has been developed, which operates in parallel manner to the conventional process. The advantage of this concept is that the conventional controller is not substituted, it is just complemented. This leads to a higher acceptance in practise and makes it more easy to implement the fuzzy controller.

The disturbance rejection of the mold–level control circuit has been increased by 50 %. This has been achieved by taking the error and the derivation of the error as the input to the fuzzy controller. Finally, this leads to a fuzzy–PD–controller.

Using this fuzzy controller, with two inputs and one output, provides an easy implementable controller, in opposite to other methods where deep theoretical knowledge of the process and the used control theory is necessary.

The next task is to implement this fuzzy controller to the process. The nowadays tools for implementation provide a good possibility to implement this type of controller to the existing controller.

REFERENCES

Boverie, S., B. Demaya and A. Titli (1991). *FUZZY LOGIC CONTROL COMPARED WITH OTHER AUTOMATIC CONTROL APROACHES.* Conference of Decision and Control.

Kaufmann, A. (1975). *Introduction to the theory of fuzzy subsets.* Academic Press.

King, P. J. and E. Mamdani (1977). *The Application of Fuzzy Control Systems to Industrial Processes.* Vol. 13. Automatica. Pergamon Press.

Kiupel, N. (1994). 'PI–Fuzzy–D'. *VDI/VDE/GMA–Ausspracheta g Fuzzy Control, Langen, Germany.*

Kiupel, N. and P. M. Frank (1993). 'Fuzzy control of steam turbines'. *International Journal of System Science.*

Zadeh, L. A. (1973). *Outline of anew approach to the analysis of complex systems and decision processes.* IEEE Tr. on Systems, Man and Cybernetics.

Zimmermann, H. (1991). 'Fuzzy set theory- and its applications'. *Kluwer academic publishers.*

RIP CONTROL IN KNOWLEDGE-BASED SYSTEMS

D. DRECHSEL and M. PANDIT

University of Kaiserslautern, Department of Electrical Engineering, 67663 Kaiserslautern, GERMANY

Abstract. Rule based interpolating control (***RIP*** control) is an approach to deal with linguistic expert knowledge in the form of production rules. As an alternative to fuzzy control, *RIP* control transforms the production rules into support points in a hyperspace and applies multidimensional interpolation methods to obtain the input/output mapping of the controller. The paper introduces *RIP* control and its application in knowledge-based systems. The potential of hybrid *RIP* rule bases are demonstrated using flow control as an example.

Key Words. Rule based interpolating control, *RIP* control, multidimensional interpolation, knowledge-based system, linguistic expert knowledge, hybrid rule bases, fuzzy control.

1. INTRODUCTION

The paper presents a knowledge-based system using rule based interpolating control (***RIP***). The goal of *RIP* control is to convert linguistic expert knowledge in form of production rules into an input/output mapping of a controller. At present, fuzzy control is widely used for this purpose. However, in certain aspects, fuzzy control has its disadvantages like the lack of methodology and transparency.

The basic principle of *RIP* control is to generate support points in a hyperspace based on the production rules. Subsequently, multidimensional interpolation methods are employed to obtain a comprehensive input/output mapping of the controller.

The paper introduces *RIP* control and sketches the distinctions to fuzzy control. A complete *RIP* control design shell and knowledge-based system shell is described. Hybrid rule structures will be discussed, their capabilities in flow control will be demonstrated and compared with fuzzy control.

2. *RIP* CONTROL

The general structure of a knowledge-based system using the *RIP* method consists basically of 3 blocks, as depicted in Fig. 1. The input & output filters incorporate the dynamic parts of the *RIP* controller. The core of the *RIP* controller is the *RIP* block whose transfer characteristic is purely static and in most cases non-linear.

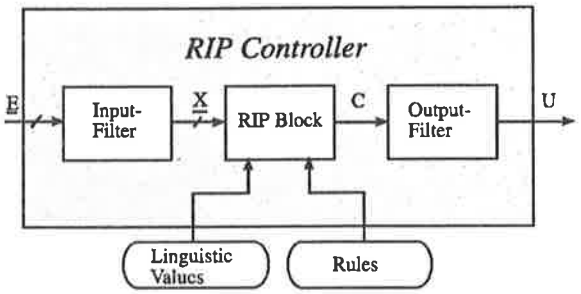


Fig. 1: *RIP* controller

The linguistic values and rules determine the mapping of the inputs $\underline{X}$ to the output C. A *RIP* block with m inputs and one output represents a mapping of a m-dimensional input vector $\underline{X} = (x_1, x_2, ..., x_m)$ to an output value and thus, defines a static function $C=F(\underline{X})$ by means of linguistic values and rules.

The basic principle of a *RIP* controller is explained with reference to the flow chart depicted in Fig. 2. A user of the *RIP* method will mainly be concerned with the definition of linguistic values and rules; the subsequent steps are automatically processed by the computer. The different steps are explained in detail in the following sections.

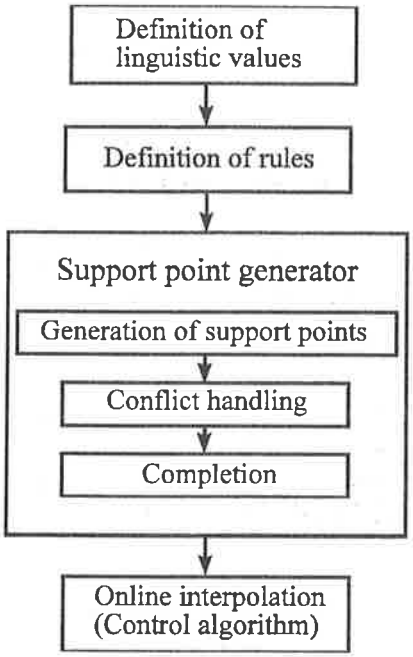


Fig. 2: Flow chart of a *RIP* controller

2.1. Linguistic values

Within *RIP* control, linguistic values represent crisp subsets or points (not fuzzy) of an universe of discourse. Thus, a linguistic value labels a classical subinterval of the universe of discourse.

As depicted in Fig. 3, a linguistic value for instance *cold*∈ [0, 15°C] can be defined as an interval of the universe of discourse of the summer temperatures. It is not required to define the linguistic values such that the universe of discourse is covered, entirely. Hence, merely points of the universe of discourse can be defined as linguistic values to obtain a complete mapping, too, as it will be explained later.

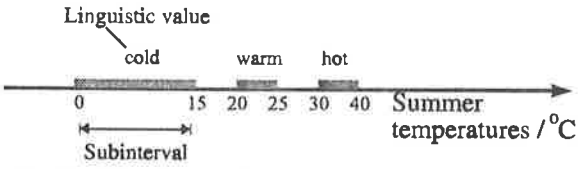


Fig. 3: Linguistic values

2.2. Rules

The rule structure of a *RIP* controller is similar to other knowledge-based systems which are based on production rules.

if {*premise*} *then* {*conlusion*} | *weight*

The premise consists of elementary expressions and logical operators. In the elementary expressions, the linguistic variables and values are connected by relational operators like == (equal) or != (not equal). The expressions are interpreted as logical expressions which are *true* or *false*. An elementary expression like X_i==*linguistic value* represents the mathematical expression $X_i \in \{subinterval\}$. Complex expressions are composed of elementary expressions and logical operators like for instance *and*, ∧, or *or*, ∨, as shown in Fig. 4.

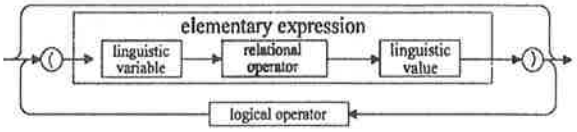


Fig. 4: Structure of the premise

Fig. 5 depicts the structure of the conclusion where linguistic values or functions are assigned to the linguistic variable of the output.

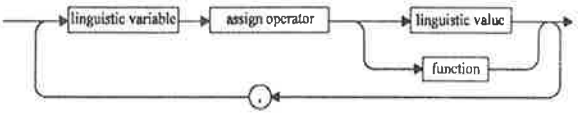


Fig. 5: Structure of the conclusion

In order to explain the *RIP* method, a rule base for the control of a ventilating fan is used as example.

R_1: *if* (*Temp*==*cold* ∧ *Breeze*==*strong*) *then* *Fan*=min | 1
R_2: *if* (*Temp*==*hot*) *then* *Fan*=max | 5
R_3: *if* (*Temp*!=*cold* ∧ *Breeze*==*strong*) *then* *Fan*=*med* | 1
R_4: *if* (*Temp*==*cold* ∧ *Breeze*==*weak*) *then* *Fan*=min | 1

2.3. Generation of support points

The premise of the rule defines sets of input domains within the input space $X=X_1 \times ... \times X_m$, as depicted in Fig. 6. The rules define a relation

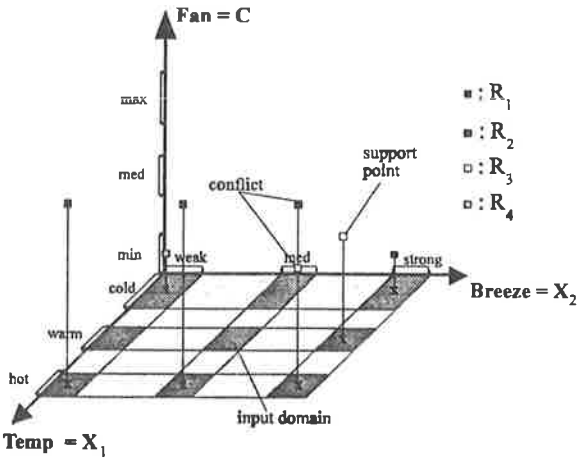


Fig. 6: Support point generation

between the input domains and the output interval of the considered conclusion. Thus, each rule defines hypercubes in the entire hyperspace. In order to obtain the mapping $C=F(\underline{X})$ between the input vector $\underline{X}$ and the output C, support points $(x^s_1, ..., x^s_m, c^s)$ are generated at the center of the hybercubes. Hence, each rule will be converted into a set of support points (Fig. 6).

2.4. *Conflict handling*

A conflict or a contradiction occurs in a rule base, if two or more different linguistic output values are assigned to a single input domain. Fig. 6 exhibits a conflict between the rule R_2 and R_3 for the input domain *(Temp == hot ∧ Breeze == strong)*. In order to resolve the conflict, a weighted average considering the weights G_i of the rules is employed. The weights G_i can be chosen within the interval $[0,\infty[$ to emphasize important rules, as in equ. 1.

$$C_j = \begin{cases} \dfrac{\sum_{i=1}^{n_j} G_i \cdot C_i}{\sum_{i=1}^{n_j} G_i} & \exists\ G_i:\ (G_i \neq 0), \\ \dfrac{1}{n_j} \cdot \sum_{i=1}^{n_j} C_i & \forall\ G_i:\ (G_i = 0). \end{cases} \quad (1)$$

n_j: Number of conflicting rules of input domain j.

In Fig. 7. the support points with the resolved conflict are depicted.

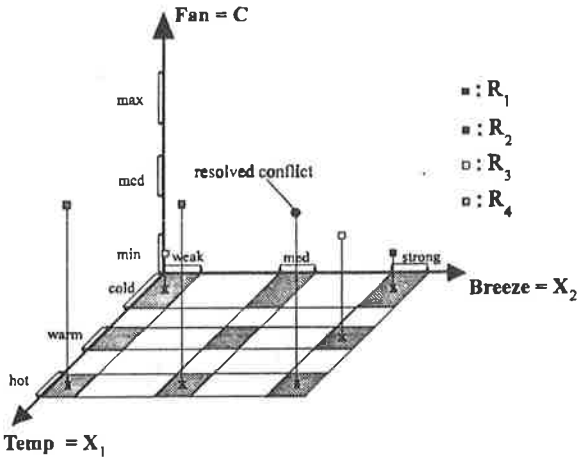


Fig. 7: Support points with resolved conflict

2.5. *Completion and global interpolation*

Until now, only support points have been defined and not a complete mapping of the inputs $\underline{X}$ to the output C. Completion is achieved by interpolation or extrapolation. It is appropriate to split the interpolation into two steps.

1. Global and scattered data interpolation.
2. Local and non scattered data interpolation.

The two main reasons to split the interpolation are:

1. Decision or rule tables should be complete.
2. Global and scattered data interpolations are time consuming. Thus, it's advantageous to compute them offline.

Global and scattered data interpolations can be applied to complete an incomplete grid of support points (also called a non-Cartesian grid) as depicted in Fig. 8.

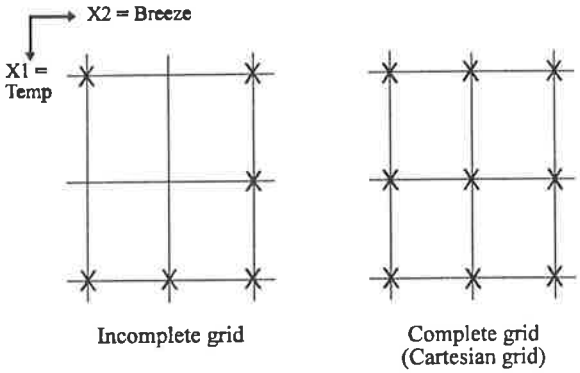


Fig. 8: Cartesian grid

In order to interpolate non-Cartesian grids of support points, global and scattered data interpolations are necessary. The interpolation methods must be applicable to the multidimensional space, since the *RIP* block can be fed by several inputs. In literature, hardly a few multidimensional interpolation methods are known which can be applied to achieve this. The Shepard (Shepard 68) in equ. 2 is one such which is especially suitable by virtue of relative easy implementation, modest storage demand and computation time.

$$S(\underline{X}) = \frac{\sum_{i=1}^{n_s} w_i \cdot C_i^s}{\sum_{i=1}^{n_s} w_i} \quad \text{with} \quad w_i = \frac{1}{\|\underline{X} - \underline{X}_i^s\|^\mu} \quad (2)$$

The Shepard interpolation $S(\underline{X})$ is also called inverse distance interpolation. The interpolated output value $C = S(\underline{X})$ of an input $\underline{X}$ is calculated by means of the inverse spatial distance w_i to the support points $(\underline{X}_i^s,$

C_i^s). The parameter μ can be used to modify the interpolation. For the purpose of completion, the parameter μ should be chosen relatively high and the universes of discourse of the inputs must be normalized to a standardized range. Fig. 9 shows the completed support point grid of the fan controller.

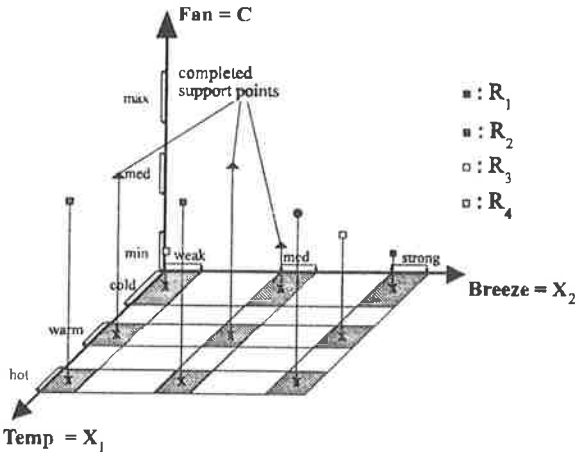


Fig. 9: Complete and conflict free support points

2.6. Control algorithm and local interpolation

Efficient local interpolation methods based on Cartesian grids are employed for the online computation of the output of the *RIP* block.

For control, the multilinear interpolation is especially suitable, since it demands very low computation time. In the case of two inputs, the interpolation formula is derived from Lagrange interpolation as in equ. 3, whereas n_r and n_c are equal to 1 (Engeln *et. al.*, 1988).

$$F(\underline{X}) = \sum_{i=0}^{n_r} \sum_{j=0}^{n_c} L_i^{(1)}(x_1) \cdot L_j^{(2)}(x_2) \cdot C_{ij}^s \qquad (3)$$

$$L_i^{(1)}(x_1) = \prod_{k=0,k\neq i}^{n_r} \frac{x_1 - x_{1_k}^s}{x_{1_i}^s - x_{1_k}^s}$$

$$L_j^{(2)}(x_2) = \prod_{k=0,k\neq j}^{n_c} \frac{x_2 - x_{2_k}^s}{x_{2_j}^s - x_{2_k}^s}$$

$F(\underline{X})$: Interpolated output value of $\underline{X}=(x_1,x_2)$,
$n_r n_c$: Number of rows or columns,
C_{ij}^s : Support point value (i : row, j : column).

If the *RIP* block is fed by m inputs, the local interpolation considers only 2^m support points in the neighbourhood of the Cartesian grid. Thus, the function $F(\underline{X})$ of the *RIP* controller is piecewise or boxwise multilinear; that is advantageous for control stability proofs like for instance in (Kiendl *et. al.*, 1993).

Fig. 10 depicts the surface of the function $F(\underline{X})$ of the fan controller rule base after the elimination of the conflict, the completion and the multilinear interpolation. It can be easily seen that the generated control surface is appropriate, even though the rule base is contradictory and incomplete.

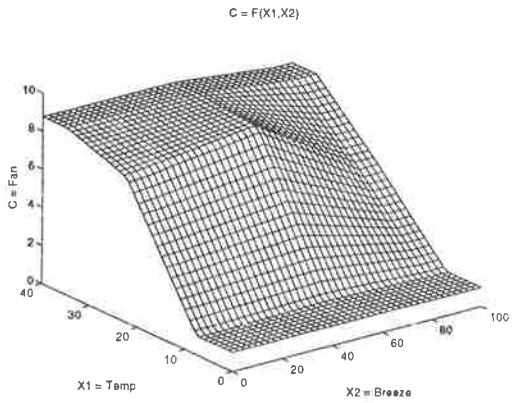


Fig.10: Surface of the fan controller rule base

3. *RIP* DESIGN SHELL

Fig. 11 depicts a *RIP* control design shell. The design shell is directly derived from the *RIP* method introduced above. Initially, the user of the *RIP* design shell has to define linguistic values. Then he can start to enter the expert knowledge into the rule editor by use of the linguistic values and considering the rule syntax. Subsequently, the support point generator converts automatically the rule base to support points. The support points determine the control surface and the mapping. They can be interpreted as the control parameters and the local interpolation as the control algorithm. There is no need for the designer of a controller to be familiar with the interpolation methods or the support point generator, since they are automatically computed by the *RIP* design shell. Optionally, the completion can be adjusted by the parameter μ of the Shepard interpolation.

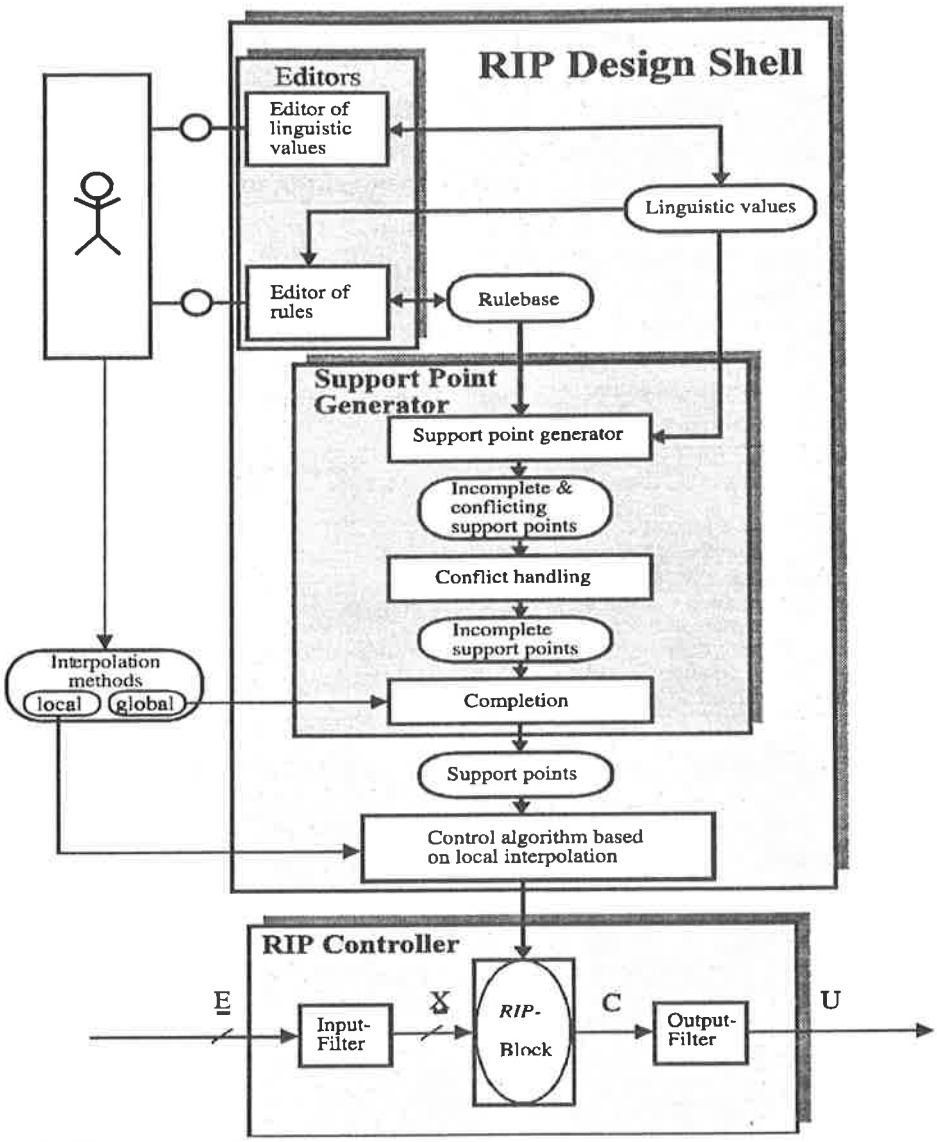


Fig. 11: *RIP* control design shell

4. HYBRID RULE BASE

Especially, for control applications, the *RIP* method enables the combination of classical control strategies with rule based strategies, like for instance:

R_1: *if* ( ) *then* $C=k_1 \cdot X_1 + k_2 \cdot X_2$ | 0

R_2: *if* $(X_1 == A_{x_1} \wedge X_2 == A_{x_2})$ *then* $C=B_c$ | 1

The parameters (k_1 , k_2) of the rule R_1 can be obtained from classical control design methods like for instance in (Ziegler *et. al.*, 1942), depending on the input & output filters of Fig. 1. The rule R_1 is valid for the entire input space, since the premise is unspecified (Fig. 4). Subsequently, rules like R_2 are introduced to optimize the control performance in a linguistic manner. Such a hybrid design incorporates several advantages.

- ☐ The presentation of the implemented control strategy is more transparent than listing all linguistic rules of a rule table.
- ☐ The number of rules can be tremendously reduced compared to the pure linguistic approach.
- ☐ The control performance can be altered by parameters like k_1 , k_2 and the experience of classical control design can be utilized.

It should be stressed that the conclusion function of a rule (Fig. 5) is only approximated by the underlying interpolation. Thus, the generated function $F(\underline{X})$ is equal to the conclusion function only in the support points. Nevertheless, a linear conclusion function is obtained for the entire input space, if multilinear interpolation is applied, as in equ. 3. Furthermore, in most of the practical control

applications, an approximated function is sufficient to achieve the desired control performance and the accuracy of the approximation can be improved by means of more linguistic values or support points.

5. RESULTS

In order to examine the properties of *RIP* control, a linear *PI* controller, a *RIP* controller and a fuzzy controller have been applied to a model of a non-linear flow process. The control performance of the various controllers is depicted in Fig. 12. The linear controller achieves an asymmetric control performance, since the process is non-linear. The fuzzy and the *RIP* controller can be designed and modified in a linguistic manner to compensate the effects of the non-linearity. The *RIP* controller has been designed by a hybrid rule base with only two rules, as described in section 4. The fuzzy controller has been achieved by 25 rules to eliminate the overshoot. This application is described in more detail in (Drechsel *et. al.*, 1994).

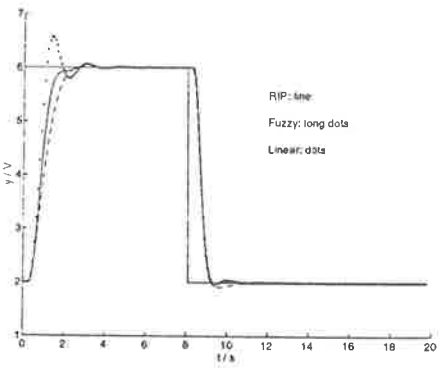


Fig. 12: Control performance

6. CONCLUSION

In this paper, *RIP* control has been introduced. It has been demonstrated that *RIP* control offers an appropriate approach to convert linguistic expert knowledge into a control algorithm. *RIP* methods seems to be more transparent for control design than fuzzy methods, since the degree of freedom is lower and the effects of parameter modifications are easier to assess. The application of the *RIP* method for flow control shows that it is a viable alternative to fuzzy methods. In particular, the use of hybrid rule bases facilitates a control design by combining classical and rule based control technics. The developed *RIP* design shell can be employed to a wide area of knowledge-based applications which need not necessarily pertain to control.

7. REFERENCES

Braae M., Rutherford D. A., (1979); "Theoretical and Linguistic Aspects of the Fuzzy Logic Controller"; Automatica, Vol. 15, pp. 553-577.

Drechsel D., Stauch W., Pandit M., (1994); "*RIP* Control im Vergleich zu Fuzzy Control"; VDE/VDI, Aussprachetag Fuzzy Control, Langen, pp. 41-55.

Drechsel D., Stauch W., Pandit M., (1993); "Fuzzy Control im Vergleich zu einem regelbasierten Interpolations-Controller (*RIP*)"; 3. Workshop Fuzzy Control, GMA-UA 1.4.2, Dortmund, pp. 127-140.

Engeln-Müllges G., Reuter R., (1988); "Formelsammlung zur nummerischen Mathematik mit Modula 2 Programmen"; B.I. Wissenschaftsverlag.

Franke R., (1982); "Scattered Data Interpolation: Test of Some Methods"; Mathematics of Computations, Vol. 38, Nr. 157; pp. 181-200.

Hoschek J., Lasser D., (1989); "Grundlagen der geometrischen Datenverarbeitung"; Verlag B.G. Teubner Stuttgart.

Kiendl H., Rüger J. J., (1993); "Fast Real-time Controller Realization and Computer-aided Proof of Stability"; Proc. Eufit'93; Aachen, pp. 124 -129.

Press W. H., Flannery B. P., Teukolsky S. A. Vetterling, W. T., (1990); "Numerical Recipes, The Art of Scientific Computing"; Cambridge University Press.

Rajaraman V., (1991); "Decision Tables"; Encyclopedia of Computer Science and Technology; Editors: A. Kent, J. G. Williams; Vol. 24, Sup. 9; pp. 85-106.

Shepard D., (1968); "A two-dimensional interpolation function for irregularly-spaced data"; AC National Conference, pp. 517-524.

Wendt S., (1989); "Nichtphysikalische Grundlagen der Informationstechnik"; Springer-Verlag.

Ziegler H. G., Nichols N.B., (1942); "Optimum Settings for Automatic Controllers." Transactions of the A.S.M.E., Nov., pp. 759-768.

Zadeh L. A., (1973); "Outline of a new Approach to the Analysis of Complex Systems and Decision Processes"; IEEE Transactions on Systems, Man, Cybernetics; Vol. 3 No. 1, Jan., pp. 28-44.

PROCESS CONTROL USING RECURRENT NEURAL NETWORKS

T. CHOVAN*, T. CATFOLIS and K. MEERT****

**University of Veszprem, Dept. of Chemical Engineering Cybernetics, Egyetem utca 10, P.O. Box 158, H-8201 Veszprem, Hungary*
***K. U. Leuven, Dept. of Chemical Engineering, Expert Systems Applications Development Group, de Croylaan 46, B-3001 Heverlee, Belgium*

Abstract. Neural network based control schemes are generally designed by implementing feedforward neural network models in standard control engineering structures. Introduction of discrete time recurrent networks, which are inherently dynamic systems, into those schemes can simplify the design of neural controllers. In this paper we describe the concept of applying recurrent networks trained with the real-time recurrent learning algorithm in the indirect adaptive control schemes. A combined network consisting of the control network and the model network is constructed to allow the simple use of the real-time recurrent learning algorithm. To demonstrate the feasability of the method two simulation examples are presented.

Key Words. Process control, Neural nets, Control system design, Artificial intelligence, Real-Time Recurrent Learning algorithm

1. INTRODUCTION

Since the eighties, artificial neural networks have been intensively studied with special regard to engineering applications. Techniques based on neural networks have been developed in several fields, ranging from banking and speech recognition to processor scheduling. Advantageous properties of neural networks, such as parallel computation, nonlinear mapping and learning capabilities make them an attractive solution in many chemical engineering applications. Chemical engineering systems are typically nonlinear with complex dynamics and our knowledge - and models - of them are often defective or uncertain.

Typical chemical engineering applications based on the excellent classification properties of some neural networks include diagnosis of chemical plants (Watanabe *et al.*, 1989; Catfolis, 1993b) and controller adaptation (Cooper *et al.*, 1992). Based on the nonlinear mapping capabilities, problems like the identification, simulation and control of chemical processes have been studied (Bhat *et al.*, 1990). Neural controllers have been used in robotics (Sanderson, 1990) and in chemical industry (Hangos, 1992).

In the second section the basic artificial neural networks architectures are reviewed, with an emphasis on the RTRL algorithm. In section three some classical control schemes with neural networks are described and the use of the clustered RTRL algorithm in control schemes is introduced. In the last section the feasability of this method is demonstrated by two examples.

2. NEURAL NETWORKS ARCHITECTURES

Numerous types of networks exist, but each type consists of the same basic features, nodes, layers and connections. The smallest element of a network is the node. Every node receives signals from connections, which it processes in a combination function and converts to an output signal with a transfer function. These nodes are connected to each other by weighted links. These weights can be adapted by one or other learning rule and represent the "long term memory" of the neural net. Different network topologies and learning methods are used depending on the problem to be solved. Learning or training of the neural network means finding a set of parameters, i.e. weights, that produce the desired behaviour. In the following part some network architectures are described. Networks can be divided into two main classes by their topologies: feedforward networks which do not contain any directed loop in their representing graph and recurrent networks which, on the contrary, do.

2.1. Feedforward Networks

Until now, feedforward neural networks hae been the more frequently used models. Their most typical form is the *multilayer network*. They basically give a static mapping of their inputs onto their outputs. It can be proved (Hornik *et al.*, 1989) that a network with two hidden layers can approximate any nonlinear function with arbitrary precision. The identification of the connection weights, i.e. the learning, is usually

performed using the *backpropagation* algorithm which is basically a form of the gradient descent method (Rumelhart *et al.*, 1986). To apply multilayer feedforward neural networks for modelling the system dynamics, a discrete time history of system inputs and outputs can be used as network input and target values. The network works as a nonlinear mapping corresponding to a nonlinear form of the input-output model, widely used in control engineering and signal processing. The formation of the mapping is performed by the learning process using a number of training patterns consisting of input vectors and the corresponding target values.

2.2. Recurrent Networks

A neural network becomes a dynamic system when feedback connections are introduced. The network operation can be described with differential equations of the nodes in case of continuous time networks, whereas in case of discrete time networks, difference equations can be used (Narendra *et al.*, 1990). Several models and learning methods of recurrent networks are used. *Discrete time recurrent networks* are built on a synchronous computing scheme, i.e. activities of all nodes are evaluated using the current inputs of the network and the current outputs of the neurons and then all new outputs are calculated and take effect. Fully connected recurrent neural networks probably have the most general topology. They can represent any feedforward or other, simpler recurrent structure. However, a discrete time delay (the minimum delay depends on the actual implementation of the algorithm) emerges due to the structure of the network. This time delay allows identification of process delays, which are usually assumed to be known a priori in the feedforward schemes, to be readily solved in recurrent networks if the number of neurons is large enough. For fully connected recurrent networks, Williams and Zipser (Williams *et al.*, 1989) derived a learning algorithm based on the gradient descent method. Since the weights are updated at each time step rather than at the end of a trajectory, the method, called *real-time recurrent learning* (RTRL), is well suited for on-line training. Although the RTRL algorithm provides a simple computation scheme it has several disadvantages. The main problem is that the calculation is not local in the sense that it requires the use of all weights and activities in the network for the evaluation of each one of the nodes. Another problem is that the RTRL algorithm shows very slow convergence in many cases. A method for speeding up learning is demonstrated by Catfolis (Catfolis, 1993a). It is based on resetting the learning algorithm with a frequency which can be related to the time constants of the process.

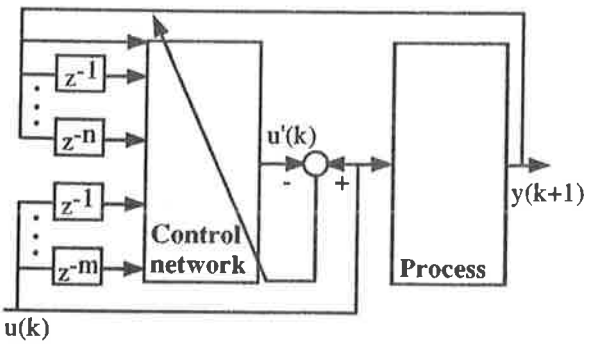


Fig. 1. Direct inverse control using feedforward networks

3. NEURAL NETWORK BASED CONTROL ARCHITECTURES

The application of neural networks for control is expected to make the achievement of the following capabilities feasible (Werbos, 1989):

-the implementation in parallel hardware
-the real-time adaptation without instability
-the handling of severe nonlinearity and noise
-the planning or optimization over time

In the next two parts some classical control schemes in which neural networks are used, are reviewed . In the third part the concept of the new control architecture is explained.

3.1. Classical Architectures

Neural network based control schemes are generally devised by using neural networks in the position of linear models in the standard control engineering structures. Possible architectures for control are discussed in several papers (Tanomaru *et al.*, 1992; Levin *et al.*, 1991; Narendra *et al.* , 1990; Werbos, 1989). Since the neural model has a nonlinear character, the analysis of the stability and the robustness poses several problems and a great deal of present research efforts is dedicated to finding a solution to these problems. In most cases, the neural network used in control schemes is a multilayer feedforward network. The input of the network is formed using a process input and process output history. The assignment of inputs and outputs is determined according to the requirements of the actual problem. The schemes can easily be adapted to multi-input - multi-output cases by applying the time window to all relevant process inputs and outputs. Most frequently used schemes are based on *supervised learning*. In *supervised control*, neural networks are trained to map the input sensor signals onto the desired actions which are given by a human expert. This kind of solution can be used to train controllers for tasks which can be successfully solved by human operators. *Direct inverse control* is based on the process input and output signals. The training scheme is shown in Fig. 1. This method can be used for both on- and off-line training. Since it heavily relies on the generalization ability of the neural

network, it is not advisable to use direct inverse control as the only training scheme. *Direct adaptive control* adjusts the controller parameters (weights) according to the error on the process output (see Fig. 2). The method requires the calculation of the sensitivity of the error measure with respect to the process inputs, which assumes the knowledge of the Jacobian of the process.

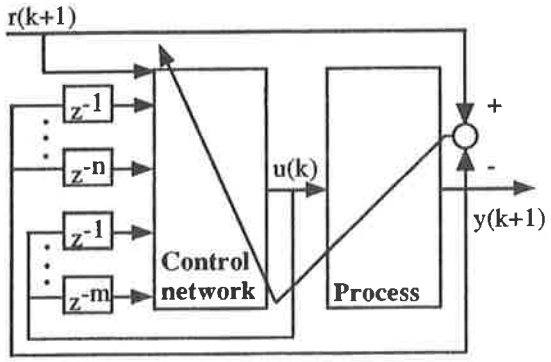


Fig. 2. Direct adaptive control using feedforward networks

Indirect adaptive control employs a process model, in this case a model network. The output error of the model is then used in a standard backpropagation algorithm to pass the error back to the controller output. The adaptation mechanism is twofold since the neural process model is adapted to the true process output as target value. The indirect adaptive control scheme is shown in Fig. 3. Learning in both the direct and the indirect schemes are essentially performed online. Two methods based on *reinforcement learning* are the backpropagation through time and the adaptive critics (Werbos, 1990; Sofge *et al.*, 1990). *Backpropagation through time* calculates the derivative of future utility or performance measure with respect to present actions using an explicit model of the environment. The *adaptive critics* method adapts a special "critic" network which estimates the future utility arising from present actions. It is basiccally an approximate of dynamic programming methods.

3.2. Architectures using Recurrent Networks

When using feedforward networks for modelling process dynamics, a careful design of history windows of process inputs and outputs is necessary. The form of the model requires (basically an input-output model) the application of external feedback loops to the network which is not accounted for in the open loop learning process. Recurrent networks offer a solution to these difficulties since they can develop through learning an internal representation of time history due to their feedback connections. When recurrent networks are used, neural control schemes become significantly simpler since only the current process inputs and outputs are required instead of their time

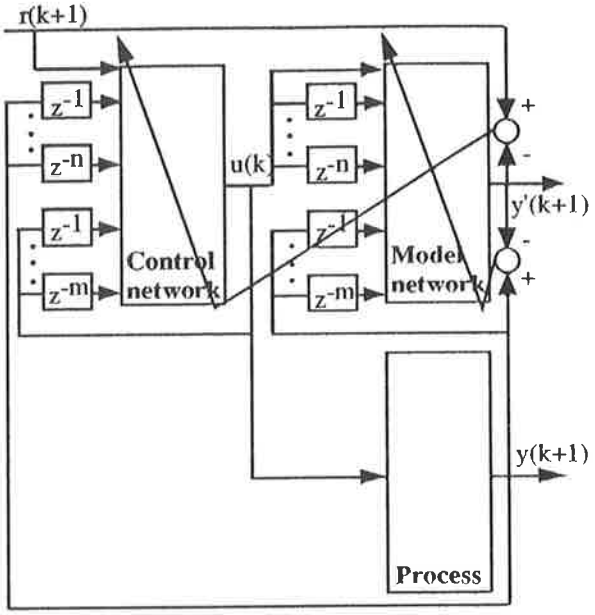


Fig. 3. Indirect adaptive control using feedforward networks

history. This also reduces the number of input connections and results in a smaller computation cost in the controller operation. A major drawback is the amount of computation time needed during the training phase. Fig. 4 shows the application of recurrent networks in the indirect adaptive control scheme.

3.3. Method using RTRL in Clustered Networks

The training of the fully connected recurrent control network requires the knowledge of the controller output error, which is generally not available. In the indirect scheme the controller error is calculated by propagating the output error of the model network back to the model input. Although this can easily be done in

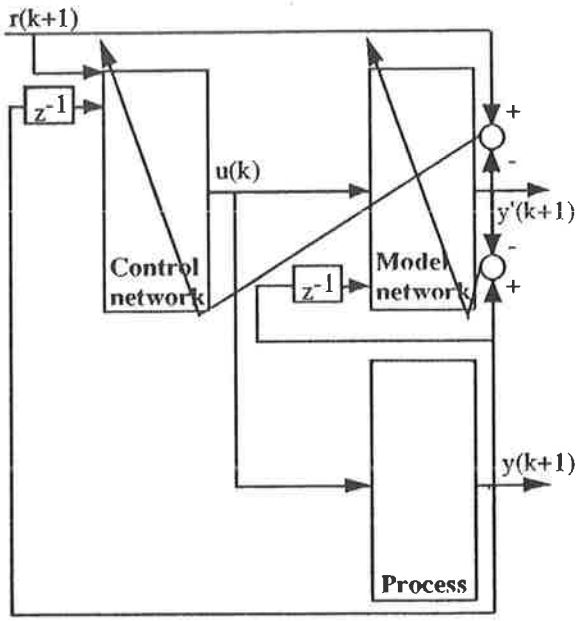


Fig. 4. Indirect adaptive control using recurrent networks

feedforward networks, by using the standard backpropagation algorithm, it cannot be solved in a simple way in case of recurrent networks. A direct error backpropagation algorithm cannot be derived using RTRL. The method suggested here applies a combined network composed of the controller and the model clusters. The output node of the controller is identical to the corresponding input node of the model network. Such a clustered network is demonstrated in Fig. 5. for the level control system presented in Section 4.

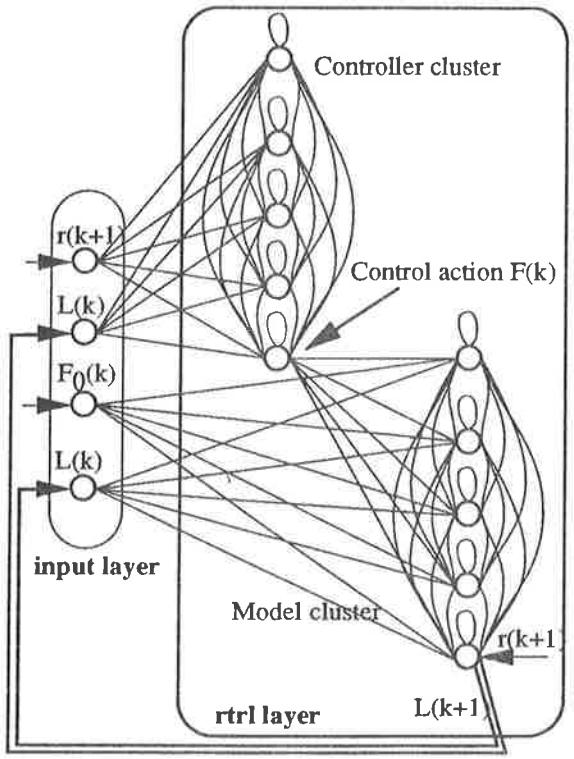


Fig. 5. Construction of the clustered network

Both the control and the model clusters are five-node fully connected recurrent networks. The simplest way to apply the RTRL algorithm is by giving zero weights to the non existing connections and assign a non adaptable status to them like to the weights in the model network. The algorithm can also be modified specifically for training the controller part of the cluster using only the existing connections. Based on the application of the network cluster, the indirect adaptive control method using recurrent networks consists of the following steps:

1. Training of the recurrent model network.
2. Construction of the untrained control network.
3. Composition of the network cluster.
4. Training of the control network.
5. Adaptation of the model based on the error.

4. SIMULATION STUDIES OF RECURRENT NEURAL CONTROLLERS

The application of recurrent neural networks in the indirect scheme was studied by simulating two processes. The cluster network method described above was applied using the standard RTRL algorithm as presented in Section 3. The model networks were trained to give a one-step-ahead prediction of the process dynamics. Then the on-line training of the control network in the inverse adaptive scheme was investigated.

4.1. Example 1. Level Control System

The first problem is the level control in a tank in the presence of an external disturbance (input flow rate). The control variable is the output flow rate. The scheme of the system is shown in Fig. 6.

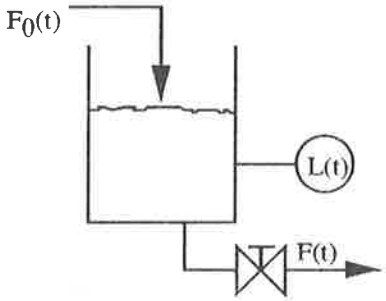


Fig. 6. The tank system

This simple linear example was selected to demonstrate the use of the inverse adaptive scheme with recurrent networks. The mathematical model of the tank is the following

$$A\frac{dL(t)}{dt} = F_0(t) - F(t) \qquad (1)$$

where $L(t)$ is the level in the tank, $F_0(t)$ is the input flow rate, $F(t)$ is the output flow rate and A is the cross section of the tank. A small simulation program applying the Euler method for the integration of the

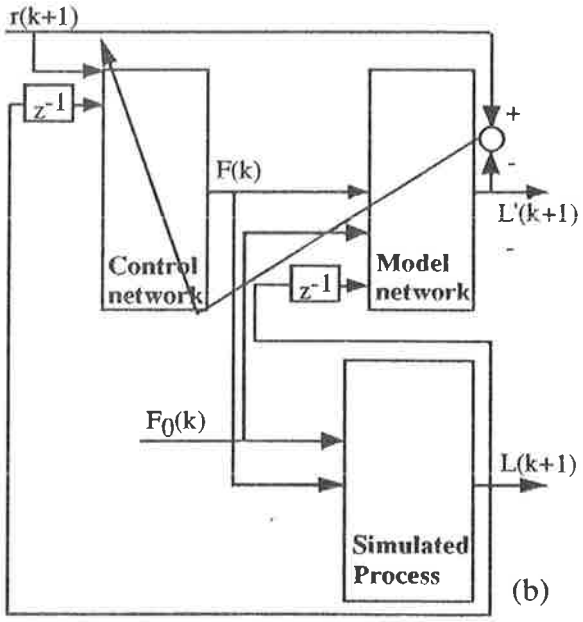


Fig. 7. The tank system training scheme

model was used for training the model network and then for training the controller network through the inverse adaptive scheme. A good average error (0.01) of the output of the neural model was achieved by a five-node network and by giving random sequences on inputs $F_0(k)$ and $F(k)$. The actual configuration of the indirect adaptive control scheme is presented in Fig. 7. Relatively slow convergence and sensitivity on the learning rate were observed during the controller training. A significant improvement in convergence was achieved by using the "clear impacts" method (Catfolis, 1993a). Control networks of 4, 6 and 8 neurons were trained and even the smallest one provided a reasonable control performance. The behaviour of the controller with 4 neurons is shown in Fig. 8. Responses to both setpoint changes and external disturbance $F_0(t)$ are demonstrated. The control offset is primarily due to the error of the model network.

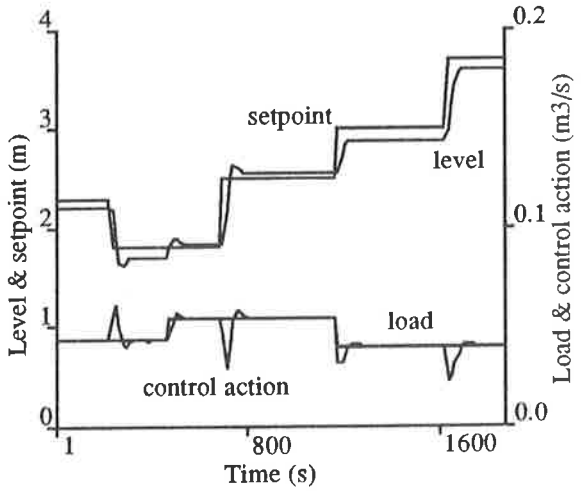


Fig. 8. Operation of the controller for the level control system

4.2. Example 2. Control of a Bioreactor

The second problem, a bioreactor model, was suggested by Ungar (Ungar, 1990) to be used as a benchmark problem for neural controllers. It is a relatively simple problem having only a few variables, however it exhibits a difficult control problem due to its strongly nonlinear character. The system is a continuous flow stirred tank reactor with nutrient being fed to it. The control target is the cell mass yield. The volume in the tank and the flow rate through the tank is assumed to be constant. The scheme of the biorector is given in Fig. 9. The mathematical model of the system gives an account of both the concentration of the cell mass and of the substrate:

$$\frac{dC_1(t)}{dt} = P(t) - C_1(t)w(t) \qquad (2)$$

and

$$\frac{dC_2(t)}{dt} = P(t)\frac{1+\beta}{1+\beta-C_2(t)} - C_2(t)w(t) \qquad (3)$$

where

$$P(t) = C_1(t)(1-C_2(t))e^{C_2(t)/\gamma} \qquad (4)$$

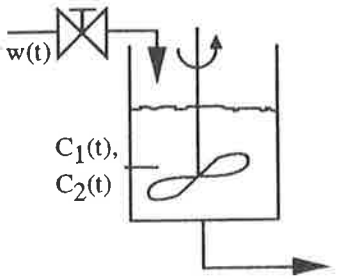


Fig. 9. The bioreactor system

and where $C_1(t)$ and $C_2(t)$ are, respectively the dimensionless cell mass and substrate conversions, $w(t)$ is the substrate feed flow rate, β and γ are rate coefficients. The process model was used in the same way as in Example 1. However, only one of the system outputs, the cell mass conversion $C_1(t)$ was represented by the model network since the other was not required for the controller training scheme - Fig. 10. Fully connected recurrent networks of 6, 8, 10 and 15 nodes were trained as one-step-ahead prediction models. Applying different learning rates and random inputs, the best solution of 8 neurons gives an average error of 0.001 (after 300,000 training cycles) on the output, only with a slight dependence on the number of neurons in the range studied. Sudden changes in the average error and high sensitivity on the learning rate were experienced in the course of controller training. Several

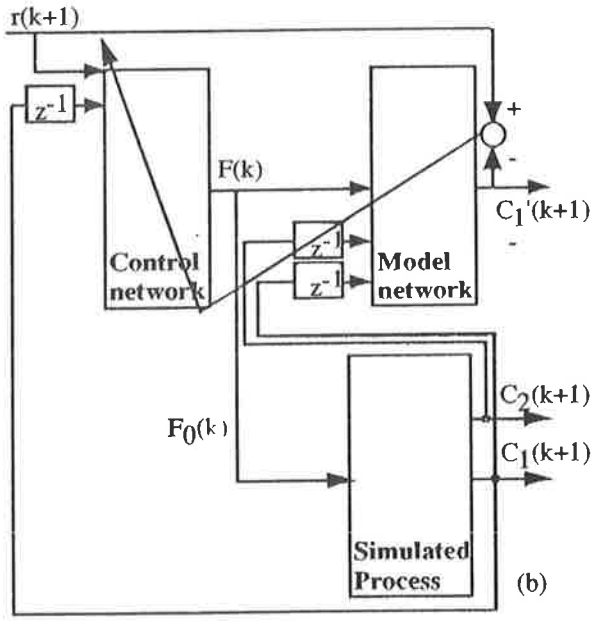


Fig. 10. The training scheme for the bioreactor system

training experiments and a continuous manual adjustment of the learning rate were required to get a reasonably good control network. The behaviour of the controlled system achieved by an 8 neuron control network is shown in Fig. 11. The control offset can also be attributed to the error of the neural process model as in Example 1. In both cases the application of Step 5 of the present indirect adaptive control pro-

cedure (see Section 3.3), i.e. adaptation of the model network, can help in solving the problem.

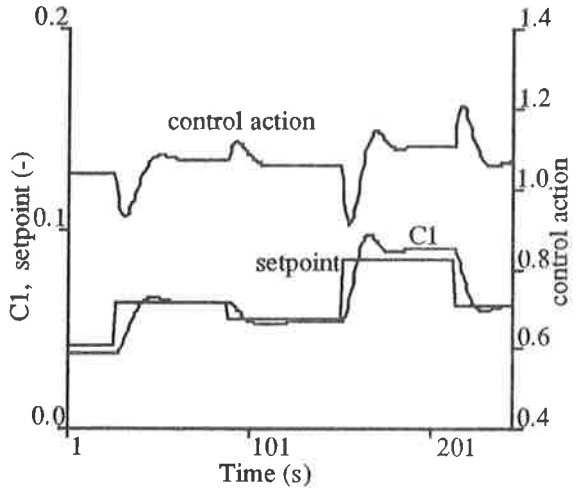


Fig. 11. Operation of the controller for the bioreactor system

5. CONCLUSIONS

Application of recurrent networks in neural control architectures offers the benefit of internal representation of system dynamics and hence a simpler design of the controller. The method described here is based on the real-time recurrent learning algorithm and uses a network cluster constructed from the control network and the model network in the inverse adaptive control scheme. The simulation experiments demonstrate the feasibility of the concept for controller training. However, further studies are needed to check the performance of this kind of neural controllers and to compare it to standard controllers. The control offset observed in both examples is primarily due to the error of the neural process model and is expected to be lowered by adapting the model networks too.

6. ACKNOWLEDGEMENTS

This research was done at the Expert Sysems Applications Development Group, headed by Prof. M. Rijckaert. This work was partially supported by the E.C. project "Cooperation in Science and Technology with Central and Eastern Europe" and by the OTKA 2547 project (Hungary).

7. REFERENCES

Bhat, N. and McAvoy, T.J. (1990) "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process Systems", *Computers and Chemical Engineering*, **14**, 4/5, 573-583.

Catfolis, T. (1993a) "A Method for Improving the Real-Time Recurrent Learning Algorithm", *Neural Networks*, **6**, 807-821

Catfolis, T. (1993b) "Monitoring a Control System with a Hybrid Neural Network Architecture", *Proceedings of the International Conference on Artificial Neural Networks '93*, p. 854.

Cooper, D.J., Megan, L. and Hinde, R.F. (1992) "Disturbance Pattern Classification and Neuro-Adaptive Control", *IEEE Control Systems*, April 1992, 42-48.

Hangos, K. (1992) "Application of Neural Networks in Chemical Process Control and Diagnostics", in *Report SCL-001-1992*, MTA SZTAKI, Budapest, 42-53 (in Hungarian).

Hornik, K., Stinchcombe, M. and White, H. (1989) "Multi-layer Feedforward Networks are Universal Approximators", *Neural networks*, 2, 359-366.

Levin, E., Gewirtzman, R. and Inbar, G.F. (1991) "Neural Network Architecture for Adaptive System Modeling and Control", *Neural Networks*, **4**, 185-191.

Narendra, K.S. and Parthasarathy, K. (1990) "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Transactions on Neural Networks*, **1**, 1, 4-27.

Rumelhart, D.E, Hinton, G.E. & Williams, R.J. (1986) Learning Internal Representation by Error Propagation. in D.E. Rumelhart, J.L. McClelland and the PDP Research group (eds.) *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, Cambridge, MA.

Sanderson, A.C. (1990) "Applications of Neural Networks in Robotics and Automation for Manufacturing" in W.T. Miller, R.S. Sutton and P.J. Werbos (eds.) *Neural Networks for Control.*

Sofge, D.A. and White, D.A. (1990) "Neural Network Based Process Optimization and Control" in *Proceedings of 29th Conference on Decision and Control*, Honolulu, Hawai, 3270-3276.

Tanomaru, J. and Omatu, S. (1992) "Process Control by On-Line Trained Neural Controllers", *IEEE Transactions on Industrial Electronics*, **39**, 6, 511-521.

Ungar, L.H. (1990) "A Bioreactor Benchmark for Adaptive Network-based Process Control" in W.T. Miller, R.S. Sutton and P.J. Werbos (eds.) *Neural Networks for Control.* MIT Press, Cambridge, MA.

Watanabe, K., Matsuura, I., Abe, M., Kubota, M. and Himmelblau, D.M. (1989) "Incipient fault Diagnosis of Chemical Processes via Artificial Neural Networks", AIChE Journal, 35, 11, 1803-1812.

Werbos, P.J. (1989) "Neural Networks for Control and System Identification" in Proceedings of 28th Conference on Decision and Control, Tampa, Fl, 260-265.

Werbos, P.J. (1990) "A Menu of Designs for Reinforcement Learning Over Time" in W.T. Miller, R.S. Sutton and P.J. Werbos (eds.) *Neural Networks for Control.* MIT Press, Cambridge, MA.

Williams, R.J. and Zipser, D. (1989) "Experimental Analysis of the Real-Time Recurrent Learning Algorithm", *Connection Science*, **1**, 87-111.

FUZZY ANTI-RESET WINDUP FOR HEATER CONTROL

A. Hansson[†] **P. Gruber**[‡] **J. Tödtli**[‡] **P. Ries**[‡]

[†]*Department of Automatic Control, Lund Institute of Technology, LUND, Sweden*

[‡]*Landis & Gyr Betriebs A.G., ZUG, Switzerland*

Abstract In this paper a fuzzy anti windup scheme for an observer for the reference value of the flow temperature of a heating installation is designed. The observer consists of an averager in series with a characteristic curve. A nominal design of the time constant of the averager is done by considering only the heater dynamics. When there are large load disturbances or set point changes, the reference value runs away from the heater due to state and actuator constraints of the heating installation. To avoid this fuzzy anti windup is considered. The resulting scheme has only one parameter to tune. To obtain a good value of this parameter simulations are performed. It is seen that the scheme is robust with respect to the parameter, and thus it is easy to find a good value. Further the simulations show that the performance is improved when anti windup is used. A similar scheme has successfully been incorporated in the heating controller Sigmagyr RVP110.

Keywords Anti-Reset Windup, Fuzzy Control, Heater Control, Conditional Integration

1. INTRODUCTION

In this report part of the design procedure for the heating controller Sigmagyr RVP110 will be described, see Figure 1. This controller is manufactured by Landis & Gyr AG, and is used mainly for small gas and oil heaters which are installed in single family homes and residential flats. This product covers the lower range of temperature control applications. Simple hydraulics and few sensors are typical for this market segment. Figure 2 shows a standard heating installation for which the controller can be used. The controllers are delivered to the original equipment manufacturer of the heaters which build them into their products. The controller has several modes of operation: the setpoint of the water temperature of the heater can either be driven by the outdoor temperature or by the load or by both. The mode discussed in this report is the load driven one. This mode can only be used, if the radiators installed in each room are equipped with temperature controllers acting on the valve positions. For this case no outdoor sensor is needed.

High performance control is always of interest. Usually this is not accomplished by means of only sophisticated analysis and design, but also expensive measurements are needed. This is due to the fact that it is difficult to design observers whenever the process to be controlled is non-linear or has state or actuator constraints. One of the more apparent constraints in heater control is the maximal power that the burner can deliver. In Sigmagyr RVP110 an observer for the setpoint of the flow temperature is used. It will be seen that it is possible to obtain sufficiently high performance without expensive measurements. To this end fuzzy logic will be used. However it must be stressed that without traditional analysis tools such as the method of harmonic balance it would not have been possible to find a good observer.

2. CONTROL PROBLEM

In this section the control problem will be described. The control objective is to find a cheap solution with few measurements but which still gives as high performance as possible.

The process to be controlled is a heating installation in a house, see Figure 2. The heater water temperature is controlled by means of a relay acting on the burner, and each room is equipped with a temperature controller acting on the valve position of the radiators. There is no mixing of the return water from the radiators with the heater water. All return water goes directly back to the heater. Thus the temperature of the water to the radiators, i.e. the flow temperature, is controlled by controlling the temperature of the heater.

The reference value for the flow temperature can be obtained as a feed-forward from the outdoor temperature via some characteristic curve, or it can be set to a constant value. The feedforward solution gives higher performance. Here a cheaper solution with no outdoor temperature measurement is considered which still gives high performance. The

Figure 1. Heating controller Sigmagyr RVP110.01.

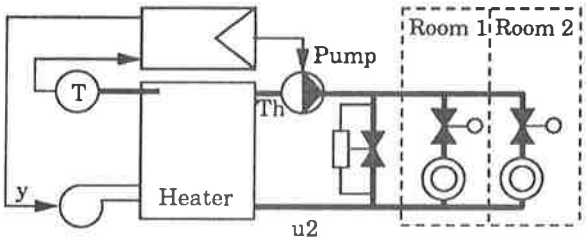


Figure 2. The heating installation with controller.

idea is to build an observer for the flow temperature needed in order to keep the valves of the radiators in operational range as good as possible, i.e. to prevent them to saturate. To make the solution cheap only measurements of the flow temperature and indirectly also of the heater on-off signal are used in the observer.

The observer is composed of two parts. This is due to the fact that it is possible to prove that there is a certain relation, characteristic curve, between the power demand in stationarity and the flow temperature needed in order to keep the radiator valves 80 % open in average. Thus one part is the characteristic curve, and the other part is an averager for computing a signal proportional to the

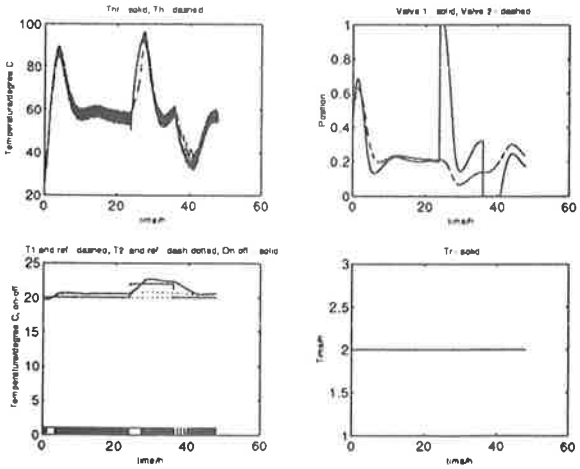


Figure 3. Simulation with a constant value of the time constant T_r = 2 h.

stationary power demand from the heater on-off signal. The characteristic curve is dependent on the house characteristics, and can be obtained in a similar way as the traditional curve from outdoor temperature. The averager is a transfer function of first order with unit stationary gain and a time constant T_r to be chosen.

The primary control objective for the heater control is to keep the valves of the radiators in operational range, such that the valves are able to increase or decrease the water flow when there are control errors in the room temperatures. The secondary goal is to speed up the temperature control in the rooms by also increasing the flow temperature when the valves open, and decreasing it when the valves close. This may of course be accomplished by choosing a small time constant of the averager. However, too small a time constant may cause undesirable oscillations, see Figure 3.

The nominal design of the averager will be done by only considering the dynamics of the heater. This is advantageous, since these dynamics are well known. Then some fix, i.e. anti windup, is needed in order to cope with the less well-known house dynamics whenever these interfere with the desired performance. This will be described in Section 4. Here only the motivation for the anti windup will be given. There are different types of constraints that can cause detoriation of performance. These can be classified into two groups—hard constraints and soft constraints. Examples of the former ones are saturations of control signals and limitations of state variables within the process. Examples of the latter ones are non-linearities, less abrupt than the previous ones, and unmodeled dynamics. Both types mentioned above are present in the heater control problem. The limitation of the heater power and the limitation of the water flow are hard constraints, whereas the non-linearities of the heater as well as the unmodeled house dynamics are soft constraints. As is seen in Figure 3 these constraints do not interfere with the performance in stationarity. However, as soon as there is a load disturbance or a change of reference value sufficiently large, the constraints result in bad performance. In Section 3 the nominal design considering only the heater dynamics will be done, and then in sections 4 and 5 an anti windup scheme will be designed to cope with the constraints.

3. ANALYSIS

In this section analysis of the controlled heater will be presented. Some guidelines for how to chose the averager will be given. Since these guidelines are only based on the heater model which does not include the house dynamics, the final design of the averager has to be done iteratively using simulations. This will be described in Section 5.

3.1 *Model*

The controlled heater with reference value generated via the characteristic curve is described in Figure 4. The idea is that the output $\bar{y}$ of the averager

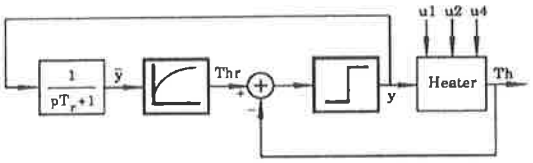


Figure 4. The controlled heater with reference value generated via the characteristic curve.

$$\frac{1}{pT_r + 1} \tag{1}$$

is proportional to the static power demand. Using the characteristic curve

$$T_h^r = 22 + 48\left(\frac{\bar{y}}{1-\xi}\right)^{1/1.3}, \quad \xi \in [0.02, 0.7] \tag{2}$$

for the heater temperature will give a reference value T_h^r for the heater related to this power demand, which then is used in the on-off control of the heater. The parameter ξ is used to adjust the curve to the specific house that is heated. The averager in series with the almost linear characteristic curve may be approximated by

$$T_h^r = T_0 + \frac{K_r}{pT_r + 1} y \tag{3}$$

where y is the heater on-off signal—0 for off and 1 for on. This approximation will be used in the sequel. It can be shown that $T_0 \in [28, 39]$ and that $K_r \in [44, 110]$ for $\xi \in [0.02, 0.7]$.

The differential equations for the heater can be summarized in

$$\begin{aligned} \frac{dx}{dt} &= A(u_1)x + B(u_1)u \\ T_h &= Cx \end{aligned} \tag{4}$$

for some $A(u_1)$, $B(u_1)$, and C. The states $x = (x_1\ x_2)^T$ are the temperatures of the heater wall, and the heater water, the same as the flow temperature. The inputs u_1 and $u = (u_2\ u_3\ u_4)^T$ are the water flow, the return temperature, the heater on-off signal, and the temperature outside the heater. Note that $u_3 = y$. Assuming that the water flow u_1 is constant, the differential equations describing the heater dynamics are linear, and it is possible to define the transfer functions relating the inputs u to the output T_h, i.e. to the heater water temperature. Simple calculations will give

$$T_h = G_2(p;u_1)u_2 + G_3(p;u_1)u_3 + G_4(p;u_1)u_4 \tag{5}$$

Some calculations show that $G_2(0;u_1)$ is approximately 1 , that $G_3(0;u_1)$ is varying drastically with the flow, and that $G_4(0;u_1) < 0.6$. Thus, since u_2 and u_4 are normally varying slowly, a good approximation to Equation 5 for values of u_1 in the range of interest is

$$T_h = u_2 + G_3(p;u_1)u_3 \tag{6}$$

This approximation will be used in the following analysis.

Using the approximate model in (6) and assuming that the characteristic curve is given by (3) the controlled heater with averager can be described as in Figure 5 where

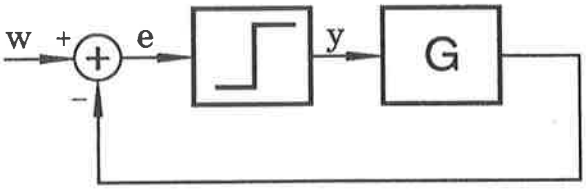


Figure 5. Approximate model of the controlled heater.

$$G(p) = G_3(p;u_1) - \frac{K_r}{pT_r + 1} \tag{7}$$

and where

$$w = T_0 - u_2 \tag{8}$$

This description will be used later on in Section 3.3.

3.2 *Transient Behavior*

It is obvious that for the averaging scheme to work, the averager must in some sense be slower than the heater, otherwise it will run away from the heater. In Figure 6 the step responses for $G_3(p;u_1)$ have been plotted for values of u_1 in the range $[0.001, 0.01]$ with steps of 0.001. It is seen that the lower u_1 is the

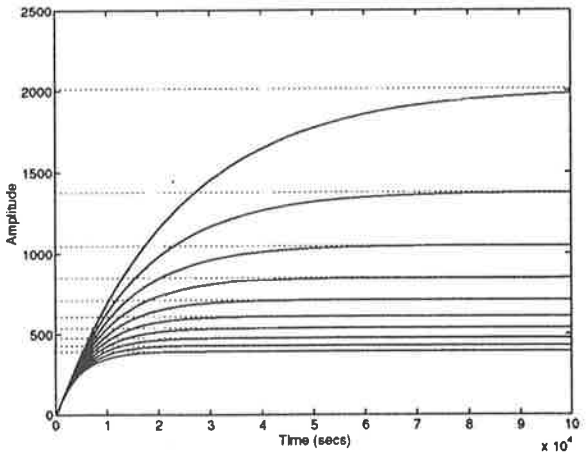


Figure 6. Plots of step responses for $G_3(p;u_1)$ for values of u_1 in the range $[0.001, 0.01]$ with steps of 0.001. The higher stationary values of the step responses corresponds to the lower values of u_1.

higher is the stationary gain. Further it is seen that the slope of the step response for small values of the time is independent of u_1. Thus it seems to be that it is only the low-frequency characteristics that are dependent on u_1.

The slope $S_h(u_1)$ of the step response for small values of the time is approximately given by

$$S_h(u_1) = \frac{K(u_1)}{T(u_1)} \tag{9}$$

where $K(u_1) = G_3(0;u_1)$, $T = \tau_1 + \tau_2$, and where τ_1 and τ_2 are the time constants of the heater. It can be shown that the slope is approximately constant and equal to 0.082 K/s. So for the heater to be faster than the averager, the slope

$$S_r = K_r/T_r \tag{10}$$

of the averagers step response has to be smaller than S_h. This implies the following inequality for choosing T_r for a given K_r

$$T_r > \frac{K_r}{K}T = \frac{K_r}{0.082} = 12.2K_r \tag{11}$$

As was pointed out in the previous section K_r may vary in the interval $[44, 110]$ with the user adjustment. Thus it is obvious that T_r also should vary as the user adjusts the characteristic curve. How much larger T_r should be made as compared to the right hand side in the inequality above will be investigated in the next subsection. This is related to the desired period and amplitude of the oscillation of the heater temperature. The actual value of T_r will be obtained by chosing a certain κ-value in the formula below:

$$T_r = \kappa\frac{K_r}{K}T = \kappa \cdot 12.2K_r \tag{12}$$

where κ is some constant strictly larger than 1.

3.3 *Harmonic Balance*

To further investigate the behavior of the controlled heater when the reference value is given as the output from the characteristic curve, the method of harmonic balance will be used. The idea of this method is to assume that there is an oscillation in the closed loop in Figure 5, and that

$$e(t) = E_0 + E\sin\omega t \tag{13}$$

Then it is assumed that all other signals in the loop are well described by their bias and first harmonics due to the low-pass characteristics of G, and that $\omega = W_0$ is constant. The equations of the harmonic balance will then after tedious manipulations imply the following inequalities

$$\begin{cases} |W_0 - E_0| \le |K - K_r| \\ E \le \dfrac{2}{\pi}|G(i\omega)| \end{cases} \tag{14}$$

where the ultimate frequency ω is obtained as the solution of

$$\arg G(i\omega) = -\pi \tag{15}$$

From the first inequality it follows that it is easier to make E_0 small the larger $|K - K_r|$ is made. Equation 15 implies that the amplitude of the error signal may be chosen by a proper selection of G. Remember that G is dependent on K_r as well as on T_r. This dependence will now be investigated.

If $K \neq K_r$, it can be shown that

$$G(p) = \frac{N(p)}{D(p)} \tag{16}$$

where

$$\begin{aligned} N(p) &= (K - K_r)\left[1 - p/\left(\sqrt{a^2 + b} + a\right)\right] \\ &\quad \cdot \left[1 + p/\left(\sqrt{a^2 + b} - a\right)\right] \\ D(p) &= (p\tau_1 + 1)(p\tau_2 + 1)(pT_r + 1) \\ a &= \frac{1}{2}\left[KT_r - K_r(\tau_1 + \tau_2)\right]/\left[K_r\tau_1\tau_2\right] \\ b &= \left[K - K_r\right]/\left[K_r\tau_1\tau_2\right] \end{aligned} \tag{17}$$

The behavior of the transfer function is qualitatively different for different signs of the parameters a and $K - K_r$. For positive values of a, which is equivalent to the inequality in (11), the nominator N will contribute with phase advancement, while for negative values of a it will contribute with phase retardation. Since the transfer function G has a decreasing magnitude for high frequencies, it is advantageous to have a positive in order to get a high value of the ultimate frequency given by (15), and thus small values of the amplitudes $|G(i\omega)|$ and E. Further the larger a is made, the larger will T_r be, and the smaller will $|G(i\omega)|$ be due to the denominator D being dependent on T_r. Thus it is also seen from this analysis that T_r should be chosen to fulfill the inequality in (11).

Further, since a positive value of $K - K_r$, which is obtained for low values of the flow u_1, will give lower value of the argument of G, and thus a lower value of the ultimate frequency and a higher value of $|G(i\omega)|$ and E, as compared to a negative value of $K - K_r$, it is the lower values of the flow u_1 that are the critical ones when chosing T_r. From various Bode-diagrams it can be seen that chosing $\kappa = 8$ will in this example imply an amplitude of G that is lower than 10 dB for the ultimate frequency and all values of the flow u_1. This value of κ corresponds to a time constant T_r of 2 hours. It can also be seen that the ultimate period of the oscillation is about 3 to 10 minutes. This is of course too small a period from a practical point of view. It will be seen later in the simulations that a hysteresis of ± 1 in the relay will increase the period to about 15 minutes while still keeping the amplitude E of the oscillation below the maximum allowed value of 5 K.

4. ANTI WINDUP SCHEME

The constraints mentioned in Section 2 will cause windup in the averager scheme for sufficiently large disturbances and set-point changes. Windup is here used in a more general sense than usual. Normally the notion of windup is used when the integrator state of a controller becomes large due to saturation of the control signal. Here the reference value for the flow temperature becomes large when the heating system cannot deliver the power needed, and this may thus be interpreted as windup of the reference value. Fuzzy anti windup for PID controllers has been described in Hansson *et al.* (1994). The development of the fuzzy scheme for the heater temperature reference value will be similar to the development of the anti windup schemes for PID controllers given there.

As was mentioned above the idea is to decrease the rate of change of the reference value for the flow temperature when the on-off signal to the heater is saturated, i.e. has been on or off for a longer period of time. To this end let τ be the time since the last change of the heater on-off signal and let K be the inverse of T_r, i.e. $K = 1/T_r$. Notice that this is not the same K as in the previous section. The rules for the fuzzy scheme inspired by Scheme 2) in Rundqwist (1991), p. 14 are

1) If TAUS then KN
2) If TAUL then KZ

where TAUS, KN, TAUL, and KZ means τ is small, K is nominal, τ is large, and K is zero respectively. Introduce the membership functions

$$\begin{cases} \mu_{\text{TAUS}}(\tau) = \exp(-\alpha\tau) \\ \mu_{\text{TAUL}}(\tau) = 1 - \exp(-\alpha\tau) \end{cases} \tag{18}$$

for TAUS and TAUL, where α is some positive constant. Further let

$$\begin{cases} \mu_{\text{KN}}(K) = \begin{cases} 1, & K = K^{\text{nom}} \\ 0, & K \neq K^{\text{nom}} \end{cases} \\ \mu_{\text{KZ}}(K) = \begin{cases} 1, & K = 0 \\ 0, & K \neq 0 \end{cases} \end{cases} \tag{19}$$

be the membership functions for KN and KZ, where K^{nom} is the nominal value of K, i.e. the value of K that results from the design of the averager when only considering the heater dynamics.

Then using the max-min-inference rule and taking as value of K the mean of the resulting membership function will give

$$\begin{aligned} K(\tau) &= \frac{\mu_{\text{TAUS}}(\tau) \cdot K^{\text{nom}} + \mu_{\text{TAUL}}(\tau) \cdot 0}{\mu_{\text{TAUS}}(\tau) + \mu_{\text{TAUL}}(\tau)} \\ &= K^{\text{nom}} \exp(-\alpha\tau) \end{aligned} \tag{20}$$

The scheme for T_r is thus given by

$$T_r(\tau) = T_r^{\text{nom}} \exp(\alpha\tau) \tag{21}$$

where T_r^{nom} is the nominal time constant.

Since T_r^{nom} is the time constant chosen in the design of the averager when the constraints are neglected, there is only one parameter, α, to be tuned for the fuzzy anti windup scheme. Further notice that the expression for $T_r(\tau)$ is given explicitly. Thus the implementation of the scheme will be computationally cheap. A discrete time implementation is given by

$$\begin{cases} T_r(k+1) = \exp(\alpha h)\,[1 + |u_3(k) - \eta(k)|]\,T_r(k) \\ \qquad\qquad + |u_3(k) - \eta(k)|\,T_r^{\text{nom}} \\ \eta(k+1) = u_3(k) \end{cases} \tag{22}$$

where h is the sample interval.

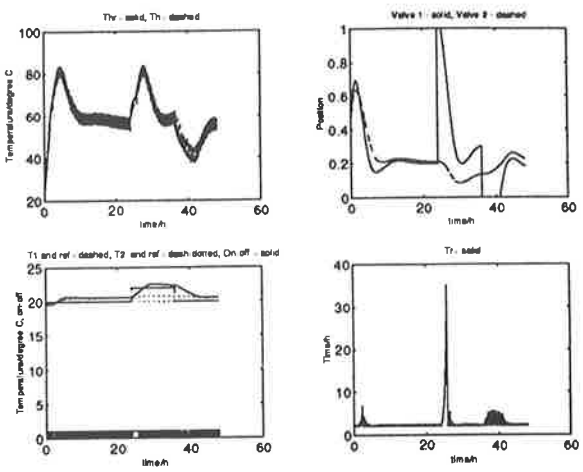


Figure 7. Simulation with a nominal value of the time constant $T_r^{\text{nom}} = 2$ h and a value $\alpha = \ln(1.5)/900$ of the fuzzy parameter.

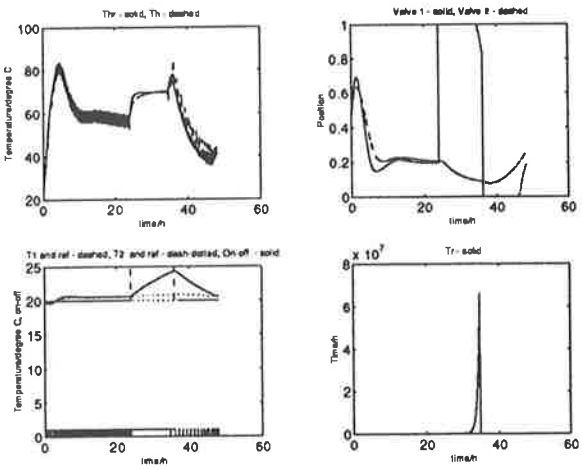


Figure 8. Simulation of the fuzzy scheme with step responses of hight 5 for the temperature in room 1.

5. SIMULATIONS

In this section the fuzzy anti windup scheme designed in Section 4 will be simulated. This will give a feeling for how to chose the parameter in the anti windup scheme. Further it will be seen

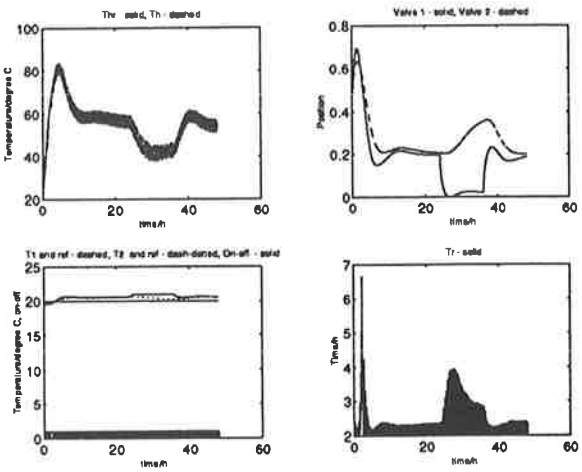


Figure 9. Simulation of the fuzzy scheme with load disturbance of size 100 W/m^2 in room 1.

that anti windup improves the performance. Parts of the model, the heater and the averaging scheme, have already been described in the previous sections. Here the hydraulics and the room dynamics will be described briefly. The house to be heated has two rooms. For each room there are 4 state variables modeling the temperatures of the air in the room, of the furniture in the room, and of the inside and outside of the wall respectively. The radiators in each room are modeled with 8 state variables each. The heater, the house, and the radiators are connected via the hydraulics. This is modeled with one state each for the flow and return temperatures, and with one state for each temperature sensor. Further there are P-controllers acting on the valve positions in the rooms to control the room temperatures T_1 and T_2. The power of the burner has been set to 18 kW. The temperature u_4 outside the heater has been kept constant equal to 15 degree Celcius. All initial temperatures have been set to 20 degree Celcius except for the wall temperatures which have been set to 10 and 15 degree Celcius respectively . The out-door temperature u_1 has been kept constant equal to 10 degree Celcius.

The simulation experiments performed have been done to design a good anti windup scheme and to compare this with averaging schemes without anti windup. The experiments performed contain set point changes for the room temperatures and load disturbances. The simulation time is 48 hours. First a constant values of the time constant in the averaging scheme have been used. The experiment performed has two set point changes for the temperature in room 1. After 24 hours the reference value is set to 22, and then after another 12 hours it is reset to 20. The results are seen in Figures 3. It is seen that, for a time constant of 2 hours, the step responses is fast. However, the behavior is almost unstable. In Figure 7 the same experiments as described above have been performed. However, now the fuzzy anti windup scheme is used to modify the time constant from a nominal value of 2 hours to a higher one. The parameter α in the fuzzy scheme has been chosen to be $\ln(1.5)/900$. Thus the time constant will have a 50 % higher value than the nominal one after 15 minutes. In the sequel the value $\alpha = \ln(1.5)/900$ will be used. In Figure 8 a step response with step size 5 is shown. Further in Figure 9 the behavior with respect to a load disturbances of size 100 is shown. It is seen that the fuzzy windup scheme manages well both with respect to set point changes and with respect to load disturbances.

6. CONCLUSIONS

In this report an observer for the reference value of the flow temperature of a heating installation has been designed. Possible speeds of the observer have been investigated. First a nominal fast design has been made utilizing the heater dynamics. Then an anti windup scheme has been designed by means of fuzzy logic to slow it down when large disturbances interfere with the nominal design.

The observer is composed of two parts: an averager with a time constant to be chosen, and a characteristic curve relating the stationary flow temperature demand to the stationary power demand. It has been seen that the time constant of the averager should be made dependent of the user adjustment of the characteristic curve. Further a lower bound on the time constant has been obtained. By means of the method of harmonic balance the amplitude and frequency of the control error signal has been related to the averager time constant. Thus the nominal design is easily done.

In an example it has been found that a reasonable value of the time constant is 2 hours. It must be stressed that the value of the time constant is dependent on the heater characteristics. It is easily seen that the higher the power and the smaller the water volume and mass of the heater is made, the smaller may the time constant of the averager be made. Further by introducing hysteresis the period of oscillation may be increased in order to find a practical value.

It has been seen in simulations that it is not possible to find a constant value of the time constant with sufficiently good performance. Thus anti windup has to be considered. A fuzzy anti windup scheme inspired by the so called conditional integration methods has been designed to prevent this windup. The scheme has only one tuning-parameter. This will depend on the dynamics of the heated rooms as well as the hydraulic characteristics. Further the scheme is given explicitly as a function of the time since the heater on-off signal last changed its value. Thus the implementation of the scheme will be computationally cheap, i.e. no special purpose signal processor will be needed, which is often the case with fuzzy control. The behavior of the anti windup scheme seems to be robust with respect to the tuning parameter. Thus it is easy to find a good value of this parameter. Further it has been seen that the anti windup scheme is robust with respect to different sizes of set point changes and load disturbances. Thus the overall performance of the anti windup scheme seems to be good. A similar scheme has with success been incorporated in the Sigmagyr RVP110 controller which has been introduced recently on the market.

7. REFERENCES

HANSSON, A., P. GRUBER, and J. TÖDTLI (1994): "Fuzzy anti-reset windup for pid controllers." *Control Engineering Practice.* To appear.

RUNDQWIST, L. (1991): "Anti-reset windup for pid controllers." Technical Report TFRT–1033, Department of Automatic Control, Lund Instiutute of Technology, Lund, Sweden. PhD Thesis.

ACTION PLANS DYNAMIC APPLICATION IN THE ALEXIP KNOWLEDGE-BASED SYSTEM

S. CAUVIN

Institut Français du Pétrole, 1-4 Avenue de Bois-Préau, BP 311, 92506 Rueil-Malmaison, France,
email : cauvin@c1.ifp.fr

Abstract : Alexip is a knowledge-based system for the supervision of refining and petrochemical processes. It analyzes the dynamic behavior of a unit and suggests corrective actions to maintain it or bring it to an optimum operating state. This paper focuses on the guidance part of this system. The method presented includes a description of operating situations allowing the exclusive application of knowledge concerning the overall state of the unit at a given time, a management of those situations, and a formalism for describing the inference mechanisms used for selecting the plan of actions to be applied at a given instant.

Keywords : Real-time Knowledge-based Systems, Knowledge representation, Process Control, Petroleum Industry

1. INTRODUCTION

The control room for an industrial site may be considered as the nerve center of the installation. Several operators are responsible for an increasing number of processes, for which they must ensure the "proper operating" or the "supervision." This job involves various tasks, *i.e.* analyzing the validity of the data received, hierarchizing the process alarms, analyzing the evolution of parameters to preventively detect problems, determining the overall operating mode of the unit and the material or strategic constraints, dynamically applying operating procedures for start-up, shutdown and resumption when problems arise. The Alexip (Cauvin *et al.*, 1992) knowledge-based system aims to be a real-time helper for the operator in these tasks of diagnosing, guiding and operating units. It is generically designed.

This paper concentrates on the guiding and operating aspects. A knowledge representation for dynamically selecting the actions to be undertaken is proposed. It can be actions of maintenance, adjustments of operating parameters, tuning of the control system, or action sequences included in known shutdown or start-up operating procedures. The method is based on a description of the elements to be taken into consideration as a function of the observed situation. The plan of action that is suggested at a given time takes into account the diagnosis phase conclusions. Therefore, it is always adapted to the new data acquired on the plant. We begin by giving a detailed description of the situation graph and then of the graphic and standard representation of the reasoning processes for determining plans of action. The method proposed has been implemented with Gensym's G2 software and has been tested on the IFP Alphabutol petrochemical process (Commereuc *et al.*, 1994; Couenne *et al.*, 1992).

2. SITUATION GRAPHS

The knowledge to be applied at a given time depends on the context. Only the part of the knowledge-based system corresponding to the general observed situation of the process is activated at a given time. Here, "activated" means available, and more precisely, possibly used by the inference engine. This corresponds to the focusing concept often discussed in the field of real-time knowledge-based systems. A situation in the program corresponds to a set of states of the process. It is defined by all the events acting on the process at any given time. We will start by characterizing these events and then will define the different situations and specify how to go from one to another. We will see that several types of situations exist (normal, disturbed, degraded and repairing), which are characterized by the way they are reached or abandoned. We do not propose any general mechanism for managing contexts (Mac Carthy,

1993), but we formalize a specific method for taking into consideration classes of events acting on the processes. In particular, we will not manage any overlapping of contexts.

2.1. Events

The set of the events that may occur in the unit contains all the *disturbances* and *operator actions*. Disturbances may be classified in two major types: *ordinary disturbances* corresponding to a change in unmastered variables and against which it is impossible to act directly (*e.g.* variations in the outside temperature) and *incidental disturbances* such as breakdowns or the improper operating of components of the system. It is posssible to remedy these disturbances by repairing a part of the device involved. Operator actions may be *adjustment actions* based on control parameters, and *actions on parts of the industrial device* or maintenance actions.

The diagnosing module is responsible for detecting these events. This is done either directly by a sensor or by analyzing the simultaneous evolution of a number of process variables. As long as an event has evident repercussions on the variables, it is considered to be present. For example, a repair (restarting of a pump, for example) is not considered to be present solely at the time of the change but as long as it affects the characteristic variables of the process. The diagnosing algorithm (Cauvin *et al.*, 1993a; Cauvin *et al.*, 1993b) determines the dominant events for which all the long- or medium-term characteristic consequences of the event are observed as well as the masked events for which some consequences are not observed because of the presence of other dominant events.

2.2. General Description of Graphs

Actually, the type of situation is linked to the type of event governing the evolution of the process at that time. All possible situations make up the *normal situations* corresponding to different control modes and *abnormal situations* corresponding to both the operating phases in a *disturbed* mode, *i.e.* in the presence of an incidental disturbance, and to operating phases in a *degraded or repair mode*, *i.e.* in the presence of the operator's actions enabling the process to operate either in a degraded mode in the presence of the disturbance or in the presence of repair actions returning the process to a normal state. To this set, let us add a so-called "Unknown" situation into which the system enters when no other known situation has been identified. This serves to determine the limits of the application.

This distinction in classes of situations is required since the way of going from one situation to another is different depending on the class. This is why, in section 2.4., we will not only give exact definitions of the situations with the help of the classes of events described above but also specify possible future situations and the input and output rule packets to be examined for each type of situation.

To avoid the problem of managing contradictions, we impose to have only one active situation at a given time. The transition rules that determine a situation are organized in packets which are activated at different instants depending on the last observed situation. Some packets can be examined only when a specific normal situation is present. They enable the situation to be abandoned by going via a consecutive degraded or repair situation. They are called "output packets." Others are not linked directly to a situation but detect a state of the process independently from the past situation and reach the conclusion of a normal or disturbed state. They are called "environment packets". The current situation determination is done as soon as a diagnosis is achieved. Therefore, the graph situation analysis is permanent.

2.3. Example

Let us look at the graph proposed in Figure 1 taken from the graph we worked out for the operating of an Alphabutol unit.

Situation S-N1 corresponds to normal operating during pressure and temperature control, and situation S-N2 corresponds to operating solely with temperature control. To begin with, let us assume that the unit is operating normally under pressure and temperature control. At this time the P-Normal and P-Disturbance rule packets are active. If the operator switches the pressure regulator to manual operating, the S-N1 situation is deactivated and the S-N2 situation activated by the P-Normal rule packet. If a pump problem occurs in the recycling circuit, the P-Disturbance rule packet places the system in the S-DI1 disturbed situation. Only the P1 rule packet then becomes active. It makes the system stay in the S-DI1 situation if the operator does not do anything, it places the system in the S-DE1 degraded situation if the operator undertakes actions which are not a repair action and places the system in S-R1 if the operator undertakes the repair action. In the situation S-DE1, the pump is not repaired. The P11 rule packet enables the system to

enter the S-R1 repair situation in which the pump is started up again. In this situation S-R1, the repair action causes various disturbances in the unit, which has to be stabilized. The P-Normal and P-Disturbance rule packets are then active. When the unit has been stabilized, it returns to a normal situation if no other problem is detected. If a new problem occurs, it immediately enters the associated situation.

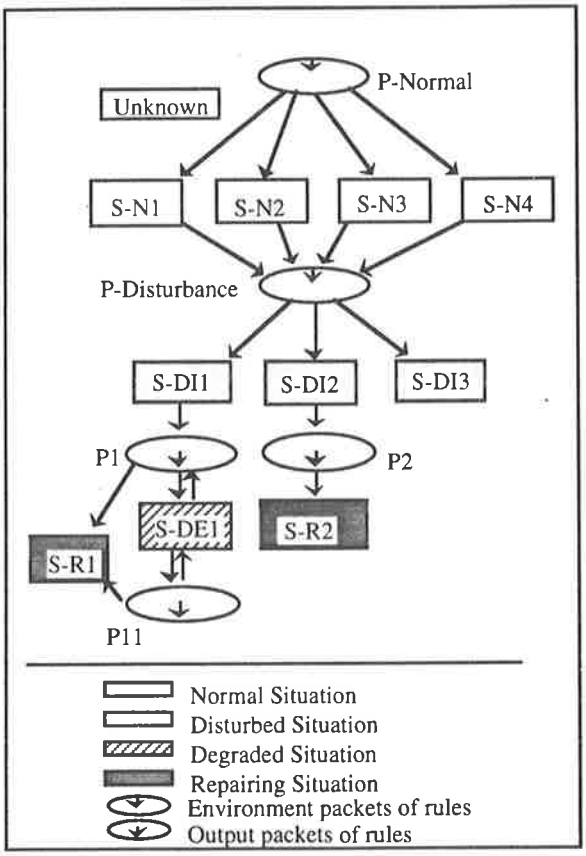


Fig. 1: Situation Graph

This graph must provide for the possibility that several disturbances may occur at the same time. In some cases, this requires describing a specific guidance situation in case there is a juxtaposition of events. However, in some cases, it is possible to consider that an event takes priority and to enter the associated state. At the level of the operating procedures, there is a system of constraints that can propose other actions if the ones we want to perform are impossible. This is often sufficient.

2.4. Formalization

Let us now give an exact definition of each type of situation and the form of each rule packet.

A situation is **normal** if the system is in a normal operating range and no event is detected or only ordinary disturbances are found or operator regulation actions are detected

Each normal situation is associated with a control mode for the process. A normal situation is activated at time t if the above conditions are checked and if the associated control mode is found.

Let's be more formal. We have the following sets :

A = {Operator actions}
D^i = {Incidental Disturbances}
S^n = {Normal situations}
S^{di} = {Disturbed situations}
S^{de} = {Degraded Situations}
S^r = {Repairing Situations}
V = {Variables describing the process behavior}
Ii : Normal interval for a variable vi
M = {Control modes}, T = {Instants}

We will note hereunder that s(t) is the current situation at time t.

As we define a normal situation for each control mode, we have:

$\forall m \in M \; \exists S\text{-}Nm \in S^n$ Associated-situation(m)=S-Nm

A normal situation is defined as followed

$\forall t \in T$ s(t)=S-Nm with $S\text{-}Nm \in S^n$ <=>

$\forall Vi \in V \; Vi \in Ii$ (a1)

AND $\forall d \in D^i \; d(t) \neq$ present (b1)

AND $\forall a \in A \; a(t) \neq$ present (c1)

AND m(t)=Observed$\wedge$
Associated-situation(m)=S-Nm (d1)

Set P^n of rules for entering a normal state is obtained directly by defining normal situations. This set is as follows:

$P^n = \{IF\ (a1) \wedge (b1) \wedge (c1)\ THEN\ s(t) = S\text{-}Nm\ / m \in M\}$

Since we are in a normal situation s(t), we must examine the rule packets for entering a normal situation or a disturbed situation. The set of rules P^d allowing to enter a disturbing situation will be defined later on. Nevertheless, we have:

$s(t) \in S^n$ ==> Activated Rules = $P^n \cup P^d$

AND $s(t+1) \in S^n \cup S^{di}$

A state is **disturbed** if an incidental disturbance is detected for the first time or if a disturbance has already been detected but no action has been

undertaken by the operator to counter it, *i.e.* none of the actions in the plan of action PA determined at the preceding time has been undertaken. This results in:

$\forall t \in T\ s(t) \in S^{di} <=>$

$\exists d \in D^i\ d(t)=present \wedge d(t-1) \neq present$ (a2)

OR $\exists d \in D^i\ d(t-1)=present \wedge$

$\forall a \in PA(t-1)\ a(t) \neq present$ (b2)

Now we can define the rule packet P^d for entering a disturbed state after being in a normal state. There may be more disturbed situations than incidental disturbances since a specific situation sometimes has to be created for a combination of disturbances. We note D^* is the set of disturbances and combinations of incidental disturbances giving rise to situations. We thus have:

P^d = {If $\exists d \in D^*\ d(t)$ = present Then s(t) = S-DId /$d \in D^*$}
with
$\forall d \in D^*\ \exists$S-DId$\in S^{di}$ Associated-situation(d)=S-DId

Since we are in a disturbed situation, at the following time t we must examine solely the rule packets $p^o(s)$ enabling us to get out of this situation s and to enter the following situation. Here we introduce a concept of order for the situations:

$s(t) \in S^{di}$ ==> Activated rules packet = $p^o(s)$
AND $(s(t+1)=s(t) \vee$
$s(t+1)=Next(s(t)),\ Next(s(t)) \in S^{de} \cup S^r)$

The output rule packet for a disturbed state will be specified only after a degraded situation and a repair situation have been defined, which are consecutive situations.

A situation is **degraded** if there is an incidental disturbance and operator actions to counter it.

$\forall t \in T\ s(t) \in S^{de} <=>$

$\exists d \in D^i\ d(t)=present$ (a3)

AND $\exists a \in A\ a(t)=present$ (b3)

The rule packets to be actived and the ensuing possible states are identical to those for disturbed states.

A situation is a **repair situation** when an incidental disturbance has disappeared and the system is still too disturbed for normal operating procedures to be applied. The active rule packets in this situation are all the environment rule packets:

$\forall t \in T\ s(t) \in S^r <=>$

$\exists d \in D^i\ \exists t' \in T\ t'<t\ d(t')=present \wedge$
$d(t) \neq present$ (a4)

AND $\exists vi \in V\ vi \notin Ii$ (b4)

The output rule packets for disturbed states have the form:

$p^o(s)$ = {If NOT(b3) $\wedge$ NOT(a4) Then s(t) = s(t-1),
If (b3)$\wedge$NOT(a4) then s(t)=Next(s(t)), Next(s(t))$\in S^{de}$,
If (a4) Then s(t) = Next(s(t)), Next(s(t))$\in S^r$)}

It is useless to rewrite the conditions for the existence of the incident since we are in a situation with an incident by definition of the disturbed and degraded situations.

The output rule packets for degraded states are quite similar and have the following form:

$p^o(s)$ = {If NOT(a4) Then s(t) = s(t-1) ,
If (a4) Then s(t) = Next(s(t)) with Next(s(t))$\in S^r$) }

A situation is **unknown** if no situation is detected.

2.5. Discussion

To conclude, the graph plotted is entirely standard. It is used for the systematic management of contexts. For another process, it "suffices" to determine the operating conditions and to write the above formalized rule sets. Each situation and each rule packet correspond to a G2 object which has a subworkspace containing the relative knowledge. Those subworkspaces are simply activated by an activate instruction. Therefore, we exactly have the same diagram as the one provided in figure 1 in the G2 application.

Note that situations may be considered as states in the sense of Petri networks except that, in each state, we will perform new reasoning before determining the action to be performed. Transitions are expressed in the rule, which have a great power of expression and deal with conclusions of the diagnosing phase since they take the events detected into consideration.

3. SELECTION OF PLANS OF ACTION

3.1. Representation of Plans of Action

For suggesting action sequences in a given situation, consideration must be given to the major conclusions of the diagnosis part and the overall evolution of the variables during the last hours.

We propose a graphic representation of the reasoning performed to achieve proper maintainability of the knowledge and an easy validation by experts. The goal is both the easy modification of the reasoning mechnisms leading to the creation of plans of action and the explanation of the suggestions put forward by the system at any given time. We will thus plot graphs representing the rules to be applied in time. The system will make a real-time analysis of the graph corresponding to the current situation. It will deduce the plans of action from this. This is a graphic representation of reasoning methods using standard tools dedicated to the creation of operating procedures. Other software has adopted this approach, GDA (Stanley *et al.*, 1991) in particular. Our approach has the advantage of being a good synthesis.

Figure 2 gives an example of this type of graph. The objects Pi are premises of rules that will be combined, and the objects Ci are conclusions, *i.e.* actions to be taken, for which the amplitude is calculated dynamically as a function of the values of the variables. To facilitate the creation of these graphs, classes of standard premises have been developed, including classes of standard conclusions and types of links between standard blocks. Several action plans can be created to take into consideration the assumption that a given plan of action may not be executed for any unknown reason by the computerized system.

In the example in Figure 2, by considering verified P1, P2 and P4 and unverified P3, the following two plans of action PA1 and PA2 have been generated.

PA1 = {C2,C3,C4} et PA2 = {C5}.

3.2. Classes of Premises

The classes of premises compiled are of three types: temporal, encapsulating and conclusion-diagnosis.

• A *temporal premise* is "verified" if the variable associated with it has a given sense of variation during a fixed period.

• An *encapsulating premise* contains rules concluding as to its validity.

• A *conclusion-diagnosis premise* is "verified" if the event associated with it has a given status following the diagnosis phase.

3.3. Classes of Actions

The conclusion blocks are used to define operator actions on parameters or PID controllers, maintenance actions and a specific action called "Wait and Do Nothing".

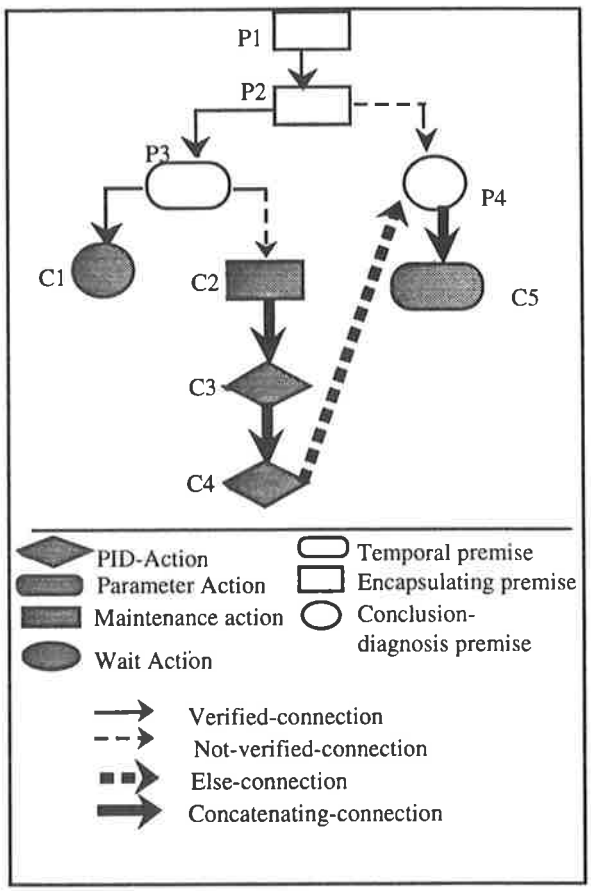


Fig. 2 : Creation of Plans of Action

3.4. Classes of Connections

The premise and conclusion objects must be connected together to formulate a reasoning. Four types of connections are defined as follows:

• A *verified-connection* linking a premise P1 to a premise P2 or a conclusion C1 indicates whether P1 has been verified whereas P2 must be tested or C1 must be concluded.

• A *not-verified-connection* linking a premise P1 to a premise P2 or a conclusion C1 indicates that if P1 has not been verified then P2 must be tested or C1 concluded.

• An *else-connection* linking a conclusion C1 to a premise P1 indicates that it is possible to create another plan of action by again starting from P1 for the reasoning.

• A *concatenation-connection* linking a conclusion C1 to a conclusion C2 indicates the concatenation of several actions in the same plan of action.

The system makes a real-time analysis of these graphs (the one corresponding to a given situation) at each time step in the reasoning. It deduces the plans of action that are lists of Conclusion blocks corresponding to the route of the graph. This last type of link can possibly be used to create several plans. At each connection of the concatenation-connection type encountered, the priority of the plan diminishes. (The first plan found is always considered to be the best. Others are created in case the user could not or would not want to undertake various actions for an unknown reason of the system.)

4. CONCLUSIONS

The method proposed for representing and selecting plans of action is based on an object representation. All the elements required are defined, and the user can graphically create his logical links. It would have been entirely possible to write graphs in the form of rules, but of specific rules in the propositional logic. If this were the case, a focusing mechanism on the rules in the sense G2, for example, consisting in invoking all the rules in a given category, could have been used to replace the activation of the situation. However, a mechanism would have had to be written for determining the rules to be invoked. The direct writing of a rule raises problem at the management level of the coherence of the reasoning and the real-time following of the execution. To help the user-developer, we have changed the color of the states, premises and conclusions used by the system at a given time. This enables him to follow and modify, in real time, his diagram while providing additional ergonomy.

We have tried to define the graph to be as much of a synthesis as possible. It can be seen that the blocks AND/OR are not on the graph proposed above. The nature of the connections suffices to carry out the reasoning.

The plans of action created take into consideration the evolutions of the variables in time and the conclusions of the diagnosis phase. The system is thus complementary to automatons and control systems already implemented in industrial units. The system works with and can be adapted to different processes.

5. ACKNOWLEDGMENTS

I am grateful to the Institut Français du Pétrole for having authorized the publication of this paper.

6. REFERENCES

Cauvin, S., B. Braunschweig, P. Galtier, and Y. Glaize (1992). Alexip, expert system coupled with a dynamic simulator for the supervision of the Alphabutol process, *Revue de l'Institut Français du Pétrole, Vol. 47,* No 3, pp 375-382.

Cauvin, S., B. Braunschweig, P. Galtier, and Y. Glaize (1993a). Model-based diagnosis for continuous process supervision: the Alexip experience. *Engineering Applications of Artificial Intelligence Journal, Vol. 6, No. 4,*

Cauvin, S. and B. Braunschweig (1993b). Graphical knowledge representation in the Alexip system for petrochemical process supervision, *Applications of Artificial Intelligence VIII*, Vol. 2

Commereuc, D., Y. Chauvin, J. Gaillard, J. Leonard, and J. Andrews. (1984). Dimerize ethylene to butene-1, *Hydrocarbon Processing.*

Couenne, N., and V. Catinaud. (1992) Commande multivariable d'un réacteur chimique (in French), *Pétrole et techniques,* No. 369, pp 31-38.

Dubois, D. and S. Gentil, (1991). Intelligence artificielle : un outil pour l'automatique ?, (in French) *Rapport groupe de travail automatique et intelligence artificielle, G.R. Automatique, pôle automatisation integrée,* pp 1-19, CNRS.

Georgeff, M.P., and F.F. Ingrand (1990). *Research on Procedural Reasoning Systems. Final Report, Phase 2, SRI Project 2851.*

MacCarthy, J. (1993). *Notes on formalizing contexts, IJCAI 93*, pp 555-560.

Stanley, G.M., F.E. Finch, S.P. Fraleigh, (1991) An Object Oriented Graphical Language and Environment for Real Time Fault Diagnosis, *European Symposium on Computer Applications in Chemical Engineering, COPE 91.*

COMPUTERISED SUPPORT IN THE PREPARATION, IMPLEMENTATION AND MAINTENANCE OF OPERATING PROCEDURES

J. TEIGEN* and E. NESS*

**Institutt for Energiteknikk, OECD Halden Reactor Project, P.O. Box 173, N-1751 Halden, Norway*
(Internet E-mail: John.Teigen@HRP.No, Eyvind.Ness@HRP.No)

Abstract. An overview of COPMA-II – an advanced, interactive, computer-based procedure handling system – is given followed by a description of how COPMA-II supports operating procedure preparation and implementation. Some experiences made in using a system like COPMA-II are discussed together with a presentation of ideas for further system development.

Key Words. Process control; man-machine systems; operator support; integrated plant control; software tools; computerised procedures; computer-aided instruction; on-line operation; monitoring

1. INTRODUCTION

A large number of observed and potential problems in the nuclear industry as well as in other industries are related to deficiencies in the current practices for management and implementation of operating procedures. Many of the problems identified in procedure preparation, implementation and maintenance can be directly addressed by developing computerised procedure handling tools. There is a growing interest in taking modern computer technology into use for improving today's practice in this field.

The OECD Halden Reactor Project (HRP) has since 1985 performed research work within the field of computerised operating procedures. The work has focused on methods and techniques for supporting both the procedure writing staff in the preparation of procedures, and the control room operators during procedure implementation. The research has been practically oriented in that software systems implementing and demonstrating the proposed methods and techniques have been developed and evaluated. A product of this effort is the development of the second version of the Computerised OPeration MAnuals (COPMA) system. This paper provides a description of the main features of the COPMA-II system followed by a discussion on the major strengths and weaknesses of the current system functionality.

1.1 COPMA-II Overview

COPMA-II has two main system components: The procedure editor, PED-II, is a tool designed to be used by the procedure writers during procedure preparation and procedure maintenance. Procedures to be used with COPMA-II must be expressed in a formal, general purpose procedure language, PROLA, developed by the HRP. The COPMA-II On-line procedure following system is the tool developed for supporting the process operators during retrieval and execution of procedures. The term *on-line* reflects that the system is designed to work with a live data communication link to the process computer, simulator, or any other external software component. Figure 1 illustrates the relationship between PED-II, COPMA-II On-line and the plant computer or simulator.

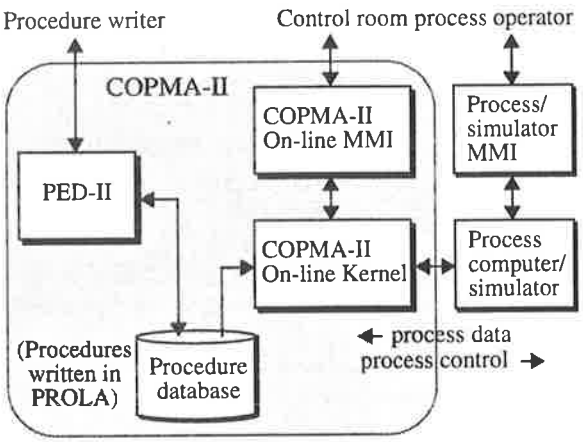


Fig. 1. COPMA-II system components.

1.2 The COPMA-II Approach

Before proceeding to the more detailed system description, some words about the overall approach taken in COPMA-II can be appropriate.

COPMA-II is a *tool* for preparing and implementing procedures. It is not a customized, plant-specific, turn-key system. There are no procedures delivered with COPMA-II. The system may be installed and used at any type of plant using operating procedures.

COPMA-II is intended to *replace* the traditional system of paper-based procedures. Existing hard-copy procedures must be transferred to COPMA-II by using the procedure editor. A more or less thorough rewriting of the procedure using the PROLA procedure language is necessary. There are *no* elements of automatic procedure generation or procedure synthesis during on-line operation.

COPMA-II acts as a *shell* for storing procedural information, for access and implementation by the operating crew. As designed, COPMA-II is not supposed to automate the actual execution of procedures. Normally, the operator drives the execution by acknowledging individual instructions within the procedure, making his personal judgements as much as he did when using hard-copy procedures. COPMA-II may also, if permitted to do so, act as a partial control interface to the process, because certain actions specified in the procedures can be carried out directly through the COPMA-II On-line user interface. The integrated information available in COPMA-II combined with the support functions offered by the system, is intended to improve operator performance when implementing operating procedures compared to when doing the same job with paper procedures.

2. PROCEDURE PREPARATION

COPMA-II requires that procedures to be used with the system must be formalised according to the syntax and semantics of the procedure language. Both the procedure editor and the COPMA-II On-line system are directly based on the definition of PROLA.

2.1 The Procedure Language

The PROLA language is general purpose. It is intended for use with any kind of procedure (e.g. event-oriented, symptom- or function-oriented procedures).

The main structure of a PROLA procedure is simple. Each procedure must, in addition to a title, have a short and unique identifier. The identifier should reflect a categorization of the procedures. A well designed set of procedure identifiers facilitates efficient search among the procedures during procedure retrieval. The body of the procedure contains a sequence of steps. Further, a *step* contains a sequence of instructions. In order to get a well structured procedure, the procedure writer should carefully collect related instructions into steps and give each step a well-chosen name reflecting the purpose of the step. The procedure language defines 12 *instruction* types:

ACTION. Used for specifying one or more process control actions. The operator is allowed to disable any action from being executed. Instruction example:

```
INSTRUCTION 2 ACTION
  START PumpX
  OPEN ValveY
```

AUTOCHECK. Used for specifying branching points in the procedure where the branching condition can be expressed as a logical and/or algebraic combination of process variables available through the on-line connection to the process computer. The current value of the branching condition is automatically evaluated by COPMA-II On-line at procedure implementation time. Example:

```
INSTRUCTION 3 AUTOCHECK
  IF NOT (ValveX IS OPENING)
  THEN GOTO Step 3-2-1 Instruction 2
```

FINISH. Terminate the current activity (the activity concept is explained below).

GOSUB. Causes the control flow to jump to a specified instruction in the procedure. The instruction specified should be the first in a sequence of instructions constituting a subroutine within the procedure. Intended to be matched by a Return instruction.

RETURN. Causes the flow of control to return to the first instruction following the last executed Gosub.

GOTO. Goto causes the flow of control to jump to a specified instruction with *no* implicit link to a following Return instruction.

INITIATE. COPMA-II On-line can handle several parallel executions of the same procedure. The *activity* concept has been introduced to distinguish between such parallel executions. The Initiate instruction is used for specifying that COPMA-II On-line shall automatically load a specific named procedure from the procedure database, and initiate a new activity associated with this procedure.

MANCHECK. Used for requesting the operator to do some manually performed check and respond to COPMA-II On-line by choosing among a predefined set (as specified by the procedure writer) of outcome alternatives. There is one branching instruction associated with each outcome alternative.

MANUAL-ACTION. Used for specifying manual actions to be performed by the operator.

MESSAGE. Used for presenting text messages (e.g. cautions, warnings or notes) to the operator.

MONITOR. A process condition similar to the type specified in an Autocheck can be continuously monitored by COPMA-II On-line during a specified time interval. Once the monitoring has been initiated, the operator can proceed with the execution of the next instruction in the procedure. The monitoring takes place "in the background", and the operator is notified if COPMA-II On-line detects that the process condition monitored evaluates to true. Example:

```
INSTRUCTION 4 MONITOR
  IF      LevelInSteamGenerator2 >= 3.28
    OR LevelInSteamGenerator3 >= 3.28
  INSIDE-INTERVAL
    FROM ValveX IS OPENING
    UNTIL 30 MIN AFTER ValveY IS CLOSED
  THEN INITIATE PROCEDURE D-YB-001
```

WAIT. Used for preventing the operator from proceeding to the next instruction in the activity until some specified amount of time has elapsed and/or

until some specific process condition is fulfilled. The operator is allowed to abort/skip any Wait instruction if he finds it necessary. Example[1]:

```
INSTRUCTION 5 WAIT
  FOR ValveX IS OPEN AND FlowY >= 25.5
```

2.2 The Procedure Editor

When preparing procedures by use of PED-II, the major part of a procedure is entered by just typing in text into predefined instruction formats. In principle, a procedure written in PROLA can be prepared using a plain text editor. PED-II, however, provides the procedure writer with some additional support:

Syntax check. By use of an incremental syntax check strategy, each step and instruction in the procedure is checked as it is entered or modified.

Control flow consistency. When attempting to save an edited procedure, the procedure writer is warned about any loose-ended control flow transitions (Gosub or Goto arguments without a matching instruction in the procedure). A saved procedure still containing loose-ended control flow will be flagged as incomplete and rejected by COPMA-II On-line.

Step and instruction numbering. A semi-automatic numbering strategy ensures that all steps and instructions within a procedure are numbered using a continuous and monotonously increasing sequence based on (composite) natural numbers. Example:

```
STEP 2
  INSTRUCTION 1
  INSTRUCTION 2
STEP 3-1
STEP 3-2
```

In this way it is easy to identify the structure of the procedure, the location of individual steps and instructions and to follow control flow transitions.

Process condition graphics. Process conditions, such as those specified in Autocheck and Monitor instructions, may include complex combinations of logical operators (AND, OR, NOT) making them hard to read. When improperly stated or formatted they may even be ambiguous. By use of an integrated graphical editor, PED-II supports graphical representation of the logical part of Autocheck and Monitor instructions. By representing the logical terms graphically by use of logical ports (AND, OR and NOT) even complex logical expressions should become easy to comprehend. In the COPMA-II On-line system at procedure execution time, colors are used in the graphical representation to indicate dynamically evaluated truth values. See e.g. Fig. 2, upper right for an example. The Autocheck includes the condition: "IF NOT (RL.. IS OPENING OR RL.. IS OPEN)".

Automatic procedure flowchart generation. The main output from the editor is a human readable text representation of the procedure specified in PROLA, ready to be interpreted by the COPMA-II On-line system. In addition to generating the PROLA text, PED-II also generates a file containing a description of a flowchart-like representation of the procedure. This description is used by COPMA-II On-line as a basis for presenting a simplified flowchart diagram of the procedure. See the middle window labelled "Flowchart Page" in Fig. 2. Automatic flowchart generation ensures consistency between the textual version of the procedure and the flowchart representation.

1. The exact semantics of "OPEN", "OPENING", etc. used in these examples, can be configured at the end-user site.

3. PROCEDURE IMPLEMENTATION

With *procedure implementation* we understand the process of actually stepping through the procedure doing the work prescribed in the procedure text.

3.1 The COPMA-II On-line System

The integrated information available in the procedure following system, combined with the set of support functions offered to the operator, is intended to make the job of stepping through a procedure easier, faster, and more accurate than when performing the same job with paper-based procedures. Figure 2 shows a screen dump with a typical layout of the COPMA-II On-line MMI. The main system features, as seen from plant operations staff's point of view, include:

Procedure retrieval. Using indexing schemes like procedure categorisation together with appropriate search tools, correct procedures are found with a minimum of efforts. Automatic procedure retrieval can be used if there is a well defined alarm condition associated with the procedure.

Simplified procedure execution. The system maintains the thread of control in all executing procedures (activities). The operator's role is to supervise execution of individual instructions within the procedures. The operator may choose to Execute the current instruction, Skip it, or go back to the Previous one.

Parallel execution of procedures. COPMA-II On-line can manage the execution of several procedures (activities) in parallel. The system keeps track of the location of the current instruction within each activity. The operator can easily switch his focus between the different activities he is currently working with.

Visual procedure overview. A generated, simplified flowchart representation of the procedure is used for providing the operator with an easily perceivable overview of the procedure, and keeping the operator updated on the current position in the execution of the procedure. The flowchart representation is dynamically updated using colors to indicate which instructions have been executed, which is the current one, and which instructions that have not yet been visited.

Process control. The live data communication link to the process computer enables COPMA-II On-line, if explicitly permitted to do so, to send control signals to the process computer (according to the description of the Action instruction).

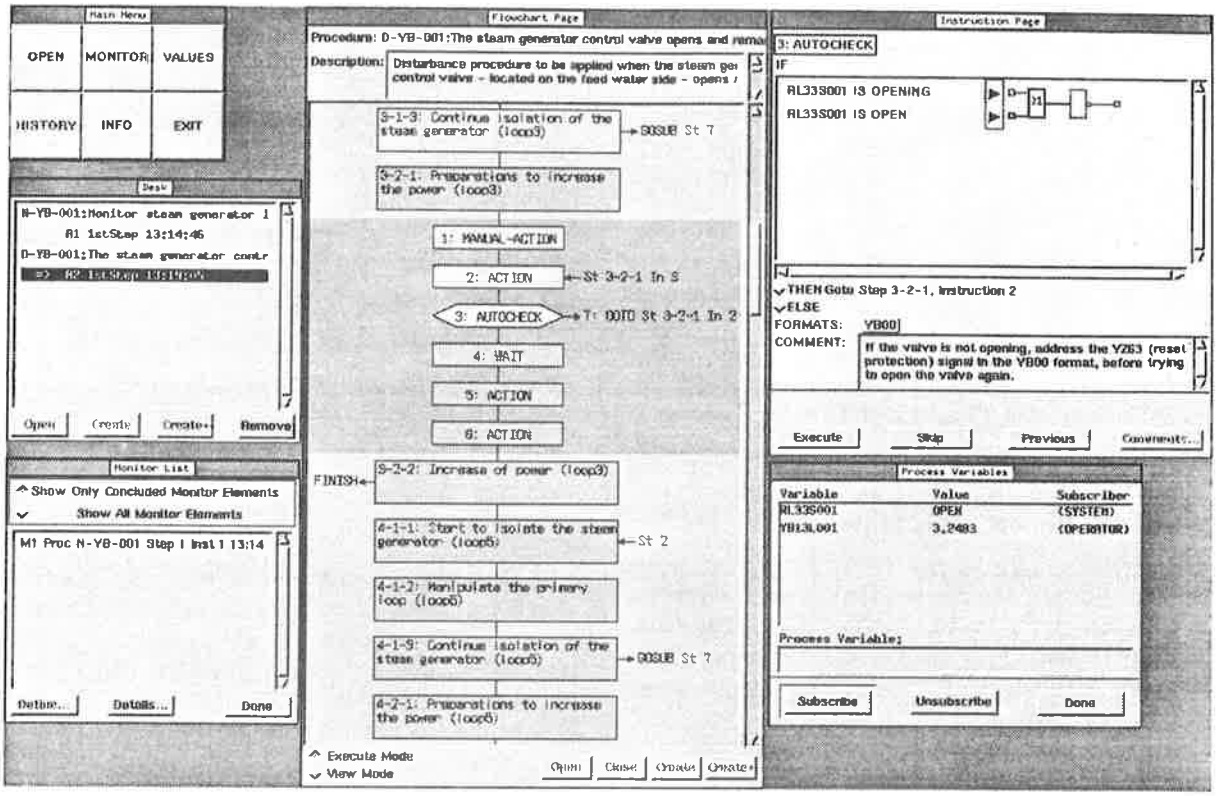


Fig. 2. COPMA-II On-line MMI.

Process measurements. The current values of process measurements referred in an instruction are automatically presented to the operator when the instruction becomes current at procedure execution time. The operator also has the option to subscribe (on a more permanent basis) the value of any process variable available to COPMA-II On-line through the data communication link.

Automatic data collection. Dynamic process measurements received from the process computer can be used in automatic evaluation of expressions, e.g. branching-decisions, contained in the procedure (used in Autocheck, Monitor and Wait instructions).

Automatic process monitoring. The system can be employed to do process state monitoring with subsequent operator notification and automatic retrieval of the correct procedure when the condition monitored becomes true, or if the monitoring interval expires.

Procedure execution log. COPMA-II On-line can be configured to keep history records of procedure executions. The history log will keep track of *which* instructions have been executed, the execution *time*, and the status of all process parameters referred in each particular instruction at the time it was executed.

4. DISCUSSION

In this section some of the experienced strengths and weaknesses of COPMA-II are discussed. Throughout the discussion we refer to some common problems and lessons learned with paper-procedure systems in the nuclear industry as reported by Lapinsky (1989).

4.1 COPMA-II Strengths

Procedure structure and format. A consistent and clearly defined structure and format can contribute significantly to increase procedure comprehensibility, minimize operator confusion and errors, and aid the operator in quickly finding the necessary information. When operators are trained to have a common understanding of and familiarity with what the procedures look like and how they work, variations in operator performance will be minimized. Lapinsky (1989) states that:

- There should be only one method for presenting each procedure component.
- Action statements should not be embedded in notes and cautions.
- There should be no confusion about procedure entry and exit points.

The procedure language in COPMA-II is structured and consistent. If used as intended, the procedure language is clarifying. There is a limited set of instruction types. Each instruction type has a well defined syntax and semantics. Further, the procedure language enforces the writing of explicit and precise procedures. We have e.g. uncovered several problems of ambiguity with existing paper-based procedures when converting them for use with COPMA-II.

Flow of control transitions. When implementing procedures operators often have to perform transitions either to other parts of the same procedure, or to another procedure. An operator may also be directed to work with several procedures at a time. Lapinsky

(1989) refers to some commonly observed problems:

- Movement within and between procedures can be disruptive and confusing and can cause unnecessary delays and errors. Excessive transitions or incorrect transitions that increase the possibility of operator error were found at many plants.

He also proposes some good rules to stick to:

- The least number of transition terms possible should be used to indicate different types of transitions. These should be used consistently to minimize confusion and ensure operator recognition of transition structure.
- When transitions cannot be avoided, it is important that the transition direction to the operator be clearly and consistently structured.

In COPMA-II control flow transitions are specified by use of the PROLA instructions Goto, Gosub + Return, and Initiate + Finish. The syntax and semantics of these instruction types are well defined. When executing a Goto, Gosub or Return instruction, COPMA-II On-line automatically keeps track of which instruction is the next to be made current. The step and instruction numbering strategy applied in COPMA-II ensures that there should never be any confusion about control flow transition directions.

COPMA-II applies a clear and consistent way of referencing other procedures (by unique identifiers) in connection with control flow transitions between procedures. By use of the Initiate instruction, COPMA-II also supports automatic retrieval of the new procedure to be executed.

Placekeeping mechanism. During execution of a procedure the operator may be required to reference tables, charts, supplemental information, or other procedures. Both when referencing information that is not included in the procedure, when performing transitions within a procedure, or when he must keep track of the execution of several concurrent procedures, he has a problem associated with placekeeping. This is reported by Lapinsky (1989) as a common problem area in the industry.

COPMA-II supports the execution of several procedures in parallel. The system keeps track of where the operator is in the execution of each procedure. By use of the procedure flowchart, the operator can easily get an overview of which parts of the procedure have been executed and the location of the current instruction when switching between procedures.

Logic statements. Decisions play an important role in the execution of an operating procedure. It is important that decision and logic steps are clearly, consistently, and appropriately structured. However, Lapinsky (1989) states that ambiguously worded logic statements appear to be widespread.

Autocheck, Mancheck and Monitor are the instruction types used for implementing decisions in COPMA-II. When specifying process conditions for automatic evaluation in Autocheck and Monitor instructions, as directed by the definition of the procedure language, they will not be ambiguous.

Process monitoring. Lapinsky (1989) reports some problems regarding process parameter monitoring:

- At some plants the level of detail and the scope of the procedures required that all operating resources be dedicated to monitoring critical safety function parameters and control board manipulations.

If the process parameters to be monitored are available in the process computer, COPMA-II supports automatic process state monitoring and operator notification through the Monitor instruction mechanism.

Procedure flowchart. Procedure flowcharts provides the operator with a visual overview of the procedure. However, Lapinsky (1989) reports that:

- Flowchart format is extremely difficult to develop and implement properly. As a result, flowcharts are being developed that are difficult to read, understand, and physically use, with a high potential for confusion and operator error. Many flowcharts currently in use are so difficult to use that they impede rather than support operator performance.
- Control rooms often lacked sufficient table top or desk space to allow operators to physically spread out the procedures. This was especially a problem in those plants using large flowcharts or where several procedures had to be used concurrently.

In COPMA-II we have applied a simplified (one-dimensional) flowchart representation of the procedure with a consistent format that is automatically generated based on the procedure text. All control flow information is present in the flowchart. Colors are used in the flowchart to show which instructions have been executed as well as the location of the current instruction. The flowchart representation is mouse sensitive, so the operator may address and open individual steps and instructions to see their contents in full detail. The actual details of the instructions are not included in the flowchart format, but displayed in a separate instruction window (see Fig. 2). Only the flowchart of the procedure currently executed is displayed. When switching to another concurrent procedure (activity), the flowchart of this procedure will appear.

Implementation time and failure probabilities. COPMA-II encourages more explicit procedures. This should result in improved procedure quality. By use of Action instructions the possibilities for operator control errors should be reduced because COPMA-II explicitly provides operations to be directly performed on the correct process component through the procedure system interface. By use of Autocheck, Monitor and Wait instructions, the COPMA-II system provides automatic process parameter sampling to reduce the operator's workload and the potential for making errors when checking and monitoring plant parameters. These functions should also make it easier to perform time-critical operations more quickly, both during normal operation and when han-

dling disturbances.

Both COPMA-I and COPMA-II have been subjected to human factors evaluation experiments with process operators as test subjects (Nelson *et al.*, 1990; Converse, 1993). Briefly, these studies have shown that operators can increase their performance and reduce their error rates when using COPMA, compared to using paper-based procedures.

4.2 COPMA-II Weaknesses

There are problems within the field of procedure systems that the present version of COPMA does not handle well. Some of these problem areas, which are topics for further work, are discussed below.

Transfer of procedures to COPMA-II. In order to switch from a hard-copy based procedure system to COPMA-II, there will be a need for full reimplementation of all procedures by use of the procedure editor. The amount of efforts required depends on how much the old structure and format of the hard-copy procedure deviates from the structure and format required by COPMA-II. It is not easy to think of options for automatic transfer of existing paper-based procedures into COPMA-II. This is due to the fact that COPMA-II explicitly represents the *semantics* of the procedure content. Some sort of manual intervention will be required for converting the paper-based procedure into a computer-based one. The best thing to hope for is perhaps a semi-automatic approach supporting the transfer.

The procedure language. The procedure language – once developed – may put unwanted restrictions on the structure, contents and formatting of a procedure. It is a challenging task to develop a general purpose procedure formalism that suits the needs of rather different plants and procedures. One of the reasons for implementing COPMA-II is to have a running system usable for evaluation of the methods and techniques used. The current version of PROLA contains some weaknesses, and the language will be further developed as we get more experience with implementing different types of procedures by use of COPMA-II.

Hard-copy fallback procedures. Before installing a system like COPMA-II at a power station, it will be necessary to have a high quality paper-based backup system in case the computerised system for some reason should become unavailable. Paper-based procedures may also complement computerised procedures. Hard-copy procedures may be necessary for operators or technicians to perform local actions in some distance to the computerised procedure system interface. Procedures prepared for COPMA-II are available on a human readable text format. A simple formatting is provided, but the layout is not satisfactory for control room use.

Improved procedure preparation system. Not all industries are (at least at the present time) prepared to take a procedure following system relying on advanced computer technology into the control room. That is, however, probably not the situation for a well-designed off-line procedure preparation system. PED-II has not been designed to be an independent stand-alone product. Here are some ideas for a revised design of the procedure editor:

- The procedure preparation system should have features like those found in computer-based document preparation systems. It should be possible to prepare high-quality paper-based procedures with user-configurable format and layout.
- Tools for different kind of document language style checks (e.g. checks on acceptable acronym and action verbs, plant equipment nomenclature etc.) could be included.

Procedure graphics. The current version of COPMA does not support graphics (figures, diagrams, trend--curves etc.) prepared as a part of the procedure. Such extensions would be relatively straight-forward to implement, though, either directly in COPMA-II or via an interface to a dedicated, external graphics system like PICASSO (Barmsnes *et al.*, 1994).

Procedure revision control. The procedure preparation system will be used both for writing new procedures and for revising existing ones. PED-II does not provide satisfactory support for the procedure revision and maintenance phase. A future version should support a proper procedure revision control strategy.

5. CONCLUSIONS

The life-cycle process of preparing, implementing and maintaining operating procedures within the industry will benefit from using appropriate computerised tools. In COPMA-II we have addressed many of the experienced problems related to the handling of operating procedures. COPMA-II is, of course, not the ultimate solution and the only factor in ensuring high quality operating procedures, but provides a collection of carefully designed features that have demonstrated their usefulness in an operating context.

REFERENCES

Barmsnes, K. A., Ø. Jakobsen, T. Johnsen, H. O. Randem (1994). Developing Graphics Applications in an Interactive Environment. Proceedings of the 1994 SCS Simulation Multiconference, April 10-14, 1994, San Diego, CA. The Society for Computer Simulation.

Converse, S. A. (1993). Evaluation of the Computerized Procedures Manual (COPMA-II). Paper presented at the OECD Halden Reactor Project EHPG Meeting, Storefjell, Norway, March 7-12, 1993.

Lapinsky, G. (1989). Lessons Learned From the Special Inspection Program for Emergency Operating Procedures (Conducted March-October 1988). NUREG-1358, U. S. Nuclear Regulatory Commission, Washington.

Nelson, W. R., N. T. Førdestrømmen, C. B. O. Holmström, M. Krogsæter, T. Kårstad and O. Tunold (1990). Experimental Evaluation of the Computerised Procedure System COPMA. HWR-277, OECD Halden Reactor Project, Halden, Norway.

COAST - COMPUTERISED ALARM SYSTEM TOOLBOX

A.BYE, S. NILSEN, F. HANDELSBY and T. WINSNES

Institutt for energiteknikk, OECD Halden Reactor Project, P.O. Box 173, N-1751 Halden, Norway

Abstract. The OECD Halden Reactor Project has for several years been working with computer-based systems for determination of plant status including alarm filtering, early fault detection, and function-oriented plant surveillance. The methods explored complement each other in different plant operating regimes and provide diversity in plant monitoring systems. A new toolbox, COAST, has been made to enable integration of these different methods into an alarm system. Coast will be easy to couple to different processes. It will be an integral part of the final alarm system and not only act as a tool for building dedicated systems.

Key words. Alarm systems; failure detection; guidance systems; man-machine systems; on-line operation; operator support; software tools

1. INTRODUCTION

One of the main tasks for operators in nuclear power plants is to identify the status of the process when unexpected or unplanned situations occur. The alarm system is the main information source to detect disturbances in the process, and alarm handling has received much attention after the Three Mile Island accident in 1979 (Kemeny, 1979). It was realized that conventional alarm systems created cognitive overload for the operators during heavy transients.

The Halden Project developed an alarm filtering system called HALO (Handling Alarms using LOgic) using logic filtering to reduce the number of active alarms during process transients (Øwre and Marshall, 1986). HALO has been subject to a number of validation experiments with different presentation techniques, as described by Marshall *et al.* (1987).

Another approach is taken in the EFD (Early Fault Detection) system. The method used is to run small, decoupled models which calculate the state of the process assuming no faults, in parallel with the process. The behaviour of these models is then compared with the behaviour of the real process, and if there is a deviation, an alarm is issued. In this way false alarms are avoided, and one will only get one alarm for one fault. Prototypes developed for simulators and installations in real power plants e.g. the Imatran Voima owned plant in Loviisa, Finland, have demonstrated the feasibility of this methodology, provided that enough measurements are available for the process area considered (Sørenssen, 1990).

In case of major disturbances in a plant with a large number of alarms issued, a function-oriented approach is in many cases recommended. Instead of looking at single systems or variables and alarms within a system, one monitors critical safety functions in terms of whether these functions are challenged. The Halden Project has explored this concept through several systems, like CFMS (Critical Function Monitoring System) and SPMS (Success Path Monitoring System), and the post trip guidance system SAS-II (Øwre *et al.*, 1991).

Effectively handling and integrating different types of alarms into one system would improve the operator's overview and thereby the overall safety of any industrial plant. This was a major motivation for initiating the development of the COmputerised Alarm System Toolbox, COAST. The experience gained from the work with various alarm systems was utilised when designing the basic functionality of Coast. It should thus be possible to utilize Coast to make most kinds of alarm systems. The goal is to reach solutions that can minimize the cognitive overload of the operating crew, but still fulfil the basic philosophy of alerting, informing and guiding the operator, as well as confirming whether his/her actions have the desired result.

An alarm system toolbox which can be used on different processes, applying different alarm handling methods, has to be generic. The toolbox must be easy to use to tailor-make alarm systems for many different processes, e.g. nuclear power plants or oil production platforms, and it must possess a flexible

development environment, to improve and simplify the task of the alarm system designer.

2. FUNCTIONALITY OF COAST

2.1 Integration of Alarms

One aim is to combine the functionalities of the different alarm systems explored by the Halden Project into one integrated alarm system. This can be done in several ways, where the most generic way is to make a general software-package which is capable of making all types of alarms. Equally important is to facilitate the handling of these different alarms by making relations among them. The objective with Coast as such is to make the integration of different types of alarms possible. Coast will include the different functions needed, and it will be generic, such that it will be possible to use it to implement alarm systems for many different processes and plants.

In an integrated alarm system, the alarm processing may be divided into three stages: alarm generation, alarm structuring and alarm presentation. Alarm generation is the phase where all new alarms are generated from process measurements. All types of alarms should be generated by this "module", conventional alarms, function-oriented alarms, as well as model-based alarms. The advantage of having different types of alarms available when the operator investigates the status of the plant, is the diversity in the underlying methods, which contributes to a broader view and possibly a more robust conclusion. Model based alarms may be more sensitive than conventional alarms, and useful in dynamic situations, while function-oriented alarms are most useful in severe transients.

However, if only more information is presented to the operator, based on more information sources, the danger of cognitive overload may increase. In the alarm structuring phase one tries to cope with this problem. Structuring includes what normally is known as filtering of conventional alarms, and will thereby reduce the amount of information automatically presented to the operator, in this way clarifying the disturbance situation.

It will be possible to couple Coast to an existing alarm system. Existing alarms will then be structured or filtered by Coast before presentation.

2.2 Flexibility

The results of advanced alarm structuring may support the operator with different types of alarm lists, which he can use interactively in his status identification task. Further, non-observable events such as internal leakages may be inferred by the alarm system. Let us illustrate this by considering the simple example given in Fig. 1.

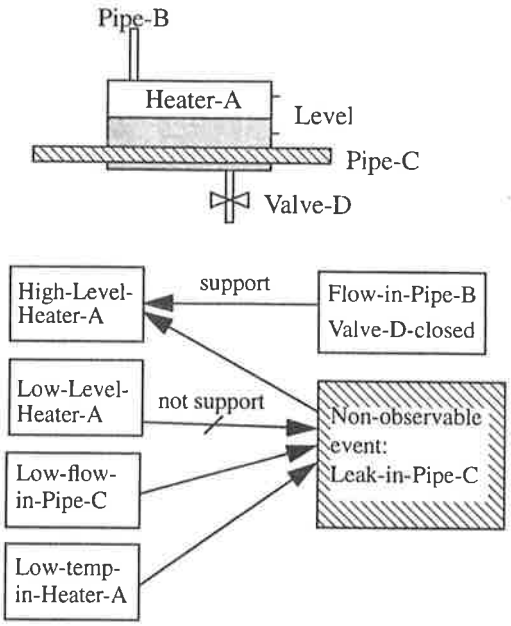


Fig. 1 Event / alarm network for hypothesis related alarms. In this network the shaded box represents a non-observable event, and the others represent alarms or observable status events.

The arrows without the crossing line are relations which indicate explanatory alarms or events, for example *flow-in-pipe-B* combined with *valve-D-closed* may explain the alarm *high-level-heater-A*. The arrow with the crossing line indicates the opposite, i.e. that a *low-level-heater-A* alarm probably not supports the event *leak-in-pipe-C*.

For the alarm *high-level-heater-A* in the example in Fig. 1, lists of explanatory alarms and events are specified:

high-level-heater-A:

Explanatory alarms and observable events:
flow-in-pipe-B, valve-D-closed

Explanatory non-observable event:
leak-in-pipe-C

Flow-in-pipe-B combined with *valve-D-closed* may then explain why the alarm *high-level-heater-A* has appeared. *Leak-in-pipe-C* refers to a non-observable condition. This hypothesis could be the *diagnosis* if no other alarms or observable events are present.

The main idea is that when the operator is searching for a solution to a problem he can make a hypothesis. This can either be an alarm which has not yet occurred, or a non-observable event. The a priori established relationships between the alarms can be used to confirm or disconfirm this hypothesis. The

hypothesis must be present either as an alarm not yet observed, or as a non-observable event. Alarm lists can be generated and presented for the operator when he proposes a hypothesis:

Hypothesis: *leak-in-pipe-C:*

Supporting alarms:
12:03:15 Low-flow-in-pipe-C

Opposing alarms:
11:45:20 Low-level-heater-A

Not yet observed alarms:
??:??:?? Low-temperature-in-heater-A

The presence of both confirming and disconfirming alarms cannot be excluded, because the relations are only normative. For instance if both *low-flow-in-pipe-C* and *low-level-heater-A* have occurred, the operator may ask himself whether really *leak-in-pipe-C* is true.

In this very simple example, it is straightforward to keep the overview of the relations. But in a realistic situation, the number of relationships and active alarms will be too many to handle for the operator. The system aims to offer the operator help in selecting those alarms which seem to confirm or disconfirm his hypothesis. In this situation the operator will use the alarm/event network interactively on-line, by interrogating the system with respect to some particular hypothesis. This network could even be subjected to a systematic search to find those hypotheses which are most supported by the active set of alarms.

Coast offers a number of facilities to handle the above example. Alarm objects and relations between the alarm objects are the basic building blocks in the toolbox. This facilitates a flexible structuring of alarms, which may be used both for conventional filtering, and for the more general structuring purposes as shown in the example above. It will be possible for the alarm system designer to make different dynamic relations between alarms. Together with very strong on-line selection capabilities, different lists can be made available to the operator, which he can use in his diagnostic reasoning task. This provides a best possible tool for the operator to search for and test different hypotheses.

2.3 Use of COAST

Configuration. The alarm system designer will use the toolbox to make a specific alarm system. It is possible to operate Coast without active external connections, so classes etc. can be defined before the toolbox is coupled to any external system. However, one may also run with all external systems connected. The designer has to configure the alarm system and decide which external systems that are going to be connected to the toolbox. An application programmer's interface (API) provides easy coupling of external systems to Coast. A library of API-functions must be included in the application, and all exchange of data is done through these functions.

Coast Language. The alarm system designer defines the specific alarm system by writing definitions according to the specific syntax provided by the COast LAnguage (COLA). This can be done in two ways: Writing the Cola definitions in text-files, which are fed into the kernel, or by using a graphic, syntax sensitive editor. Pure text-files will be useful in cases where parts of the definitions are already prepared and stored. It may then be read into the kernel with minor modifications to fit the Coast language.

O-O Design. Coast offers an object-oriented way to construct alarm systems for the alarm system designer. Object-oriented design includes classes with attributes, instances of the classes, methods and relations. The attributes may have dynamic behaviour, which may be given default on class level, or overridden by specific behaviour on instance level. Typical attributes of an alarm object may be the type and the priority of the alarm. Typical methods may be the updating of status etc. The alarm objects may be related to other alarm objects through relations. The relations are also defined by the alarm system designer, and may be considered as active or non-active pointers between the objects. A set of basic alarm classes can be collected in a library.

Allowed constructs in Cola comprise arithmetic, conditional and logic expressions, including "for every", and other first order logic. The actions of methods includes very powerful selections, and in addition time dependent constructs are available for filtering. The definitions of these dynamic elements thus describe the dynamic alarm system.

The first task of the alarm system designer is to define alarm classes with attributes and methods, and relations. A very simple example of a class definition and a relation definition with the Coast language is shown below:

```
Classdef StatHiAlarm
    Attr Priority : string
    Attr HiAlarmStatus : string
    Attr System : string
    Attr Time : int
    Attr Meas : real with history
    Attr HiAlarmLimit : real
Method whenever Meas > HiAlarmLimit
    assign "on" to HiAlarmStatus
    assign clock to Time
EndMethod
EndClassDef
RelationTypeDef x : StatHiAlarm member-of
    Group1 if x.HiAlarmStatus == "on"
```

The attributes, relations and methods are the dynamic elements of Coast. In the example above the attribute *HiAlarmStatus* gets a new value whenever the condition in the method is true. Another way to write this is:

```
Attr HiAlarmStatus:string = if (Meas > HiAlarmLimit)
                            then "on" else "off"
```

Coast is an event-driven program, and updates all necessary values whenever something happens, e.g. when process measurements are fed into the kernel. In the above example '=' means "is always the same as". Coast does not execute as a sequential program. The updating of necessary attributes etc. is done by a kind of forward chaining. Consistency of values is ensured automatically, such that no cycles are present in the definitions. Also no value is updated more than once, and this ensures the best possible efficiency, which is highly needed, as the Coast language itself offers flexible and computational heavy constructs. The evaluation in the kernel is however fully deterministic, which is required from a system which shall be used to make alarm systems.

The graphic MMI equivalent of the above example will be to activate a ClassDef window, click on *new..* in an attribute popup window, and write the names of the attributes.

After having defined alarm classes, the next step for the designer is to instantiate alarm objects from the classes. Initial values and expressions may be given for attributes which are not going to keep the default behaviour given on class level. Using text-definitions according to the allowed syntax, it may look like:

```
ObjectDef RL10L001 : StatHiAlarm
    Priority := "Severe"
    System := "RL10"
    HiAlarmLimit = Basic.HiAlarmLimit
EndObjectDef
```

In Fig. 2 the instantiation of an alarm class using the graphic MMI is illustrated. Popup windows with defined alarm classes and attributes are shown. The user may type in the initial values of the attributes. In addition the example shows how limit check objects which generate alarms, can be coupled together in a group alarm. The arrows indicate the relations, and this is the same functionality as described by the relation definition in the textual Cola language.

2.4 Alarm Presentation

The alarm display system is not a part of Coast, and it is looked upon as an external system. However, during a concept study one proposed the alarm presentation done in three types of displays: Overview display, the ordinary process displays with alarm

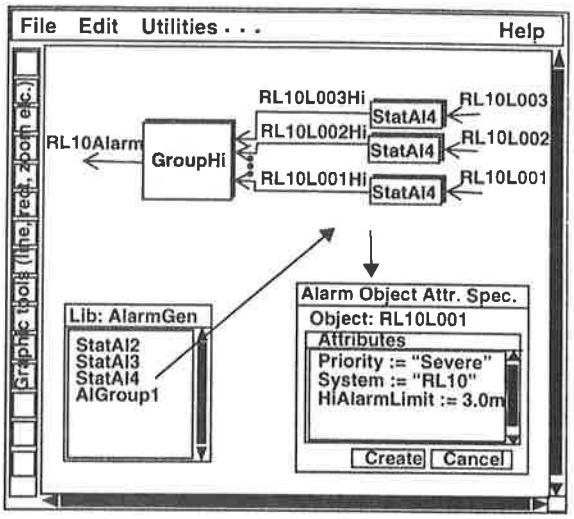


Fig. 2 Example of instantiation of an alarm class, and coupling between alarm objects, via relations. The alarm class StatAl4 is selected from a library, instantiated, and given the id RL10L001.

information, and selectable displays, which the operator can use interactively in his investigation task.

Coast offers a very strong functionality to enhance the operator's diagnostic task. The operator may want to look at dynamic alarm lists of various types interactively. The selective constructs comprise a subset of Cola (Cola light), which is made available for the end user. Cola light is interpreted, so it is possible to write in new selection criteria for new, dynamic alarm lists on-line. This feature will only offer the possibility to search among existing alarm objects, it will not produce new objects in the kernel, i.e., it is a reading, not a writing facility.

In practical use the operator will thus be in an X-windows environment, clicking on icons, or he may even write in new selection criteria on-line, to get up different alarm lists. This enables him/her to find out as much as possible about the current alarm situation and the state of the plant. It is however important to emphasize that Coast also can be used with great success in more "conventional" types of alarm systems, maybe only to filter alarms. Coast will then provide the filtered alarm lists for the conventional display system. The selective functionality in an alarm system described above is a type of advisory system which might be more common in the future. In any case Coast will feed the process displays with alarm information, and these displays have to be made to optimize the performance of the operator regardless of where they get the alarm information from.

3. STRUCTURE AND INTERFACES

Coast will contain facilities for building specific alarm systems, as well as facilities for alarm system

execution. It will be an integral part of the final alarm system, and not only act as a tool for building dedicated systems. Coast is shown in its final environment in Fig. 3.

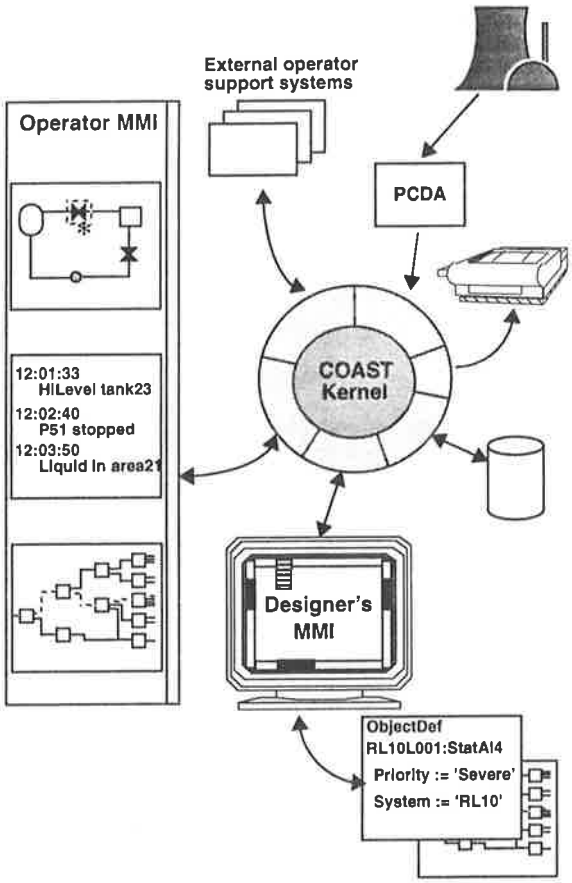


Fig. 3 The COmputerised Alarm System Toolbox, COAST, with interfaces to external systems.

Coast is meant to be an add-on possibility to conventional process control systems. As shown in Fig. 3, it receives process measurements from the process computer (PCDA - Process Control & Data Acquisition), updates all necessary statuses, and sends updated alarm information to the display system for the operators in the control room. Coast itself does not possess graphic capabilities, but will be easy to couple to different graphical systems. It has been coupled to the PICASSO-3 user interface management system developed in Halden (Barmsnes *et al.*, 1994). Coupling to all external systems is done through an application programmer's interface, which includes simple functions to get data in and out of Coast, e.g. process data must be provided as input to the alarm objects.

The designer's MMI will be designed as an alarm editor, but it will also be possible to operate Coast through text-files.

Coast will be tested in a full scope training simulator in the Halden man-machine laboratory, HAMMLAB. In this application the number of potential alarm signals is almost 5000, and Picasso-3 is used for graphical interfaces.

3.1 Main Modules

The main modules of Coast are shown in Fig. 4.

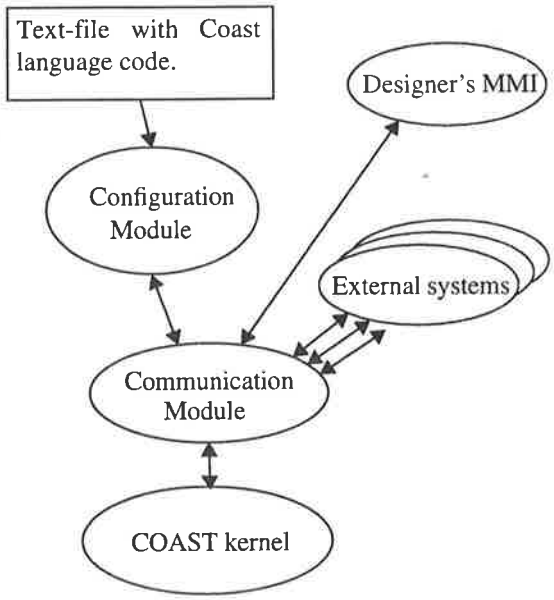


Fig. 4 Main modules of Coast.

The configuration module reads text-files with alarm codes given in the COast LAnguage (COLA), or interacts with the designer's MMI. The module checks the syntax of the Cola code, and sends a compiled version into the kernel. The communication module is generic, to take care of different types of systems to be coupled to Coast.

We consider incoming events as one out of two types: Either process-events, which contributes to new evaluations and updating of the data structures in the kernel, or operator-events, which are commands to select desired structures, or combinations of structures, from the kernel. The Coast kernel thus consists of two main modules, a process-event-module, including methods for updating the alarm objects and relations, and an operator-event-module, including methods to extract the desired alarm objects and relations for presentation. The process-event-module covers all the alarm generation and structuring, while the operator-event-module is dealing with all operator-requests and preparation for presentation.

4. CONCLUSION

The main difficulty with existing alarm systems is the cognitive overload of the operators in heavy transients. Coast offers the possibility to structure and make relations among alarms to reduce this problem.

Structuring includes both filtering the alarms automatically presented to the operator, and supporting the operator with different types of alarm lists, which he can use interactively in his status identification task in case of disturbances. The alarm presentation may thus include selectable displays, which the operator can use interactively, in addition to other alarm displays and the ordinary process displays.

Coast will contain facilities to build specific alarm systems for many types of processes. However, it is not only a toolbox for building the alarm system, it is also an integral part of the final on-line alarm system. The on-line system interacts with the display system and other external systems, updating and evaluating the data structures which initially are made by the alarm system designer.

Coupling Coast to external systems is a simple task with the application programmer's interface. This provides a set of remote functions for data exchange.

Coast is able to generate most types of alarms, and the different methods may depend on plant state and type of plant. Coast is also able to read already generated alarms from other systems to modify its own behaviour.

Coast input may be done in two ways: 1) through an interactive graphic Man Machine Interface, or 2) by means of text-files. The text-files must contain source code according to the specific alarm syntax defined in the COast LAnguage COLA.

A subset of Cola provides strong selection capabilities, for on-line selections among the automatically updated set of alarm objects.

The data driven kernel of Coast ensures consistency of all data, and best possible efficiency.

5. REFERENCES

Barmsnes, K.A., Ø. Jakobsen, T. Johnsen and H.O. Randem (1994). Developing Graphics Applications in an Interactive Environment. In: *Proceedings, 1994 SCS Simulation Multiconference,* San Diego, California, USA.

Kemeny, J.G. (1979). Final Report of the President's Commission on the Accident at Three Mile Island. Washington, D.C.

Marshall, E., C. Reiersen and F. Øwre (1987). Operator Performance with the HALO II Advanced Alarm System for Nuclear Power Plants - A Comparative Study. In: *Proceedings of the ANS Topical Meeting on Artificial Intelligence and Other Innovative Computer Applications in the Nuclear Industry*, Snowbird, Utah, USA.

Sørenssen, A. (1990). Early Fault Detection at the Loviisa Nuclear Power Plant by Simulation methods. In: *Modelling & Simulation, Proceedings of the 1990 European Simulation Multiconference*, Nuremberg, Germany.

Øwre, F. and E. Marshall (1986). HALO - Handling of Alarms using LOgic: Background, Status and Future Plans. In: *Proceedings of the International ANS/ENS Topical Meeting on Advances in Human Factors in Nuclear Power Systems*, Knoxville, Tennessee, USA.

Øwre, F., S. Nilsen, T. Forsman and J.E. Stenmark (1991). An Operator Support System for a Swedish Nuclear Power Plant Control Room. In: *Proceedings, EPRI conference on Expert System Applications for the Electric Power Industry*, Boston, Massachusetts, USA.

ADVANCED ALARM-MANAGEMENT POST-PROCESSOR PROTOTYPE FOR A FOSSIL POWER STATION

S. GLICKMAN[#], A. DEZSU[*], and G.Gy. HALASZ[*],
[*]*Israel Electric Corporation Ltd. Power Stations Planning Division, P.O.B. 10, 31000 Haifa, Israel*
[#]*Israel Electric Corporation Ltd., Rutenberg Power Station, Ashkelon, Israel*

Abstract During a power station disturbance, the data processing and monitoring system (DPMS) may present an overwhelming number of alarms (200-400) to the power station operators within a short period of time. This may lead to operator confusion and makes problem analysis and decisions difficult. An advanced post-processor connected to the existing DPMS is being developed at IEC. In the present stage of this project our aim is to develop a model for dynamic alarm management and adaptive operator guidance that will permit the verification of the implementation methodology. The hierarchical structure of the power station is mapping the different equipment features and technological processes, to the structure used by our alarm-management system. On the basis of this double hierarchy, all the DPMS points were grouped and each DPMS point labeled with its alarm priority level (i.e., urgent alarm, alarm, warning, not an alarm). Cause-consequence graphs were constructed. The nodes of the graphs were the selected points of the DPMS. According to our methodological proposal the cause-consequence connections between the selected points of the DPMS may be constructed on the basis of the ready made 'flow diagram - process and instrument diagrams', 'control logic diagrams' and 'control and instrument diagrams' of the power station documentation. Our prototype includes some limited parts of the power station such as deareator level control, one of the turbine driven boiler feed-water pumps and cooling system of the unit main transformer. About 100 rules were generated from the cause-consequence graphs. Some 200 inputs were simulated off-line. The user interface and the inference machine were built by using the KAPPA PC (IntelliCorp, Inc.) expert system shell. This prototype is being tested now in the Training Center of the Operation Division.

Key Words. Alarm systems; Expert systems; Failure detection; Power station; Steam plants; Supervisory Control

1. INTRODUCTION

Most of the systems in operation today in fossil power stations give alarms in response to signals and leave it to the operator to diagnose the fault. In the modern power stations of the Israel Electric Corporation (IEC) the data processing and monitoring system (DPMS) works on the same principle. However, the effective guidance in power station's control room requires intelligent and real-time alarm management software products. Nowadays, different real-time expert system tools support advanced alarm management aims, for example G2 (Gensym Corporation), EXSYS (EXSYS, Inc.). The final goal of an intelligent alarm management system is to provide the control room operator with the necessary information in order to minimize economic losses. This strategy complements the tactical response of control room personnel by providing a strategic overview of developing events and accidents.

The knowledge base of an application is the most expensive part of an expert system. For example, 10 men-years were needed to develop an emergency control knowledge base for a power station (Talukrad 1986). The success rate and the scale of the real time application of the A.I. in the control room of P.S. (power station) are seriously different in nuclear and fossil P.S., obviously because of the different safety requirements. Some examples of the real-time A.I. systems which have been imple-

mented successfully in nuclear power plant control rooms in safety-critical technologies: Owre (1990), Beck (1992), Cheon (1993). In spite of the large amount of literature studied (see, for example a good review of Chen 1992) and our wide personal contacts with other Electric Companies in the US, France and Germany, we were capable of finding only one large scale, real time, non prototype like application in the control room of a fossil P.S. (Staudinger P.S., Germany). As a result of the difference between IEC's P.S. and Staudinger P.S. and due to the IEC experience with the planning and operating of different P.S.s we belive that the knowledge base should be developed in IEC itself. However, the general structure of the knowledge base, some important time dependencies, the alarm filtering methodology (Crosberg 1987) etc. should be learned and applied from the literature. An other important practical question is the validation of the knowledge base. Kirk (1988) describes a different methodology for verification and validation of the knowledge base of the expert system. Unfortunately, up to now the authors could not find any solid and applicable methods for validation and verification of the logical consistency and completeness of the knowledge base.

This paper presents our know-how for the building and verification of the large scale knowledge base for a fossil P.S.. In the first part, different hierarchy layers and alarm priority levels are described in Chapter 2. The generation of the cause-consequence graphs from the 'flow diagrams - processes and instrument diagrams', 'control and instrument diagrams' and 'control logic diagrams' is shown in Chapter 3. Chapter 4. presents a turbine driven boiler feedwater pump (TDBFP) application as the focus of our prototype.

2. DESCRIPTION OF THE HIERARCHY AND THE ALARM PRIORITIES

In this section the main principles of the hierarchical structure of the P.S. and the alarm priorities will be described.

Today, typically about six thousand points in a P.S. unit are monitored by the data processing and monitoring system (DPMS). Half of them may indicate alarm states. In order to handle this huge amount of data, a special hierarchical structure suited to a P.S. was constructed. In each final group of this hierarchy 10-50 DPMS alarm points are selected.

2.1 Definition of the Alarm Hierarchy

First hierarchy level: P.S. unit. The following main areas (top layer of the hierarchy) have been proposed: (the percentage shown is the portion of the total number of possible sources of alarms):

1. *Main Cycle*, such as boiler, turbogenerator, feedwater pump system, main-steam, sea-water etc.(65%),
2. *Service and Auxiliary Systems*, such as compressed air, fire protection, ash handling, etc. (10%),
3. *Common Systems of the P.S. Units*, such as fuel oil, waste water, etc. (5%),
4. *Chemical Treatment Systems*, such as water treatment, chemical additives, chlorination, etc. (5%),
5. *Electrical Systems*, such as switchyards, transformers, emergency power, etc. (15%).

Second hierarchy level: main equipment or processes.

On the second layer of the hierarchy are the main equipment or processes such as boilerfans, airheaters, condensate system, feedwater pumps, etc.

Third hierarchy level: services or auxiliaries of the second level.

On the third hierarchy level are typically the essential parts and the main service equipment of the second layers' elements, for example the lubrication oil system of the feedwater pump.

2.2. Definition of the Alarm Priorities

Another important tool to build the knowledge base are the alarm priorities. Each fault event has its importance from the alarm management point of view. The Fig.1. demonstrates the relations of the different alarm priorities from the operator's point of view:

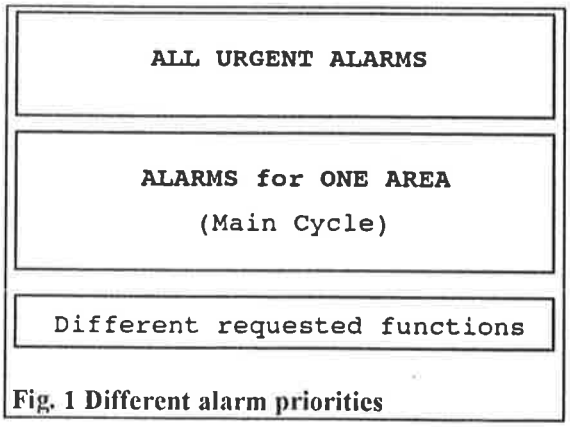


Fig. 1 Different alarm priorities

The following alarm priorities were used: Urgent Alarm, Alarm, Warning, Not in Alarm.

Urgent Alarm

Required operator action:

Urgent, well based and considered action by control room personnel. The operator may not suppress or ignore these messages.

Selected cases:

-safety hazard

-pre-trip condition of the P.S. unit *or* of the main cycle systems *or* of the main electrical systems or of any main equipment without reserve,

-hazard of trip or damage to one of the main cycle systems or equipment within few minutes,

-unknown condition of the main control or main safety systems or equipment,

-fault in the main control or safety systems or equipment.

Alarm

Required operator action:

Operator action required only if all messages with URGENT ALARM have been already properly dealt with. The operator has the right to ignore temporally these messages, but acknowledgment of the information is required.

Selected cases:

-hazard of trip or damage within 10-15 min. of the P.S. unit, or of the main cycle systems or of the main electrical systems

-automatic trip of a control loop to manual

-unknown condition of the control or safety systems of equipment.

Warning

Required operator action:

No operator action required. The operator may temporally ignore warning-messages but acknowledgment of the information at least once during the shift is required.

3. CONSTRUCTION OF THE CAUSE-CONSEQUENCE GRAPHS

The *knowledge base* of our advanced alarm management post-processor has four sources: 'flow diagrams - process and instrument diagrams' (P&ID), 'control and instrument diagrams' (C&ID) 'control logic diagrams' (CLD) and the knowledge of various field experts. The first three sources are available as a part of the P.S. plant documentation. These ready-made well-structured sources must be well utilized. The fourth source, namely the human experts, have to check and approve special logic diagrams built mainly on the basis of the three above mentioned sources. These special logic diagrams are called cause-consequence graphs. The 'P&ID' describes the physical and logical connections between the different equipment and processes. The 'C&ID' contains the location (which physical point is monitored), the type (temperature, pressure, etc.) of the specific measurement and its relation to the control system (trip, permit, set-point, information only point, etc.). The 'CLD' shows the logical conditions for start/stop/lock-out/trip of different equipment and logical-chains of the different events. The cause-consequence graphs' nodes are the selected points of the DPMS. The nodes are connected through AND, OR and NOT logic gates. According to the hierarchical structure (see Chapter 2) of the DPMS points, the most important question is, 'what are the possible causes for the P.S. unit to trip?'. Later on, going down by one layer in the hierarchy, the cause-consequence graphs for the main equipment or processes are built (for example: turbo-generator trip). On this level the relevant question is: 'what are the possible causes of a trip of a specific piece of main equipment?' or 'what are the possible causes for the alarm status of the specific equipment or process?' The majority of the answers to the above questions are in the 'CLD' and 'C&ID' drawings which contain the logical and physical conditions for the trips, pretrips and alarms on each layer of the hierarchy. The logical connection between the DPMS points of different equipment required for the cause-consequences graphs may be traced by the help of the 'P&ID'. The connection between the different hierarchy-levels is built by the 'alarm status' of the equipment or process on the given (lower) level. For example, one of the feed-water pumps may be in one of the following states: normal, warning, alarm, urgent alarm. It means that on the hierarchy layer of the P.S. unit we need to know only the status of the feed-water pump and not the possible causes of its fault.

This information flow is demonstrated in the Fig.2. On the same hierarchy layer the DPMS points are labeled also with their alarm priority (Chapter 2). The cause-consequence graphs of a P.S. unit have in our approach a ***double hierarchy***: On the same hierarchy layer, the DPMS layer (or hierarchy level) and within each layer alarm priority. The data **Processing** begins after each data refresh from the DPMS (about 2 sec.). **First**, we scan the DPMS points belonging to the first two (top) layers. If at least one point is firing (in a fault condition) in the highest layer, then the most important (highest priority) firing point(s) are displayed immediately. Otherwise, we will go down by one layer and process it. **Second**, the inference machine starts to work from the firing points of the second layer in the direction of the ultimate consequence, namely in upward direction in the Fig. 2. As a result of this procedure, the status of the connection points between

the second and first layer is determined. **Third**, the inference machine starts to work from the firing point(s) within the upper layer in both directions: the causes and consequences are chained for each

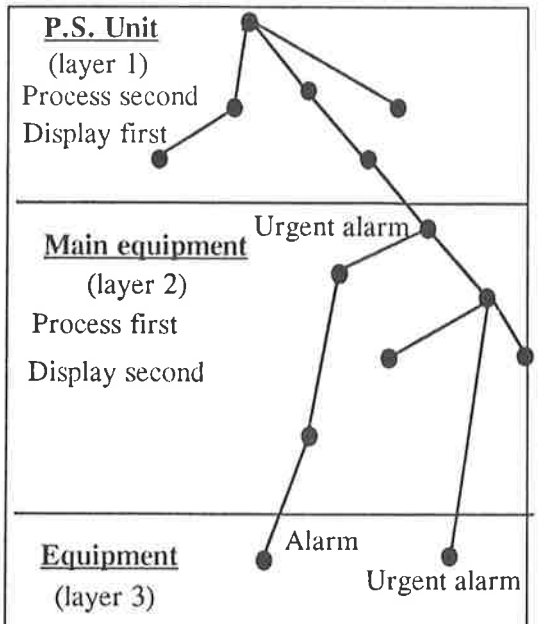


Fig.2. Information flow between the different layers

firing point. The result is displayed only on operator request.

4. EXAMPLE

The boiler feed water pump (BFP) system is one of the main systems of the P.S. unit. Its function is to force the required feedwater into the boiler by increasing its pressure to about 203 atm at full load. At full load, boiler feedwater flow is 1700 m^3/h. At Rutenberg P.S., Israel Electric's newest station, the feedwater system is comprised of three pumps, two of them are turbine (steam) driven (TDBFP) and one is motor driven. At full load, the normal operating configuration is with the two steam driven pumps in operation, and the motor driven pump in hot reserve (standby). At the *top layer* (P.S. unit) the BFP system may be in urgent alarm, alarm, warning or not in alarm (O.K.) status. If the P.S. unit is operating at full load, then the simplified table (Table 1.) shows the status of the BFP system:

Table 1. Different statuses of the BFP system

status of reserve	status of TDBFP1	status of the system
yes	O.K.	O.K.
yes	Urgent	Warning
yes	Alarm	Warning
no	O.K.	O.K.
no	Urgent	Urgent
no	Alarm	Urgent

The status of the BFP system connects the two layers (P.S. unit and main equipment or processes). On the second layer (the level of the TDBFP1 as main equipment), the Table 2. shows the DPMS points used in this example:

Table 2. Selected DPMS points in example

Description of fault	Fault status	Status of TDBFP1
bearing temp. of turbine	<85	Alarm
bearing temp. of turbine	<90	Alarm
bearing temp. of turbine	high (<95)	Urgent
bearing temp. of turbine	high-high (<100)	Urgent
oil flow to main pump lubrications	low	Alarm

On the basis of the above data and the 'C&ID' and 'CLD' drawings, parts of the cause-consequence graph are shown in the Fig. 3 and 4. at the end of this paper. These two cause-consequence graphs demonstrate the connection between the third and second layers.
In each alarm-node of the graph there are the alarm priority (PRIOR) and an identification tag (for example: A2OV042) making connection with DPMS and the corresponding 'C&ID' and 'CLD' drawings.

5. CONCLUSIONS

Our prototype includes some limited parts of the P.S. such as deareator level control, one of the turbine driven boiler feed-water pumps and cooling of the unit main transformer. About 100 rules were generated for the cause-consequence graphs.
Three hierarchy layers were used:

- P.S. unit,
- main equipment or process (primary)
- secondary equipment or processes of the main equipment or process.

Some 200 inputs of DPMS were simulated off-line. The user interface and the inference machine were built by using the KAPPA (IntelliCorp Inc.) expert

system shell. This prototype is being tested now in the Operations Division Training Center. During the off line testing period we have to finalize our methodological approach. More over, the operators from different P.S.s may provide us with very valuable feedback. During the operators' training period, they are familiarizing themselves with the new approach. We hope they will come to appreciate the advantages of the advanced alarm-management post-processor, and at their initiative, will request - as the final users - that this approach be applied in the control rooms of the various P.S.s of IEC.

References

Beck, C.E., Behera, A.K., Reed, M.L.(1992). Expert System Apllications in Nuclear Plants:Discussion of the Key Issues. *IEEE Transaction on Nuclear Sciences* **39**(5), 1363-1368

Chen-Ching Liu, Dillon, T.(1992). State-of-the-the-art of expert system applications to power systems. *Electrical Power & Energy Systems* **14**(2/3), 86-96

Cheon, S.W., Chang, S.H., Chung, H.Y. (1993). Development Strategies of an Expert System for Multiple Alarm Processing and Diagnosis in Nuclear Power Plants. *In: IEEE Transaction on Nuclear Sciences* **40**(1), 21-30

Corsberg, D.R.(1987). Functional Realtionship-Based-Alarm Processing. *Patents*-US-A7184927 DE89 009608, DE-AC07-761001570.

Kirk, D.B., Murray, A.E.(1988) (1988). Verification and Validation of Expert Systems for Nuclear Power Plant Applications. *Report:EPRI-NP-5987*, Project;2582-2586

Owre F.:(1990).Safety Assesment and Post Trip Quidence - SAS II system, HWR-262, OECD *Halden Reactor Project*, Nalden Norway

Talukdar, S. N., Cardozo, E. (1986). Artificial Intelligence Technologies for Power System Operations_EL-4323, *Research Project 1999-7*, January 1986, Carnegie-Mellon University

ACKNOWLEDGMENTS

We wish to express our thanks to the staff of the Operations Division Training Center of IEC, as well as the operating staff of Rutenberg Power Station.

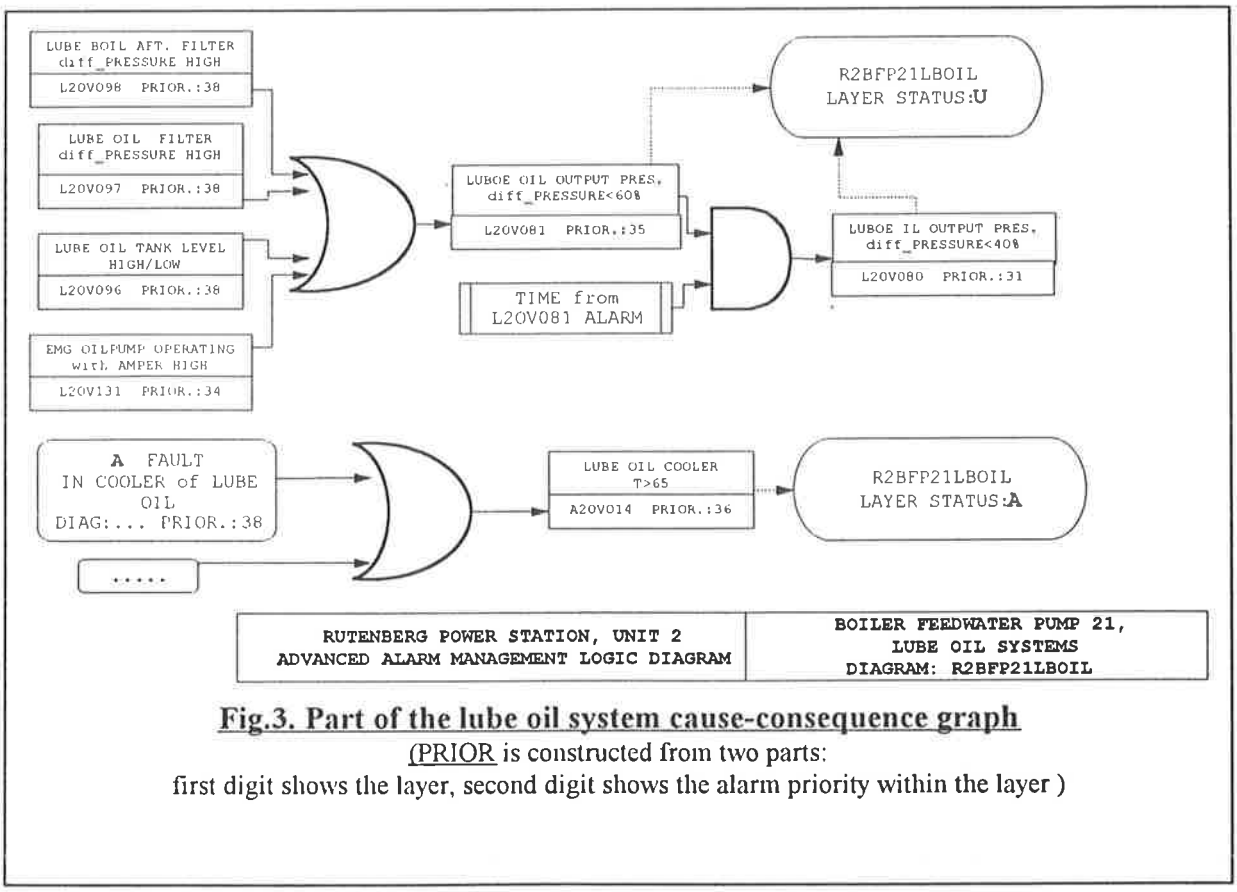


Fig.3. Part of the lube oil system cause-consequence graph

(PRIOR is constructed from two parts:
first digit shows the layer, second digit shows the alarm priority within the layer)

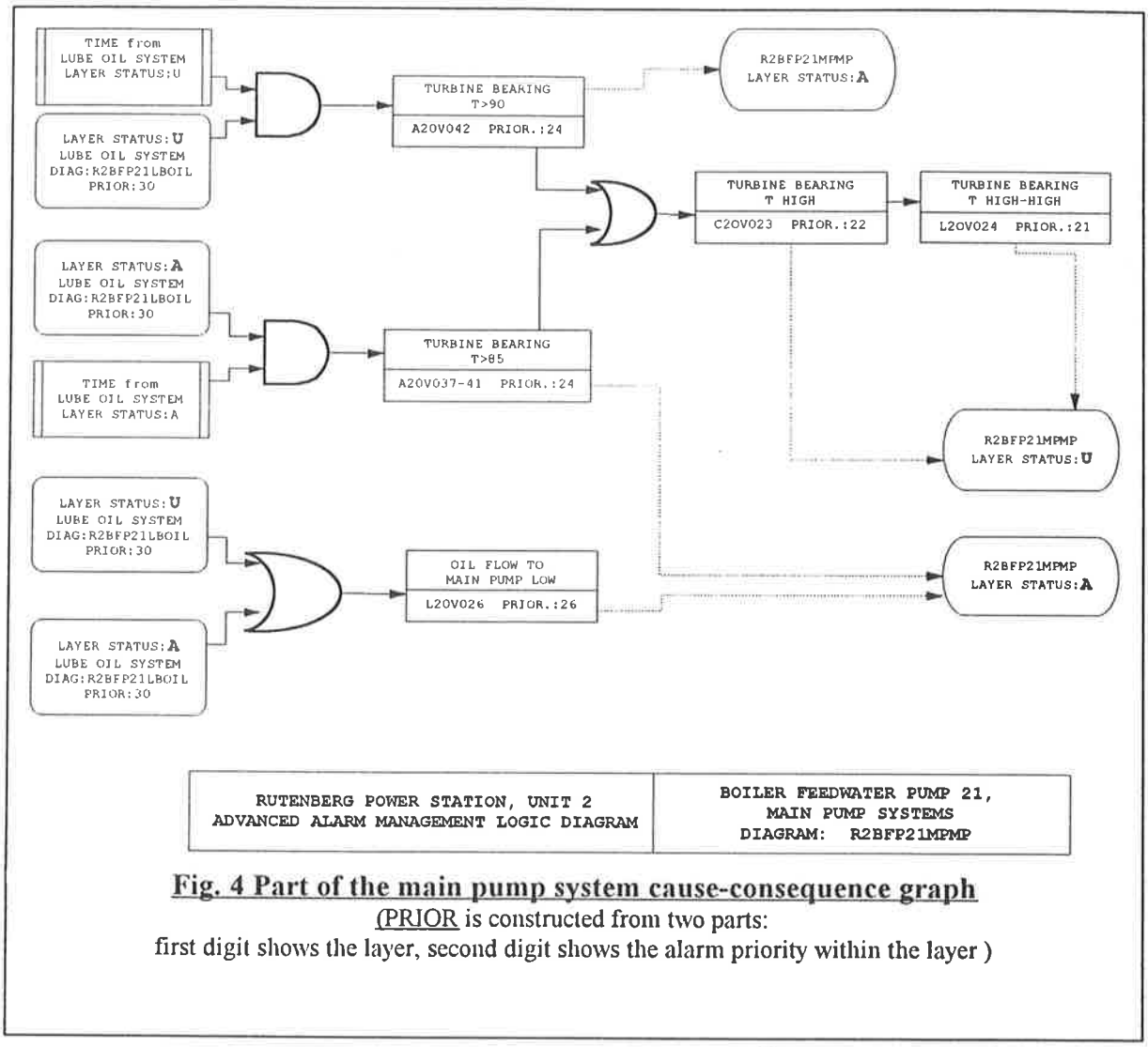


Fig. 4 Part of the main pump system cause-consequence graph

(PRIOR is constructed from two parts:
first digit shows the layer, second digit shows the alarm priority within the layer)

DIAGNOSIS OF THREE-PHASE CONVERTERS USING THE VQP NEURAL NETWORK

G. CIRRINCIONE (*), M. CIRRINCIONE (**) and G. VITALE (***)

* *INPG, Labo. TIRF, 46, av. Félix-Viallet F-38031 Grenoble, France.*
** *Dipartimento di Ingegneria Elettrica dell'Università degli studi di Palermo, viale delle Scienze, Palermo., Italia*
*** *CE.RI.S.E.P. CEntro RIcerche sui Sistemi Elettrici di Potenza del CNR, c/o Dipartimento di Ingegneria Elettrica, viale delle Scienze, Palermo. tel. +39-91-6566209/6, fax +39-91-488452*

Abstract. A neural network based diagnosis system is presented for fully controlled converters. The system, by monitoring the current harmonics, is able to recognise the fault conditions and represents them in a 2D and 3D space. After describing the VQP algorithm a case study regarding three-phase converters is presented. By use of the VQP algorithm two different representations of normal and fault conditions have been obtained. The benefits of such approach are then discussed.

Key Words: Power converters, neural nets, system failure and recovery, pattern recognition, self-organising maps.

1. INTRODUCTION

A static power converter often requires high-level reliability in order to perform well. Unfortunately that is not always the case and therefore a real-time diagnosis system is required to avoid major damages to the driving system to which it belongs. Low cost and fast data processing should be achieved by non-invasive measurements. Several methods have been utilised for this purpose, such as microprocessors (Murty, 1984) or expert systems (Renfrew *et al.*, 1993). They both comply with these conditions, but they are not able to easily supply the operator with a visualisation of the different working states of the system. Thus neural networks have been suggested to overcome this problems, as the identification of the states of the monitored system is a typical pattern recognition problem (Sorsa *et al.*, 1991). A Kohonen neural map (Kohonen, 1989; Kohonen 1990) has already been utilised for a diagnostic system (Cirrincione *et al.*, 1994). As this kind of neural network is a self-organising map (SOM) a classification of the states of the system has been achieved. The learning process forming a SOM is unsupervised, i.e. no output data are presented. Moreover the input topology is preserved, so that similar input vectors, at least in the euclidean sense, are mapped into neighbouring regions. Self-organising maps, therefore, provide a simple representation of the several working conditions of the system and permit an easy quick diagnosis. The operator can have a view of what is going on, since a map is obtained where the clusters of each working state are represented. Each point of these clusters is a processing element (PE) which is fired whenever the conditions that it classifies are encountered. In this way if a fault occurs, the corresponding unit, representing this abnormal condition, fires thus allowing the on-line diagnosis and the visualization of the working states.

More components in the feature vector may be treated, but some problems may arise if the input vector is too complex, as input data may form clusters that are strongly folded. The principal problem for the projection from a high-dimensional input space to a lower-dimensional output space is to determine the smallest set of independent variables in order to describe redundant non-linear data, that is to determine the so-called variety of the data set. Furthermore very complex clusters may be split and projected into regions that may be far away from each other. If a Kohonen neural network is used, that is a predefined lattice of neurons and an a priori knowledge of the variety of the data set is required, there may be strong distortions and also the well known problem of dead units, i.e. some units not taking part in the learning process. In order to overcome all these difficulties the units should not be fixed on a defined lattice, but must be free to find their position in the output space. That is the case of the VQP algorithm (Demartines, Hérault, 1994; Demartines, Hérault , 1993; Demartines, Hérault, 1992) , which seems more flexible than Kohonen networks in creating maps useful for diagnostics.

2. THE VQP ALGORITHM

The VQP algorithm has been recently discovered by J.Hérault and P.Demartines. In the first partthis work only the fundamentals are described.

The network is composed of two layers, as can be seen in fig. 1. The first layer realises a vector quantization of the input distribution, while the second layer reproduces the local configuration of the first one in a self-organised way. The network projects any n-dimensional input vector **x** into a vector **y** with dimension p, which can take on any integer value, but it is often less than n. The topology of the input space is preserved, as in Kohonen maps. In fact **y** is defined in such a way that the topology of the input space is preserved in terms of distance conservation. Only p is defined a priori. As in Kohonen maps each i-th neuron may be considered to point to the input space with their own weight vector. But, in addition, in the VQP algorithm, each i-th neuron points also to the output space. In Kohonen maps the output space is the pre-defined lattice of neurons, so that there is no control of the physical location of the neurons.

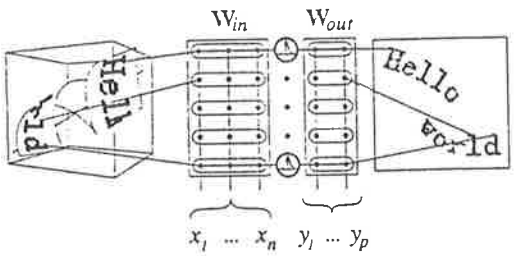


Fig 1: Structure of the VQP network

The first layer realizes a vector quantization of the input space. When an input vector **x** is presented the activity a_i of the i-th neuron is computed by the following formula

$$a_i = \exp - \left(\frac{\left\| \mathbf{x} - \mathbf{w}_{in,i} \right\|^2}{\lambda_i^2} \right) \qquad (1)$$

where $\mathbf{w}_{in,i}$ is the input weight and l_i is the influence radius of the i-th neuron. Then the output projection **y** is computed by using these activities as well as the output weights of all neurons, i.e.

$$\mathbf{y} = \frac{\sum_{i=1}^{N} a_i \mathbf{w}_{out,i}}{\sum_{i=1}^{N} a_i} \qquad (2)$$

where N is the number of neurons. The parameters that must be adapted in the learning process are l_i , $\mathbf{w}_{in,i}$ and $\mathbf{w}_{out,i}$.

As to the training of the first layer, different techniques have been used to carry out the quantization of the input space. More generally a competitive learning (Kohonen, 1989) or the neural gas algorithm (Martinetz *et al.* 1991) have been used. As to the second layer , i.e. the projection layer, it aims at preserving the topology of the first one: if two neurons are near each other, they should be also near in the output space and if they are distant in the input space, they should be distant in the output space. This is achieved by maximizing the correlation of the weight distances between the output and input spaces.

Then the following algorithm is used to adapt the projection layer:

$$\mathbf{w}_{out,i} \leftarrow \mathbf{w}_{out,i} + \alpha_{out} \beta_{i,w} \left(\mathbf{w}_{out,w} - \mathbf{w}_{out,i} \right) \quad \forall i \qquad (3)$$

where w stands for the winning neuron, i the i-th neuron of the same layer, α_{out} is a parameter or learning factor, while $\beta_{i,w}$ is given by

$$\beta_{i,w} = \frac{dy_{i,w} - dx_{i,w}}{dx_{i,w} + dy_{i,w}} H\left(dy_{i,w} \right) \qquad (4)$$

where $dx_{i,w}$ ($dy_{i,w}$) is the input (output) weight distance between the neurons i and the winning neuron w. H is given by

$$H(dy) = \frac{1}{1 + \left(dy / dy_0 \right)^p} \qquad (5)$$

where dy_0 is a parameter that gives an order of value for dy and p is a parameter controlling how quickly the H factor has to fall. In this way for long distances $\beta_{i,w}$ decreases sharply so that short distances are given more importance than long distances: the projection, therefore, is locally correct.

In order to visualize the correspondence between input and output space, the graphical representation called dx-dy relation (Demartines, 1992) is used. In this kind of visualization some pairs of units are randomly selected and then their distance is computed both in the input space (dx value) and in the output space (dy value). Then these points, with coordinates dx and dy, are plotted on a 2D plot. If the projection were linear

these points would all be aligned, otherwise they give rise to a cloud of points.

3. VQP vs. KOHONEN SOM

In [Cirrincione *et.al.*,1994] the Kohnen Som has been used. In this paragraph the reasons for the use of the VQP are discussed.

As it deeply known, Kohonen SOM's are among the most famous self-organising neural networks. Inspired by biological observations (Malsburg, 1973; Willshaw, 1976; Linsker, 1988), they give very interesting and simple model of sensory mapping. They are also of techical interest by building topological features map of high-dimensional-data, some applications of which can be found in Kohonen, 1990. As a matter of fact SOM's can be looked upon as non-linear extensions of Principal Component Analysis (PCA) (Blayo *et al*, 1991), where the linear projection subspace is replaced by a non linear manifold.

However SOM's suffer from several restrictions. In their basic form they provide a discrete projection of input vectors by using the lattice index of the winner of the competitive layer. In further applications several neurons and their respective activities are taken into account and as a result it is possible to find a continuous position of output by interpolation (Cheneval *et al.*, 1992, Hecht-Nielsen, 1987; Hecht-Nielsen, 1988). Unfortunately the existing methods of performing this interpolation are sensitive to parameters that are hard to define properly, such as the radii of kernels for kernel-based interpolation.

The main problem of SOM's is that, as the shape of the lattice of neurons is predifined (square, cube, and so on), it is impossible to know if it complies with the shape of the sub-manifold spanned by the data. For example a cactus cannot be mapped completely into a 2D-grid as such mapping does not manage to respect the neighbourhood of the input data; consequently the problem of dead-units arises, i.e. useless units not taking part in the distribution. Practically a large dimensionality reduction that keeps correct topological information by using such predefined maps is almost impossible. The result strongly depends on the intrisic shape and dimensionality of the sub-manifold (Demartines, 1992).

In comparison the VQP is also a kind of self-organising network. However, contrary to Kohonen maps where neurons are fixed on a priori defined discrete lattice, VQP neurons find their position in a continuous output projection space through a self-learning algorithm. The main property is therefore the ability to map arbitrary shapes of input distributions and to project them in a non-linear way, even if the input data sub-manifold is strongly folded.

After the learning phase the VQP provides a full correspondence between input and output spaces. The morphism is continuous and is usable in direct mode (input - output) or in reverse mode (output-input). The main parameters are the number of neurons, the input and the output dimension.

Formally it is conceivable to make Kohonen maps with more than two output dimensions (or lattice dimensions). Unfortunately in this case the neighbourhood computation becomes cumbersome. Furthermore the above-mentioned problem of the lattice shape definition becomes much harder than for the two output dimensions. For these reasons Kohonen maps do not generally have more than 2 or 3 lattice dimensions. With the VQP the output dimension is not materialised by the neighbourhood connectivity, but it is simply the number of components of output vectors. It is therefore impossible. It is therefore possible to have any output dimension. This is useful when the intrinsical dimension of data (the dimension of the submanifold) is greater than 2.

Finally the function realised by the VQP is the minimization of an explicit and simple error function which considers the matching between input and output distances of neurons pairs. This criterion, i.e. the quality of the topological mapping, can be drawn on a 2d graphic showing this matching between distances and that is valid for any input and output dimension. This graphic is very useful for determining visually the convergence of the VQP algorithm.

4. FAULT DIAGNOSIS BY VQP: A CASE STUDY

The authors have considered an application of the VQP to a three-phase PWM inverter with a six-switch inverter stage . The VQP algorithm, by using the FFT of the currents ot the phases, is able to give a visualisation of the working conditions (fig. 2).

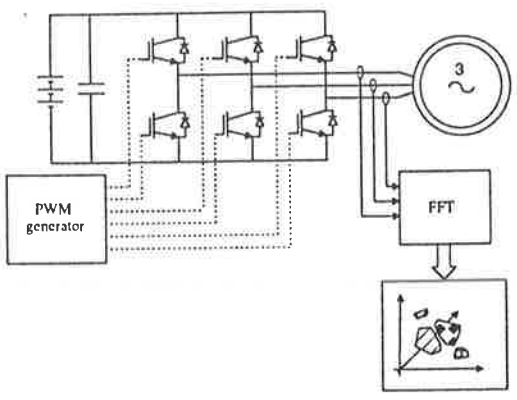


Fig 2: Inverter stage

Faults were considered on all power electronic devices in different mutual combinations. The cases considered are as follows:

- normal state
- one upper arm in short circuit
- one lower arm in short circuit
- one upper arm with power switch open
- one lower arm with power switch open
- one upper arm with both power switch and diode open
- one lower arm with both power switch and diode open
- one upper arm with diode open
- one upper power switch open and lower diode open in the same arm
- one upper power switch open and lower diode open in different arms
- two power switches open in the same arm
- two upper power switches open in different arms
- one upper power switch open and one lower power switch open in different arms.

(open stands for the interruption of the component due to a fault).

The input vector has been made by taking the DC component and the first five harmonics of the phase currents. It has been obtained a vector with 18 component. Afterwards all the harmonics have been normalized to the value of the fundamental harmonics; so the input vector has been reduced to 15 harmonics. The normalization allows the system to be indipendent from the absolute values of the moduli of the Fourier analysis.Furthermore a scaling of the components of the input vectors was introduced to get components of the same range: all kth components (k=1,...,15) have been normalized so as to have zero mean and variance equals to one; in this way a component with very small value has the same impact on the structuring of the map than a component with a large one

The inverter circuit was simulated by using PSPICE, with devices selected from the PSPICE device library (Rashid, 1989).The power switch has been modelled by series connecting a library switch (Ron=0,01 ohm) and a power diode (Cjo=0.001 fF, IS=1e-6, RS=0.01), where Cjo is the zero-bias pn capacitance, IS is the saturation current, and RS is the parasitic resistance.
The Fourier analysis was carried out by using the facilities inherent in PSPICE.

Nine different load conditions have been considered, as well as 12 different faults and the normal state. Thus the training set was formed of 13x9=117 vectors. Each n-th component has then been normalized so as to have zero mean and variance equals to one.

After the normalization in the preprocessing face, the vectors have been given as input to the VQP network. The perameters that have been choosen for the projection phases are as follows:

- α_{out} is a decreasing function as in fig. 3.

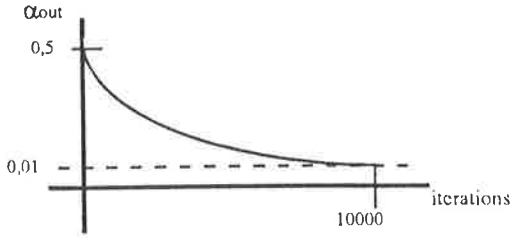


Fig. 3: learning factor vs. iterations

- dy_O has a linear decreasing function as in fig. 4.

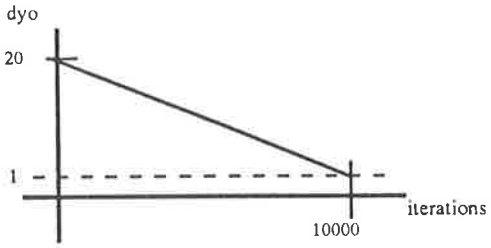


Fig. 4: dy_O vs. iterations

- p = 9
- the output space was before a 2-dimensional space and after a 3-dimensional one.

All parameters have been determined experimentally.

5. RESULTS

Figure 5 shows the dy-dx diagram of output and input distances in the case of three dimensions. This diagram shows also the final value of dy_O (the vertical line) and the weighting function β. As for all folded maps, but with good local preservation of topology, the relation begins approximately with a line, but scatters for large grid distances.

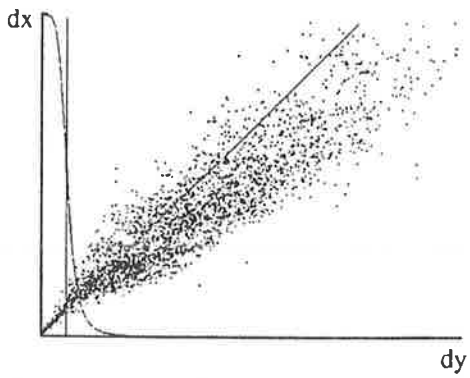


Fig 5: dx-dy diagram for the 3-dimensional case

The use of this relation in VQP, where neurons have a floating position in the output space, is straightforward: the horizontal values dy no longer refer to grid distances as in Kohonen maps, but to the output weight distances between units, while the dx values remain the input weight distances between units. The goal of the VQP is to obtain a dx-dy relation which is a thin line, therefore the projection is topologically correct, at least locally.

Figure 6 shows the 2-dimensional map of the VQP algorithm for the 14 cases presented to it. It can be seen by output data analysis that the normal state (NF) includes also the case of one upper arm with diode open. Any fault due to diode problems is hard to classify; in particular the one upper arm with power switch open, one upper power switch open and lower diode open in the same arm, one upper power switch open and lower diode open in different arms are classified by the same cluster (B). One lower arm with both power switch and diode open, one upper arm with both power switch and diode open are classified apart (M). Furthermore it can be seen that one lower arm with power switch open (P), two upper power switches open in different arms (W), one upper power switch open and one lower power switch open in different arms (G), two power switches open in the same arm (S) are mapped into separate regions. Finally, the case of one upper arm in short circuit (R) and one lower arm in short circuit (V) are represented by two regions each of which consists of two parts. These "split" regions are due to the fact that the 2-dimensional grid is not so good for such projection. That is the projection is from a very folded input space.

By using the 3-dimensional grid (see fig. 7), it can be seen by output data analysis that all regions are not spilt any longer, which means that each state is represented by one and only one cluster. The variety of input data is then 3.

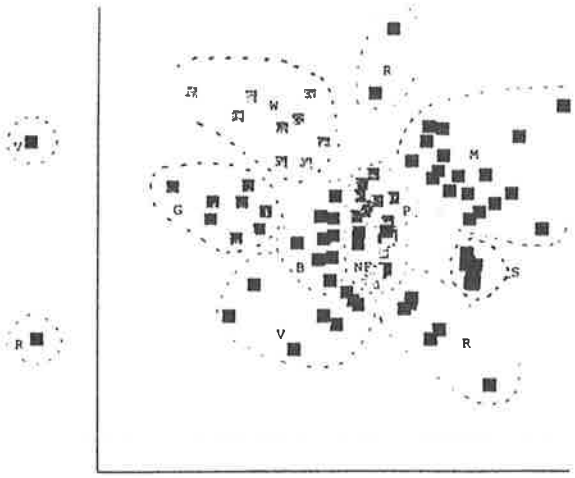


Fig 6: VQP map for the 2-dimensional case

The dx-dy diagram itself shows thegood projection. It can be seen that a 4-dimensional output space does not lead to any major improvement, that is the span of linear data in fig. 5 does not increase very much, which is an additional proof that the variety of the input data set is 3.

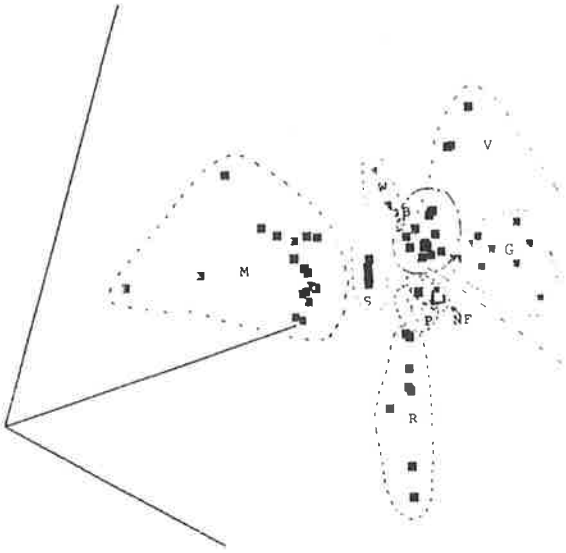


Fig 7: VQP map for the 3-dimensional case

In order to test the VQP map one hundred unknown vectors, i.e. not belonging to the training set have been given to the neural network. The error rate has been about 5%.

6. CONCLUSIONS

In this work an application of a new neural network, the VQP, to the diagnosis of the working states of a converter has been considered. As a result of the simulations that have been made, it can be seen that the use of a VQP Neural Network can easily lead the determination of the variety of the input data as well as supply the personnel with a visualization of the working condition of the system: in fact, when no faults occur, the state vector moves inside the cluster of the normal state , otherwise it moves into one of the other clusters representing the fault conditions. Thus an easy-to-interpret map is available to personnel or to other external recovery system.

With respect to Kohonen maps, VQP maps are more flexible because of the lack of a predefined lattice of neurons. A better clustering of the states of the system is obtained as well as a good interpolation and generalization.

7. REFERENCES

Blayo F. and Demartines P.(1991), "Data analysis: how to compare Kohonen neural

network to other techniques? In A. Prieto, editor, International workshop on artificial neural networks, vol. 540 of lecture Note in *Computer Science*, pages 469-476, Springer Verlag.

Cheneval Y., Demartines P., Tettoni L. 1992, "Function approximation using an architecture based on Kohonen self-organising maps", in J. Hérault, editor, *First IFIP Working Group 10/6 Workshop*.

Cirrincione G., Cirrincione M. , Vitale G. (1994), "Fault diagnosis in three-phase converters by using the Kohonen neural network classifier" SPEEDAM, Taormina, Italy, June 8-10.

Demartines P., Hérault J. (1994), "Apprentissage de structures de données par auto-organization; interpolation, extrapolation, généralization. In *Journées Neuroscience et Science de l'Ingénieur*, Chamonix.

Demartines P., Hérault J. (1993a), "Representation of nonlinear data structures through fast VQP neural network", In *Neuronîmes*, pages 411-424, october.

Demartines P., Hérault J. (1993b), "Vector quantization and projection neural network", in A. Prieto J. Mira J. Cabestany, editor, International Workshop on Artificial Neural Networks, volume 686 of Lecture Notes in *Computer Science*, pages 328-333. Springer Verlag.

Demartines P. (1992), "Organization measures and representation of Kohonen maps", in J. Hérault, editor, First IFIP Working Group 10.6 workshop.

Hecht-Nielsen R. 1987, "Counterpropagation networks", *Applied Optics* 26:4979-4984.

Hecht-Nielsen R. 1988, "Applications of ounterpropagation networks", *Neural Networks* 1:131-139.

Kohonen T. (1990), "Self-organizing maps", Proc. of the IEEE 78(9): 1464-1480 .

Kohonen T. (1989), "Self-Organization and associative memory", 3rd edition, Springer Verlag, Berlin.

Linsker R. (1988), "Self-organization in perceptual network", Computer pp.105-117, march.

Malsburg von der C. (1973), "Self-organization of orientation sensitive cell in striate cortex", Kybernetik, 14:85-100.

Martinetz T., Shulten K. (1991), "A neural gas networks learns topologies" , in T. Kohonen *et al.* editor, *IEEE ICANN Espoo*, Finland vol I, pp 397-407.

Murty Y. V. V. S. , Dubey G. K. (1984), "Fault Diagnosis in Three-Phase Thyristor Converters using microprocessor", IEEE transaction on Industry Applications, vol. IA-20, No 6 November/December.

Rashid H. M. (1989), "Spice for circuits and electronis using PSPICE", Prentice-Hall international Inc. .

Renfrew A. C. , J. X. Tian (1993), "The use of a knowledge system in power electronics circuit fault diagnosis", EPE, Brighton 13-16/9/1993.

Sorsa T., Koivo H., Koivisto H. 1991, "Neural network in process fault diagnosis", *IEEE trans. on Systems, Man and Cybernetics*, vol. 21, no 4.

Willshaw D. J. and von der Malsburg C. (1976), "How patterned neural connection can be set up by self-organization", Proceedings of the Royal Society of London, B 194:431-445 .

ACKNOWLEDGEMENTS

This work was supported by C.N.R. (Italian Research Council) and M.U.R.S.T. (Italian Ministry of University and Scientific and Tecnological Research).

Suggestions and comments can be sent to: CE.RI.S.E.P. CEntro RIcerche sui Sistemi Elettrici di Potenza del CNR, c/o Dipartimento di Ingegneria Elettrica, viale delle Scienze, Palermo. tel. +39-91-6566209/6, fax 39-91-488452

APPLICATION OF THE EXPERT CONTROL IN A SUGAR FACTORY

J. MICHAL, M. KMÍNEK, P. KMÍNEK

Department of Computing and Control Engineering, Institute of Chemical Technology, Technická 5, 166 28 Praha 6, The Czech Republic, E-mail: michalj@vscht.cz

Abstract : **This paper presents a method for using a rule-based expert system for the control of several technological processes in sugar production. The structure and principle of the expert controller are shown. Extraction, crystallisation and process co-ordination were chosen for practical implementation of the expert system as processes which are very difficult to control by classical means. Some experience from the implementation in the sugar factory in Lovosice is presented.**

Keywords: **Expert control; crystallisation control; vacuum pan crystallisation sugar extraction; co-ordination and supervisory control; control applications; expert systems; non-linear control systems; process control.**

1. INTRODUCTION

Expert control is a promising way to solve problems of complex technology control. As the complexity of technological processes increases, the problem of their control is becoming more critical. We focused our attention on a technological process whose behaviour is very difficult to define due to the number of nonlinearities and parameters that are impossible or difficult to estimate. In this case, classical feedback control cannot be used. One of the reasons for this is the difficulty of expressing the desired behaviour of the process in terms of classical control theory.

Sugar production is a technological process that has several parts that fulfil the above mentioned condition. Extraction of the sugar from the sugar beat, crystallisation process and co-ordination of the continuous and batch parts of the process are typical examples where the classical control fails both in theory as well as in practice.

The main aim of a control is to achieve profit. Let us look where is the expert control place in the scope of the control structure (fig.1). The traditional focus of process control has been at the lowest level, concerned only with the plant unit operations (Brisk,1993). In sugar production and in general in food industry the automation is restricted to maintain the working conditions in separate apparatuses. Supervisory actions which determine the overall performance of the process have been the role of the human operator. Due to a number of non measurable variables that are determined by a subjective judgement of the operator, automatic control based on mathematical description was not possible to implement. Different skills of the operators caused different results and therefore non constant quality of the production. In this case expert control can integrate theoretical background as well as practical skill of the best operator and therefore improve the performance of the whole process.

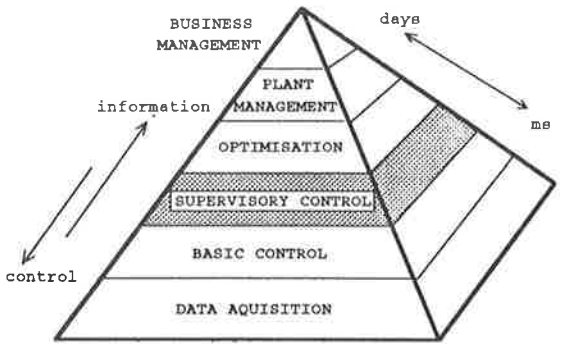


Fig.1 The global scope of a control structure of a plant

2. EXPERT CONTROL

In general expert control of technological processes acts according to the rules that are based on:

- thorough knowledge about the construction and instrumentation of the controlled system
- intuitive knowledge based on the operator's experience (rules of thumb).
- knowledge about the raw materials

Both thorough knowledge and intuition can be expressed in semantic rules as well as mathematical formulas, which allow us to express almost any desired fact about the properties of the controlled system and its behaviour. This information is concentrated in a knowledge-base by the knowledge engineer. One part of the knowledge-base is a semantic net; it consists of the statements and rules. Every rule links two statements together, an evidence to a hypothesis. In reality rules and statements are seldom categorical. The strength of the link is given either by a conditional probability or by a grade in fuzzy logic. In any case these values determine the power of the rule. They indicate how likely it is that the fact is true and reflect the uncertainty of the system.

The deterministic data are related by mathematical formulas or other algorithms, creating a parallel plane of the expert system (Fig. 2). Some facts in the rule plane can trigger computation of relevant algorithms and update the numerical values in the data plane. Some data, on the other hand, influence the semantic net.

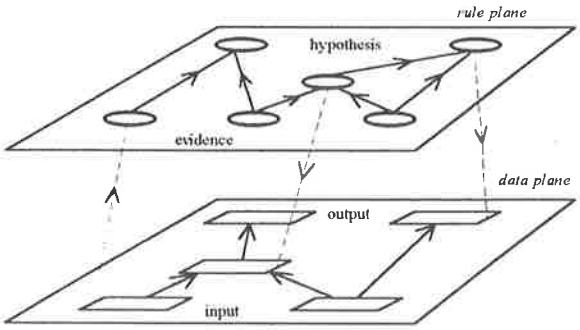


Fig. 2 Expert controller structure

Creating the knowledge-base structure for expert control can be divided into three stages.

In the first stage, the input data are examined from different points of view. The starting evidences are the input data of actual values of several - but not necessarily all - process variables. This knowledge is obtained by means of direct or indirect measurement or - especially in the food industry - by means of subjective human observation such as vision, smell or taste. The intermediate hypotheses which are to be proven at this stage concern properties of the input variables. According to the input data, the actual characteristics of these hypotheses are modified during the expert system run to reflect the actual stage of the system. Computation routines may be activated at some nodes of the semantic net to evaluate formulas expressing deterministic relation among data.

The second stage of the control starts with the proven hypotheses about the properties of input signals. Its task is to find out the domain of the present state of the controlled system. The whole state space of the controlled process can be divided into smaller parts or domains. All final hypotheses of this stage have the same form: "The present state of the system is in domain X_i". The differences in the characteristic of final hypotheses encountered in several succeeding runs will show the tendency of the controlled system behaviour.

Let X_i is set of possible states of the controlled system, R is the set of possible outputs of the expert controller and Y is the set of possible outputs of the system. Ψ is the mapping of the Cartesian product $X_i \times R$ to Y. A subset $Y_1 \subset Y$ is a set of wanted output (in case of classical feedback controller a setpoint). The control law can be expressed as a task that for every $x \in X_i$ will find such $r \in R$ that $\Psi(x,r) \in Y_1$ while x changes due to disturbances.

The last stage of the expert control must find out the appropriate controller output to force the system to follow the desired trajectory in the state space taking into consideration all criteria and limitations. Since we know from the second stage the actual domain X_i where the controlled system is and from the first stage the actual value of the output Y, the output of the controller R can be found if the mapping Ψ is known. Thus the knowledge base for the third stage must be constructed in order to realise mapping Ψ^{-1}.

3. EXTRACTION CONTROL

In sugar factories the continuous extraction plants are used to extract sugar from the sugar beet cuttings. Beet cuttings enter into the extractor at the same end where a raw juice outlet is placed and they are moved mechanically by a screw towards the opposite end where extraction water is added and spent beet cuttings are taken out of there. Thanks to the screw and a slanting position of the extractor beet cuttings and extraction water flow in counter currents.

The contents of the extractor is heated by steam. The heating system is divided into several parts to obtain desired temperature course along the whole length of the extractor. Levels are measured in several places to show the degree of the filling inside the extractor. The spin of the screw and the performance of its motor are measured too.

The main objectives of extraction control are:
-- the sugar loss as small as possible,
-- the raw juice output as small as possible,
-- the raw juice purity as high as possible,
-- operation costs as small as possible.

To influence the extraction efficiency the following factors are necessary to be considered:
-- the rate between beet cuttings and added water flows,
-- the temperature course inside the extractor,
-- the degree of filling of the extractor,
-- extraction time
-- beet cuttings quality (properties).

A smooth performance also depends on sugar beet physical and biological properties - which are impossible to be measured. Essential measure and control loops are shown on Fig.3. These loops stabilise only the basic operating values, but they

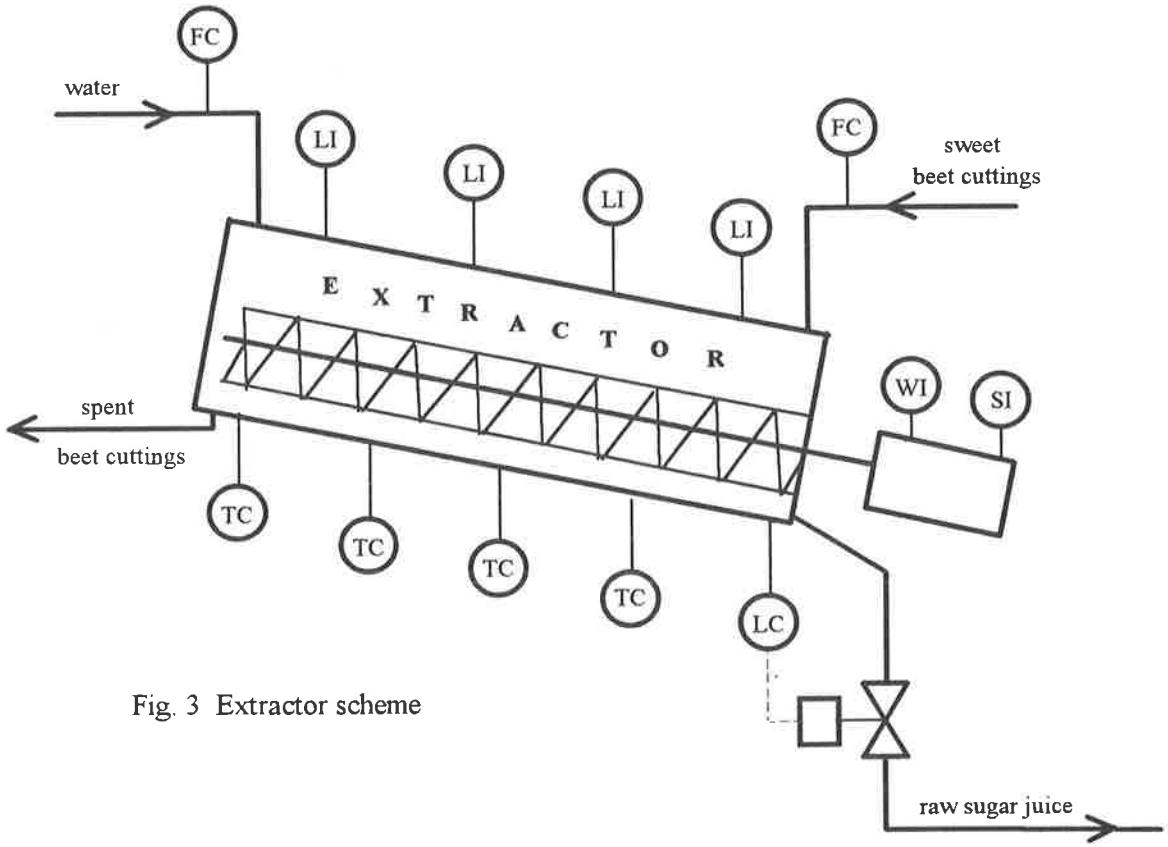


Fig. 3 Extractor scheme

are not able to reach main goals of control mentioned above. The raw juice flow at the lower end of the extractor is controlled by level control loop. According to the flow of fresh beet cuttings clear water is added at the upper end of the extractor.

In connection with the other parts of the production (technological) line and because of changing properties of the sugar beet during the campaign it is necessary to set values of the following control parameters:

-- the performance of the extractor (the amount of processed beet),
-- the transport rate of the cuttings inside the extractor (extraction time),
-- the amount of added water,
-- the temperature course along extractor,
-- the degree of the filling (level of contents of extractor).

The values of these parameters must be set by the operator according to his or her experience. Although the laboratory analyses are at operator's disposal the results come a long time after the samples were taking off.

The operator's experience of the extraction control, his knowledge of raw materials (sugar beet, steam, etc.), technological conditions of the extraction process and another knowledge, it should be included into a knowledge base of the expert system. Such an expert system is able to evaluate the state of the process and make decisions according to measured values and another information given by the operator. The setpoints of control loops are adjusted.

If we simplify the situation on qualitative description only, there are following possibilities of actions:

-- according to the size and the mechanical properties of beet cuttings the suitable course of temperature along the extractor is set; low temperatures make the extraction effect insufficient, high temperatures make problems with beet cuttings transport inside the extractor because of their soften state.

-- according to the requested performance of the extractor the degree of filling of the extractor is changed by modification of screw spin, so that the hydrodynamic condition is optimal,

-- when the sugar content in the spent cuttings is high the rate between fresh water and sweet cuttings flows is increased and the other way round,

-- according to the quality of sugar beet (fresh, frozen, half-rotten, etc.) the suitable temperature course along the extractor is set and the performance and extraction time are changed.

Of course, it is necessary to build up the knowledge base particularly for each type of the extractor. The file of input information for expert system contents the values found by direct measuring, results of laboratory analyses and quantified evaluation of beet's technological quality. The expert system detects the actual state of the extractor. According to this result the chosen system of actions moves the plant into the desirable state.

At the present time we are working with some simplified model experiments so the results of operational experiments are not available yet.

4. CO-ORDINATION CONTROL

Sugar factory in the city of Lovosice is a plant that processes about 1300t of beets daily. It consists of two parts - a continuous part and a discontinuous one.

The structure of its continual part is shown on figure 4. There are three main sections there, namely extraction of beet cuttings, purification of extracted juice and evaporation of thin juice. Three balancing tanks are maintaining an amount of juices in the process line.

The thick juice tank supplies the boiling house that's why there must be a sufficient supply of juice there so that a new brew can be started at anytime. However, the process in the boiling house is not continuous. It is a process that involves irregular intakes, moreover the volume of thick juice intake differs according to the situation inside each of the five vacuum pans.

It is also inevitable to have enough thin juice in the tank leading to the evaporation section. If there were no juice water would have been poured into it instead of juice to avoid destruction of the evaporator.

A huge complication are different time delays between balancing tanks. Especially extraction time delay is some 50 minutes and the capacity of extracted juice tank is very small. Without any supply it can be absolutely empty in 10 or 15 minutes. But the problem is usually opposite. The boiling house is full, the thick juice tank is full, the thin juice tank becomes full and there is no free capacity in the extracted juice tank - the cutting of beets must be stopped.

In this situation an expert system is probably the only one that can help. The goal of its implementation is to make this process more stable and uniform, without undesirable interruptions. It helps to save energy, fuel and steam consumption, saves time and in this way increases the volume of juice flowing through the plant.

About thirty values are to be prepared for

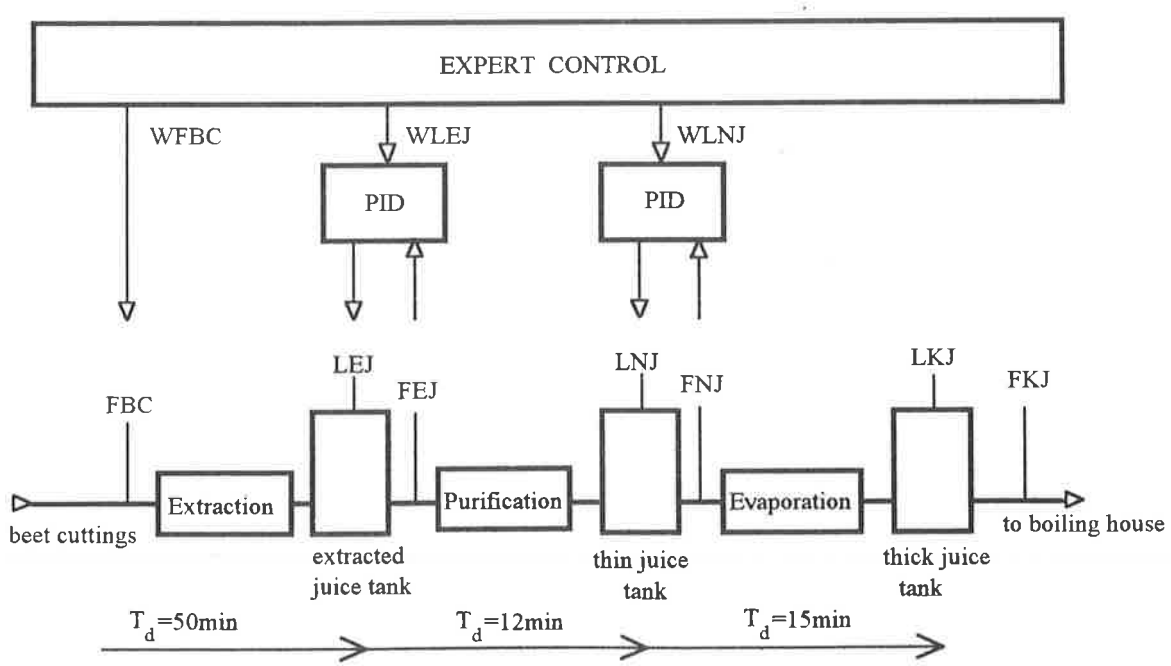


Fig. 4 The structure of the continual part of Lovosice sugar factory

consultation with the expert system. Some of them are measured, some are calculated from them - beet cuttings flow (FBC), extracted, thin and thick juice flows (FEJ, FNJ, FKJ) and levels (LEJ, LNJ, LKJ), their trends are some examples of them.

The knowledge base used by the expert system FEL-EXPERT (Mařík,1991) controls three values - beet cuttings flow, extracted juice flow and thin juice flow. Each of the outputs has three goal hypotheses - to increase, to decrease and not to change the desired value, so that there are nine goal hypotheses altogether. After consultation with the expert system these hypotheses are ordered according to their importance. For each output the hypothesis with the highest importance is chosen as the best one, and it indicates whether desired value of controlled output should be changed or not.

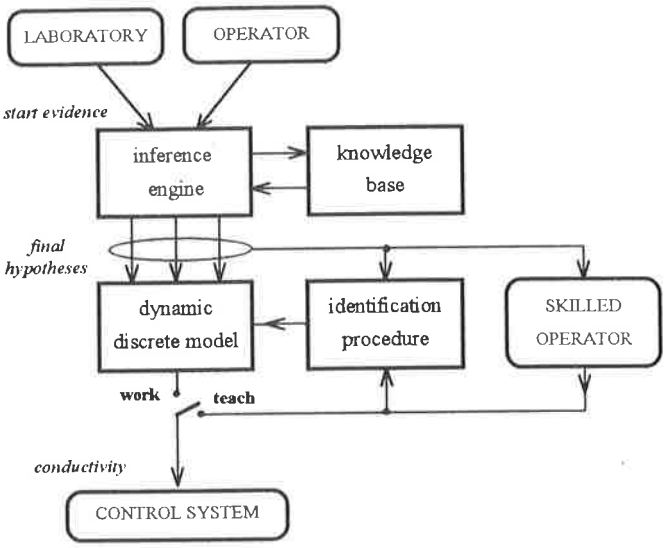


Fig. 5 Expert controller structure

Afterwards the expert control system computes and sets the value of beet cuttings flow (WFBC) and desired values of extracted and thin juice tank levels (WLEJ, WLNJ). These levels are computed according to recommended changes in extracted or thin juice flows and they are regulated by local PID controllers.

The computing algorithm of the expert control system is based on juices and beet cuttings balance calculations and it is considerably complicated and comprehensive both for the part that prepares data for the expert system and for the part that evaluates results of the consultation with it.

5. CRYSTALLISATION CONTROL IN A VACUUM PAN

During the crystallisation process the dissolved sucrose crystallises in a vacuum pan. Control of the process should guarantee the highest possible speed of crystal growth at the lowest energy consumption. Crystallisation in the vacuum pan is a batch process usually controlled by a programmable logic controller with a few analog inputs. One method to control the crystallisation process is based on measuring conductivity. It has been discovered that conductivity is an indication of the extent of supersaturation that forces the crystals to grow. The conductivity base control has several drawbacks. It depends on the impurities and contents of non-sugars that are affected by the soil in which the beet was grown and condition under which the beet is maintained and stored. They vary from region to region and from year to year. There are three key stages during the process: seed, feed and brix-up. The batch process can be satisfactorily controlled only under the condition all conductivity parameters are well adjusted. They are : seed conductivity, four conductivity limits for feed stage and final brix-up conductivity. A lot of skill and experience is required to set these parameters.

The simple control of the crystallisation process in the vacuum pan by means of a logic controller based on conductivity measurement didn't get satisfactory results as there are too many uncertainties, nonlinearities and heuristics. On the other hand a good control of this process would increase the performance of the boiling house and its economical importance is therefore far from any doubt. (Michal, 1993).

The main difficulty in present control is to adjust the characteristic parameters: seed conductivity, lower and upper conductivity limits for feed stage and final brix-up conductivity. This is the task of the expert control. The structure of the expert controller is on Fig.5. The knowledge base has three parts - for seed, for feed and for brix-up. At the end of each stage a consultation is activated to evaluate the process and to change parameters if necessary. The input evidence for the consultation are the quantitative results from the laboratory analysis and the qualitative subjective evaluation of the process made by operator. He answers several questions concerning the amount and the size of seed, the density of juice, the presence of false grain and consistency of the final product. All questions require only quantitative answers. The inference engine checks the rules in the knowledge base and derives the final hypothesis. The result of the consultation is an assertion concerning the state of the process. The assertion has the form of several final hypotheses each connected with a probability value which is a measure of theirs validity. The knowledge base is relatively simple and was created with a help of experts in sugar boiling. The second part of the expert controller is the dynamic discrete model.

$$u(k)=\sum_{j=1}^{n} a_j u(k-j)+\sum_{j=0}^{m}\sum_{i=1}^{r} b_{ij} p_i(k-j) \quad (1)$$

where

a_j, b_{ij} are coefficients of the model,
$p_i(k)$ is the probability of the final hypothesis H_i in discrete time k,
$u(k)$ is the output value in discrete time k.

We have chosen a model where $m=n=2$ and the following strategy is used to find the parameters. At the beginning of the campaign a skilled operator or an expert adjusts the characteristic conductivity according to the suggested hypothesis of the expert system, which evaluates the state of the process. Expert actions are used for on-line identification of parameters of the discrete dynamic model. As soon as the identification is complete, the expert controller can be switched to automatic mode, where the output value is calculated from the final hypotheses using the identified model.

6. PRACTICAL IMPLEMENTATION AND RESULTS

In the first paragraph there was shown that the expert supervisory control is in the third layer of the global control structure. For it's good performance both underlying layers must be reliable and there must be fast and error free communication among the layers. Fig.6 shows the block structure of the technological computer net based on the SattBus communication protocol for linking PLCs, process controllers, intelligent I/O devices and control computers and PCs. All technological data, measured as well as gained from laboratory analysis are available for all nodes in the net. A principle of distributed database is used. Data are stored in the node nearest to its origin and transmitted to another node on request.

The software of the control system is distributed over all nodes. It consists of net software, measuring and manipulation routines, operator's presentation software and expert systems. In first installation a rule-based expert system working with uncertainties was used.

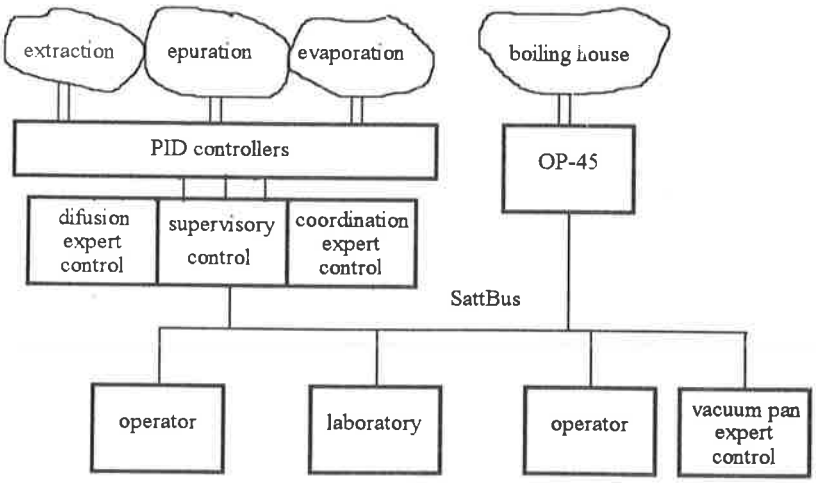


Fig. 6 Block structure of the control system

7. CONCLUSION

The application of AI in control of technological processes is moving from the theory into practice. This paper shows a practical application of an expert control for the technology of sugar production. Co-ordination of the production line, extraction of the sugar from the beet and crystallisation processes are of the type where classical control fails. That is why expert control was suggested and partially implemented in sugar factory in Lovosice. First results indicate this solution is a promising one, with good economical prospects for the future.

8. REFERENCES

Brisk, M.L. (1993), Process Control: Theories and Profits, Proceedings of IFAC Congress, Sydney, Vol.7 pp 241 - 250.

Mařík,V. and T.Vlček (1991). Expert System FEL-EXPERT. Czech Technical University, Faculty of Electrical Engineering, Department of Control Engineering, AI Group, Prague, Czechoslovakia.

Michal,J., M.Kmínek, P.Kmínek (1993), Expert Control of Vacuum Pan Crystallisation; International Conference on Systems, Man and Cybernetics ' 93, Le Touquet France, October 1993, pp 31-35.

Michal,J., D.Burian, P.Kmínek (1992), Educational Aspect of Expert Control of Technological Processes, IFAC International Symposium on Artificial Intelligence in Real-Time Control, Delft The Netherlands, June 1992, pp 319-324.

Verbruggen,H.B., A.J.Krijgsman and P.M.Bruijn (1991). Towards Intelligent Control. *Journal A*, vol.32, no. 1, 35-45.

Verbruggen,H.B. and K.J.Astrom (1989). Artificial Intelligence and Feedback Control. In 2nd IFAC Workshop on Artificial Intelligence in real-time control, China, September 1989,pp. 115-125.

Wilkie,J.D.F. (1990) Batch process control, Computer Control of Real-Time Processes, IEE Control Engineering Series 41, Peter Peregrinus Ltd., London .

A REAL-TIME KNOWLEDGE-BASED BLAST FURNACE SUPERVISION SYSTEM

LEIF KARILAINEN*,‡ and HENRIK SAXÉN*

* *Department of Chemical Engineering, Åbo Akademi, Biskopsg. 8, FIN-20500 Åbo, Finland*

‡ *Fundia, Koverhar Works, FIN-10820 Lappohja, Finland*

Abstract. This paper presents a supervision system for a blast furnace process, with emphasis on the hardware and software solutions. The purpose of the system is to give the blast furnace operator possibilities to simulate important events or changes in the process and to act as an operator aid by monitoring details of the process calculated or inferred by the software modules. Because of the high complexity of this industrial process, the final control actions are still decided by the human operator.

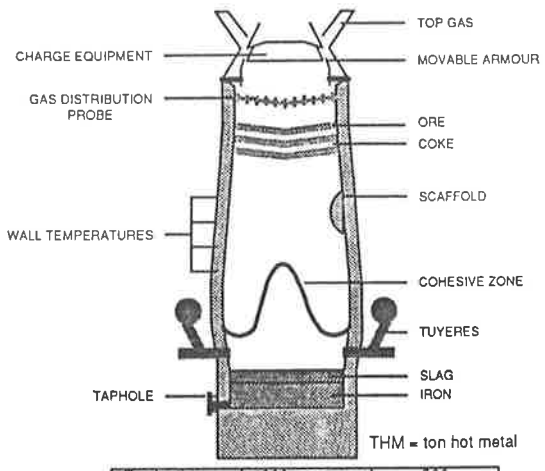
Key Words. Guidance systems; supervisory control; steel industry; real time computer systems; knowlegde engineering; monitoring; software tools

1. INTRODUCTION

The blast furnace (BF) is the most important process unit for production of iron for steel manufacturing in the world. It can be described as a huge chemical reactor and heat exchanger where heat and mass is transferred between solid, liquid and gas phases. The solid iron bearing materials (sinter and/or pellets) and the coke are charged at the top of the furnace. To melt and reduce the iron bearing materials a temperature of about 2000°C is required, which is maintained (locally) by burning the fuel with oxygen enriched preheated air (together with pulverized coal or oil), injected into the furnace through the tuyeres (see Fig. 1). After undergoing several chemical reactions, the ores soften and melt in the cohesive zone and dribble through the lower parts of the furnace into the hearth, where the *pig iron* is collected with the by-product slag, and are tapped irregularly through the taphole(s). The combustion gases, in turn, rise and leave the furnace at the top.

As the conditions in the furnace are very demanding (high temperatures and pressure, considerable mechanical wear, *etc.*) only very few direct internal measurements are available. Thus, it is very important to obtain information about the inputs and outputs, and these variables are therefore measured and analyzed quite carefully in order to provide possibilities to interpret and simulate the internal conditions of the furnace. Because of the long residence time of the condensed phases — typically 7–10 hours — it is vital to detect abnormalities and disturbances in time to be able to take efficient control actions. The gas here acts as a rapid source of information, since it rises through the furnace in 10–20 seconds: Its temperature, chemical analysis and radial distribution reflect *e.g.*, the flow conditions of the ores and the heat level of the furnace. Such information together with other measurements and (qualitative) observations, in combination with several years of process experience yield an estimate of what is happening inside the furnace.

However, the task to follow and recognize patterns in the BF operation is a very challenging and time consuming task, which should be carried out in real-time. Knowledge-based techniques dealing with heuristic process know-how have therefore proven to be helpful as an operator aid. However, the control strategies, the equipment and the external constraints may vary a lot between the iron works. Therefore, the systems developed have often been designed for particular furnaces, using local expertise, experience, and data from specific measurement devices. These are some of the reasons why it was considered feasible to develop an own system for supervision and operator aided control of the blast furnace. In what follows, the general outlines of the system are described.



Production	1000 tons/year	500
Hearth Diameter	m	6.5
Working volume	m^3	567
Fuel Rate	kg/THM	450
Hot Metal Temp.	C	1450
Blast Temp.	C	1050
Slag Volume	kg/THM	220
Sinter Rate	kg/THM	1500

Fig. 1 Schematic illustration and characteristic data for Koverhar blast furnace.

2. THE STRUCTURE OF THE SYSTEM

The rapid development in the field of computer science and technology has made it possible for research and production engineers, who are not specialized on computers, to take part in the programming tasks of complicated applications in process control and supervision. An important issue in the development of such systems is the choice of both hardware and software to be used. This chapter describes the solutions and choices made for the real-time knowledge-based supervision system for the blast furnace of the Koverhar works, Finland.

2.1 Hardware and Network Issues

The old supervision and control system at the furnace in question —- the *Koverhar system* (Moliis-Mellberg and Barnes, 1983) — runs on the process computer (PDP 11/84) where all measurement values are collected. Roughly, the BF data can be classified into two groups: regularly logged and batch type data. At Koverhar there are about 850 of the former type, including calculated quantities, while the remaining 60 tapping and charging variables belong to the latter category. Mean values are calculated using different time "windows" (10 minutes, 1 hour, 1 day) for the most regularly logged data. The data files are of cyclic nature, so data are stored only for a certain period before overwriting starts. The RSX operating system is quite old-fashioned and has severe capacity limitations. Therefore, there have been plans to update both the automation system and the process computer.

In 1993 a computer for the new system for blast furnace supervision and process operator aid (SPO) was installed at the iron plant. Special attention was paid to the user interface to ensure that the system would be easy to use with graphical screens and with mouse support, which require considerable storage and processor capacity. A DEC-System 5000-133 model with 48 Mbs of RAM memory and large disks was procured. This ULTRIX machine was connected to a local ethernet network with PATWORKS software using DECnet protocol to handle the network traffic. The relational database, Oracle (Oracle Corporation, 1989), resides on the ULTRIX server and facilitates quick data search all over the network. Data transferring routines are timed to be executed at correct moments for transferring data from the process computer to the server via an optical fibre connecting the two local ethernet networks. The system further includes a graphical modelling tool, SL-GMS (Sherrill-Lubinski Corporation, 1991), and an expert system shell, Nexpert Object (Neuron Data Inc., 1991), which both run on the server. The application utilizes these commercial software tools' C-language libraries when calculated and inferred results are displayed to the user. Figure 2 illustrates the architecture of the hardware and the network.

Since the development work is carried out in a joint project between the steel industry and a university, most of the development work is made under VMS operating system on a VAX-station 4000 VLC computer with 16 Mbs of RAM memory, connected to a university network. In order to make the use of commercial software tools as easy as possible, it was decided to use the C programming language for the modules. Another important matter was that the code should be easily portable between VMS and UNIX operating systems.

At the iron works, five PCs with 486 processors have been installed in a local network. The SPO application runs with Excursion for Windows — a standard windowing system for networks using X-window protocol — from the server with displays on the PCs. The application's user interface is therefore the same on every machine, independent of the operating system. The network makes it possible to copy and execute files on different disks and to flexibly use the network's services such as printers, tape drivers, *etc.* Moreover, a separate server is advantageous in that a possible download of the supervision system will have no effect on the process computer, which handles the BF control. This is important because it is quite possible, or even likely, that problems occur when new parts of the system are installed.

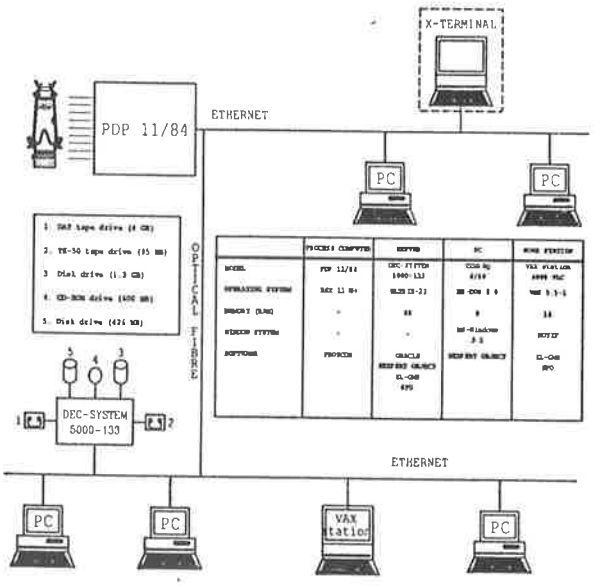


Fig. 2 Hardware and network architecture.

2.2 Commercial Software

Relational databases offer major advantages over sequential data files in that the data are clearly grouped into tables and columns. With an SQL query language it is possible to retrieve data from the Oracle relational database, with conditions, such as time restrictions or simple calculations (addition, subtraction, multiplication, division, mean values, variances, *etc.*) specified. Relational databases are, moreover, easier to edit, maintain and alter: For instance, in a situation where a new measurement is to be defined and stored in the database, the SQL language can be easily used to add one column to an existing table, and new measurement values can be inserted without any further changes or side-effects on the system. Today, most software toolkits are equipped with interfaces to commercial database programs, or at least support facilities (open architecture) for the user to write such *bridges*. In this work, PRO*C for data retrieval from Oracle in common C was used, which precompiles every Oracle command to C code. The relational database in question also supports PRO-language facilities for other programming languages (fortran, ada, pascal, cobol, PL/1, *etc*).

In process supervision and control applications, the importance of the man-machine interface must be stressed. In order to develop a user friendly interface, a graphical modelling system (SL-GMS) was used (*cf.* Fig. 3). Displays are created with a drawing editor (SL-DRAW) by drawing graphical objects and saving them in ordinary portable ASCII text files①. The object dynamics on the display are driven by data from the application or by user inputs② (keyboard and mouse) through a data definition table③. The programmer has full control of keyboard and mouse actions all the time, and external subroutines can also be called at any moment. The dynamics support more than 50 different actions, such as color, text, visibility, filling grade, $x-y$ movements, scaling, *etc.*, for the objects. With these facilities it is possible to create moveable, real-time process displays updated by a timer routine④ to the end-user⑤. SL-GMS uses C-language program libraries, which can be called from the users application. Some features of object-oriented programming have also been used in the libraries. Different methods can thus be defined at the class level, and be inherited by the members of the class. It is therefore very easy to create a "standard" for how user input and object dynamics are treated on the displays.

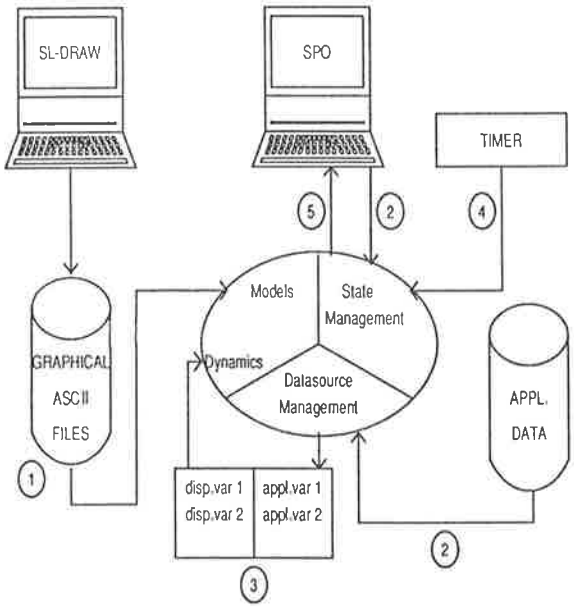


Fig. 3 The SL-GMS graphical modelling system.

When complex practical problems are tackled, heuristic knowledge, which can be difficult to represent mathematically, is often available. In such cases, it may be advantageous to use knowledge-based models for tackling part of the problems. Based on the work in a previous project (Gyllenram *et al.*, 1991), it was decided to develop the knowledge-based models of SPO using *Nexpert Object*, an expert system tool where knowledge can be represented by rules① and frames② (see Fig. 4). In this tool, the knowledge base (KB) is an ASCII text file③, which is portable between different environments (machines and operating systems) and can be executed with a run-time version of Nexpert Object. The graphical interface of the tool is easy to use, and only basic knowledge of expert systems is required to create, edit and maintain the KBs. Reasoning can be made both backwards and forwards and the user can give the rules priority indices. The tool has also an Oracle bridge④, which makes it possible to retrieve data directly from the KB. Like SL-GMS, Nexpert Object also consists of C-language library functions, a so called *application programming interface* (API)⑤ and includes some features of object oriented programming. External subroutines can be called both from the premiss (IF side) and action (THEN side) of the rules, from the objects and naturally also from the user's subprograms.

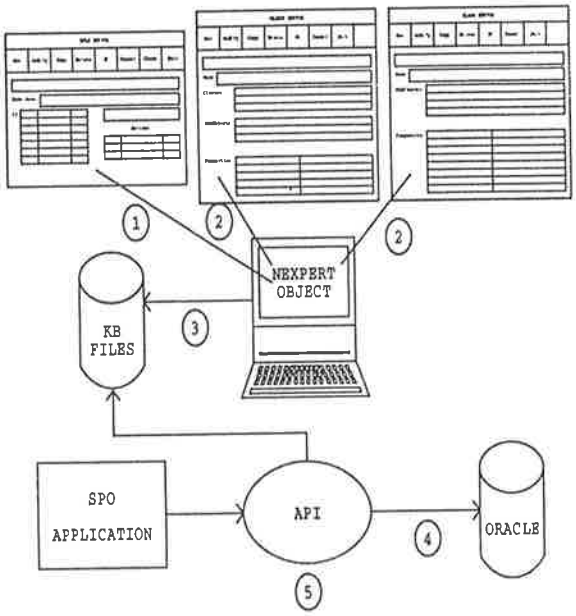


Fig. 4 The Nexpert Object expert system tool.

3. THE APPLICATION

With experiences from a Nordic joint research project (Gyllenram *et al.*, 1991), it was decided in 1990 at the Koverhar works to develop a new knowledge-based supervision system for the blast furnace. With the hardware and software solutions described above, the development started 1991 and still continues. In this chapter, some of the building blocks of the software will be described.

Since the application runs in real time, the programs were written to be executed after receiving a signal that all necessary data have been logged and copied to the Oracle database on the server machine ① (see Fig. 5). The application consists of modules which deal with some of the most important issues and phenomena in the control of the blast furnace, such as iron-slag partition of chemical components (screen display in Fig. 6), raceway simulation, charging control (Hiden, 1994), interpretation of the gas distribution (Saxén and Karilainen, 1990), reconciliation of measurements (Forss, 1992), supervision and analysis of heat losses, heat level prediction, prediction of slips and hangings, tapping control, cohesive zone estimation and hot stove control. The application programs are divided into two main groups, as illustrated in Fig. 5: ② programs that do not include graphics and are executed regularly or triggered by events (*e.g.*, tappings), or ③ routines with graphics. This partition was introduced to

make it possible to rapidly display graphics in spite of the fact that some of the routines, *e.g.*, the data reconciliation algorithms, require a considerable computational effort.

Some of the above mentioned modules were converted from the old automation system, while other were developed specifically for the new application. Work is still in progress to incorporate more modules; the reader is referred to reference (Karilainen *et al.*, 1994) for more information about the submodels. For the purpose of illustration, the next subsection presents in more detail the heat loss module of the system, which illustrates some of the features of knowledge-based reasoning and object-oriented programming used in this work.

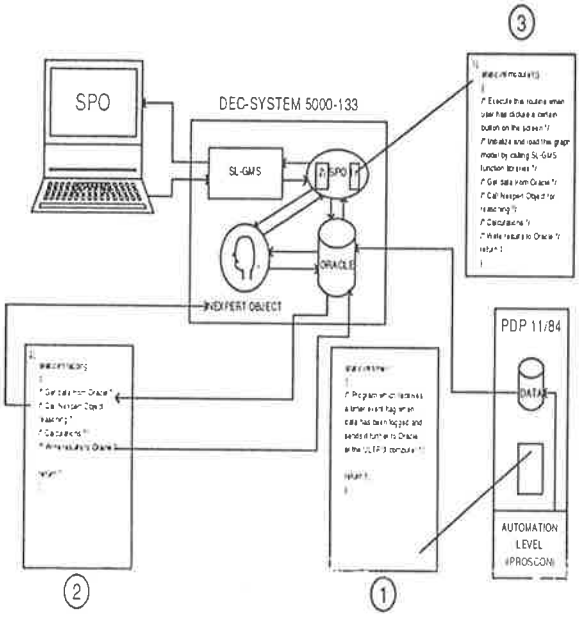


Fig. 5 Structure of the SPO application.

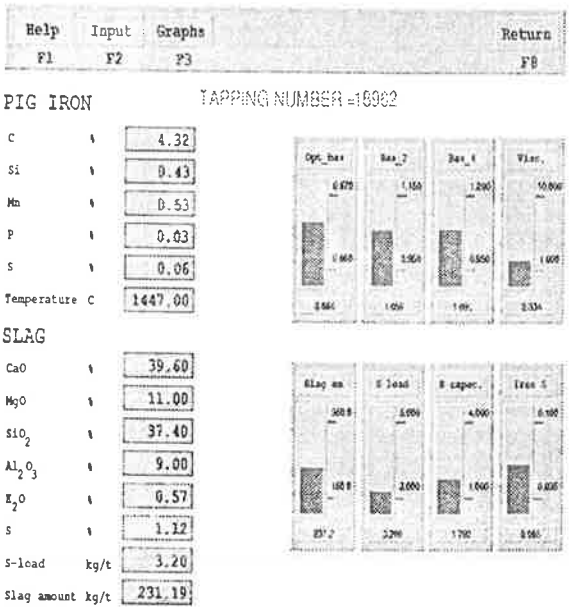


Fig. 6 A display from the iron-slag partition module.

3.1 The heat load check; an example of a software module

The BF walls must be continuously cooled with water to stand the enormous heat load caused by the high temperature in the furnace. The heat loss is usually controlled to maintain the furnace lining. Moreover, the load can give information about formation or melting of accretions (condensed material clinging on the furnace wall), which may affect and cause abnormalities in the descent of the charged materials and often result in poor pig iron quality. At the Koverhar works the cooling system is divided into four sectors around the periphery of the furnace and into four vertical levels (bosh, belly, lower shaft and upper shaft). The heat loss for each of these sixteen above mentioned regions is calculated on the basis of information about the temperatures and the flow rate of cooling water.

The rule displayed in the rule editor of Fig. 7 uses a heat loss index calculated for the losses in the different regions. A low index is desired; indices above 16 usually indicate abnormalities in the process. The object oriented and frame facilities of the expert system tool have been used in the IF part of the rule presented. The *Sector* class, visualized in the class editor, has four subclasses (*Sector_1* to 4), six inheritable properties (*index*, *index_high*, *index_high_duration*, *level*, *measurement_value*, *neighbour*) and individual values for the properties (I=integer, B=boolean, S=string and F=float). When a new object, say *Bosh_sector_4*, is to be created, all that has to be stated its name and that it belongs to the *Sector_4* subclass. The object automatically inherits the property slots from the subclass, which makes the development of the KB fast and easy, and furthermore reduces the risk of typo errors.

The classes have also another important feature which can be used in the rules. In Fig. 7, the object $<|$ *Sector* $|>$ *.index* in the rule denotes a pattern matching operation on the *index* slot of members of the *Sector* class. This generic operation creates a list, which includes all the objects with a heat level ≥ 16. This feature serves to reduce dramatically the number of the rules required, and makes the KB more transparent. If the left hand side of the rule is true, the rule fires and the (Boolean) hypothesis *index_high* will be true. Simultaneously, the right hand side actions are executed: First, information on how many consecutive discrete time steps the index has been high is retrieved from Oracle, next the *index_high* properties for the members in the pattern matching list are set to true and their *index_high_duration* property is incremented by one. The alarm is activated if a sector shows a high heat load for at least three consecutive time steps, or if two neighboring sectors are high simultaneously. The alarms are written into the database, and can now be accessed by the heat loss module, which displays the results graphically on the screen.

The use of the tool for knowledge-based reasoning makes maintenance of the system flexible and easy, in that neither compilation nor linking of the application code is required if the KB has to be modified. The changes can be introduced interactively using the development version of the tool, and immediately tested and verified.

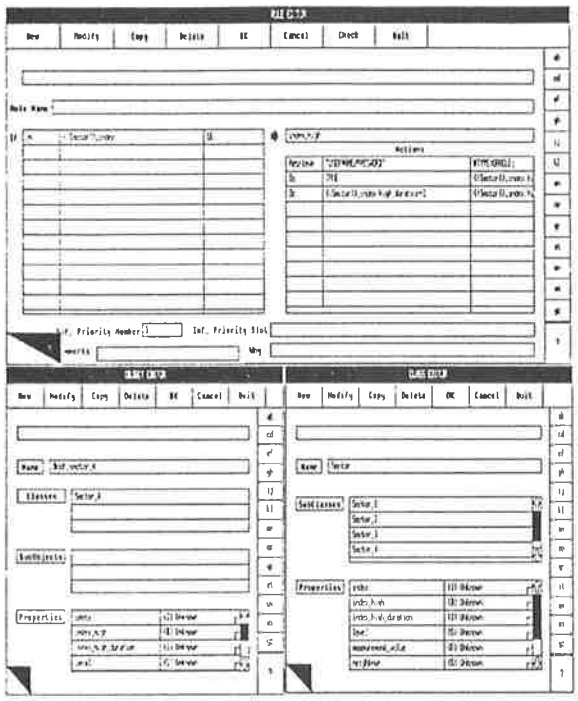


Fig. 7 Rule, class and object editors in Nexpert Object.

4. CONCLUDING REMARKS

This paper has described the hardware and software used in a system for blast furnace supervision and operator aid. The system has been developed incrementally, and new modules are created and transferred after testing to the development department at the iron works. Even though the system is still under work, the architecture has been found appropriate and efficient, and well suited for incremental development. By this approach the operators and plant engineers get acquinted with the new system gradually. Another important matter is that feedback and criticism can be considered in time before the system reaches its final state. Because of the four different operating systems used in work, some problems with file transfers and program linking were experienced initially. Part of these problems arouse because several commercial programs had to be linked together in the system. However, C for software programming proved to be a proper choice, since most of the software tools were written in this language. In general, these tools were found to be diversified and well portable between different environments, but at the same time quite demanding. In summary, the work has shown that it is feasible for engineers, who are not specialized on computer technology, to design and install own applications, such as systems for process supervision and control.

5. REFERENCES

Forss, K. (1992). Detection and elimination of errors in blast furnace data (in Swedish), MSc Thesis. Åbo Akademi, Åbo, Finland.

Gyllenram, R., J. Sandberg, H. Saxén and L. Karilainen (1991). A Prototype Knowledge-Based System for Blast Furnace Operational Guidance, *Ironmaking Conference Proceedings*, 50, pp. 201–219. Iron and Steel Society, Warrendale PA.

Hidén, A. (1994). MSc Thesis (in preparation). Åbo Akademi, Åbo, Finland.

Karilainen, L., H. Saxén, K. Raipala and J. Hinnelä (1994). An Interactive System for Real-Time Operator Aided Control of Blast Furnace, to appear in the Proceedings of the *53rd Ironmaking Conference*. Iron and Steel Society, Warrendale PA.

Moliis-Mellberg, A. and T. Barnes (1983). Ironmaking at Koverhar 1974-1982. *Iron & Steelmaker*, pp. 24–28.

Nexpert Object Reference Manual (Version 2.0) (1991). Neuron Data Inc., Palo Alto, USA.

ORACLE (Version 6.0) (1989). Oracle Corporation, USA.

Saxén, H. and L. Karilainen (1990). A Prototype Expert System for Diagnosis of Blast Furnace Conditions, Proceedings of the *IASTED International Symposium on Artificial Intelligence Applications and Neural Networks*, pp. 263–266. Acta Press, Anaheim.

SL-GMS Reference Manual (Version 4.0) (1991). Sherrill-Lubinski Corporation, USA.

REAL TIME SUPERVISION OF WASTEWATER TREATMENT PLANTS: A DISTRIBUTED AI APPROACH

M. SÀNCHEZ*, I. R-RODA**, J. LAFUENTE**, U. CORTÉS*, M. POCH**

*Universitat Politècnica de Catalunya. Dept. of Software. Pau Gargallo 5. 08028 Barcelona.
**Universitat Autònoma de Barcelona. Chemical Engineering Unit. 08193 Bellaterra. Spain.

Abstract. Conventional automatic control systems applied to wastewater treatment plants deals with some difficulties: complexity of the system, an ill-structured domain, qualitative information, uncertainty or approximate knowledge, real-time dynamic system, etc. Also, the knowledge-based approaches developed from artificial intelligence field show other disadvantages: brittleness, knowledge acquisition problem, do not learn from experience, the increasing complexity of the systems, etc. The main goal of the paper is to present a knowledge-based distributed architecture for real time supervision and control of wastewater treatment plants that overcomes some of these troubles. It is discussed the development of the application and the methodology employed in it. The prototype's architecture being developed DAI-DEPUR is detailed together with some obtained results.

Key Words. Real-time Supervision and control, Distributed AI, Knowledge-based Systems, Wastewater treatment, Knowledge Acquisition, Environmental Engineering, Biotechnology.

1. INTRODUCTION

The main goal of a wastewater treatment plant is to reduce the level of pollution of the wastewater at the lowest cost, that is, to remove -within possible measure- strange compounds (pollutants) of the inflow water to the plant prior to discharge to the environment. So, the effluent water has low levels of the pollutants (lower than maximum ones allowed by the law).

A wastewater treatment plant is composed usually by two stand-alone but interactive subsystems: *sludge line* and *water line*. The study has focused on the water line. It is formed by a sequence of unitary processes where the effluent of one process becomes the inflow to the next one. Usually there are 3 major processes: primary treatment, secondary treatment and tertiary treatment. Each one reduces or removes the concentration of several specific pollutants (Metcalf 1991). The chart of a plant may be seen in Fig. 1.

The plants taken as models -in this study- are based on the main biological technology usually applied: *the activated sludge process* (Robusté 1990). The actual wastewater plant studied is located in Manresa, near Barcelona (Catalonia). This plant receives about 30000 m^3/day inflow from 75000 inhabitants.

The setting of an automatic process control over the wastewater treatment plant system, has outlined some difficulties:

- The *complexity of the problem*. There are many facts from different nature present at the system: chemical facts, mechanical, biological, microbiological ones, ...
- A *non-well structured domain*. The relationships among the concepts or attributes of the domain are not enough known or the agreement among experts is not very high.
- *Most information is neither numeric nor quantified*. So, it is not suitable to be included in the context of a classical -numerical- control model (microbiological information, etc).
- *Uncertainty or approximate knowledge*. By example, the subjective information supplied by the experience of the plants' managers.
- A *dynamic system*. Continuous changes that modify directly the performance of the process.

Moreover, it can be said that the *classical methods* work well on the *normal* states of the plant but not in other *abnormal* states of working. How could they detect some mechanical faults?. How could they use the available information but incomplete one, to solve a specific problem?. These troubles with the conventional process control systems as *feed-back control*, *feed-forward control*, *optimal control*,

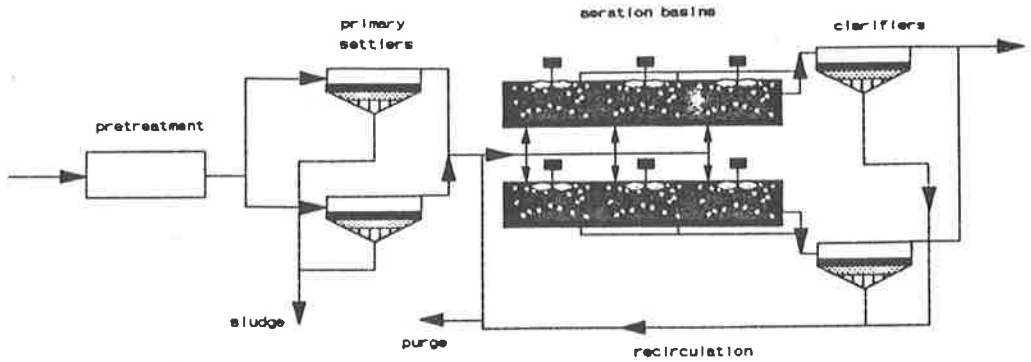


Fig. 1. Chart of a wastewater treatment plant.

adaptive control have aroused during last years much research effort in Artificial Intelligence (Coleman *et al.* 1991, Maeda 1989).

The main characteristics of Knowledge-Based Systems (KBS) point out that they could be used for the supervision and control of wastewater treatments plants:

- *Useless in concrete domains*: they are effective when applied to a certain domain where some expert people can afford their experience and knowledge.
- Supporting of *numerical and/or symbolic information.*
- Specially useful in *ill-structured domains.*
- Could be extended in some ways. As for example, the treatment of *uncertain or approximate reasoning.*

Nevertheless, KBS do not incorporate some desired features from human intelligence and have some technical difficulties in their developing (Steels 1990).

- Most KBS *do not learn from its experiences.* This, is a valuable feature to be contemplated in KBS.
- *The knowledge acquisition problem.* There are some difficulties in the process of extracting the knowledge and experience from knowledge's sources.
- *Brittleness.* Their scope is limited to the forecasted situations in the domain. They could not response wright in front of unexpected situations.
- *The increasing complexity of the systems.* As the systems grow, is more difficult to manage information and knowledge involved in them.

So, the goal of the designed architecture will be to overcome some of these troubles, in order to build a more efficient and accurate system for the supervision and real time control of wastewater treatment plants.

2. DISTRIBUTED AI AND REAL TIME PROCESS CONTROL

Real time AI systems often require several of these characteristics for real time control process (Sriram 1992):

- High performance
- Continued operation
- Response time warranted
- Uncertainty or missing values in input data
- Supporting asynchronic events and interruptions

These features could be present in a distributed AI architecture. The main reasons for distributing an AI system in a multiagent architecture could be enumerated as follows (Huhns 1987, Castillo *et al.* 1991):

- *Geographic distribution* in the domain of application (air traffic control, information systems, robots' system, etc).
- *Functional decomposition* in a natural way (top-down analysis) such in a medical diagnosis, speech recognition system, etc
- Control distribution in order to get *faster processing speed by means of parallel (concurrent) execution* of agents' work.
- Distribution affords *modularity*, and therefore *reusability* and *extendibility* of the system.
- Processing distribution is a basic strategy to *control the increasing complexity of AI systems.*
- Integration and cooperation of several intelligent agents *(systems) increases the power of the resulting system.* Therefore, more complex problems can be resolved.

Distributed AI has been applied to some areas: air traffic control (Cammarata *et al.* 1983), diagnostic and control (Roda *et al.* 1990), medical diagnosis (Gómez *et al.* 1981).

There are four main types of distributed architectures (Kirn *et al.* 1992): blackboard systems (BBS), supervisory systems (SVS), contract nets (CN), Non explicit coordinated systems (NECS). Supervisory systems has been selected as the most promising approach to this domain attending his characteristics.

3. DESIGN OF THE DISTRIBUTED SUPERVISORY SYSTEM'S ARCHITECTURE

DAI-DEPUR's architecture is the result of previously developed approaches for wastewater treatment plants: a knowledge-based diagnostic and management tool (Serra *et al.* 1994) and a real time supervisory system (Poch *et al.* 1993).

The architecture is designed to overcome the troubles of KBS and the conventional control systems too, as explicated above. The distributed supervisory system (as shown in Fig. 2) is formed of several interacting subsystems (agents) that can be executed in parallel processing: Supervisory-KBS, Water line-KBS, Numerical Control module, Case-based learning module, Acquisition module, Interface module, Explanation module, Actuator system, Data Base Management System and Water line Data Collecting system.

The Supervisory-KBS is the manager of the distributed system and acts as a master. It receives the diagnosis information from Water line-KBS and similar cases retrieved from Case-based learning module. If the diagnosed working situation of the plant is normal, then the numerical Control Module is activated. Otherwise, the Supervisory-KBS notifies the actual situation to the operator of the plant, suggesting or actuating directly to the system.

The Water line-KBS is a knowledge-based system to diagnose the state of the water line subsystem. It recalls data information from the evolutionary real time data base as described in (Sànchez *et al.* 1992, Sànchez, 1991).

The Numerical Control module implements -explained in (Serra *et al.* 1993, Moreno *et al.* 1992)- a dissolved oxygen (DO) control scheme based on four main blocks: the mathematical model of the process; the software sensor to estimate the oxygen uptake rate; a continuous-range optimization procedure and an algorithm that using the continuous-range optimal control value computed by the previous block, generates a discrete-range suboptimal control value, suited to be applied by the aeration motors.

Case-based reasoning and learning module manages a Case Base. This Case Base contains information about previously detected situations and the solutions given to them as well as their efficiency. A Case-based reasoning will be implemented in order to benefit from these past experiences and cases. The Case Base will be modified accordingly with the new information (Aamodt *et al.* 1994).

The Acquisition module is based in recent developments in knowledge acquisition. This module uses the software LINNEO+ (Béjar *et al.* 1992) for the automatic generation of inference rules as the

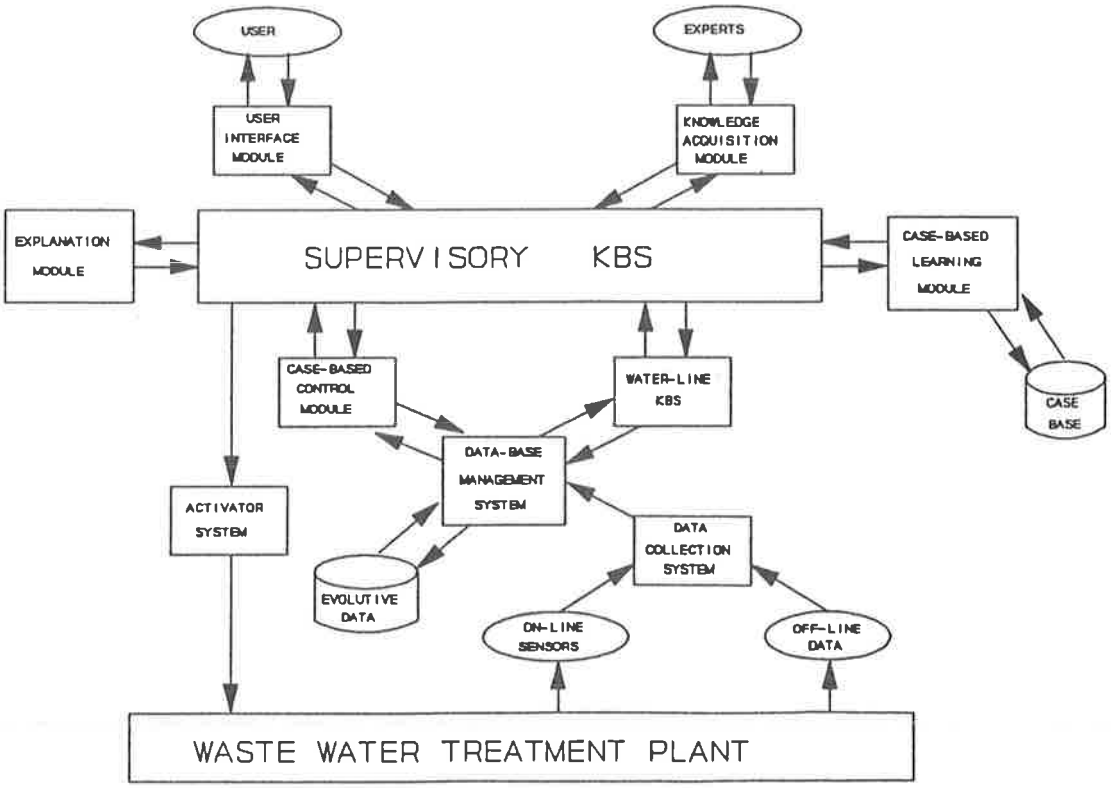


Fig. 2. A real-time distributed supervisory system's architecture.

result of a classification process of experts' defined attributes and observations.

The Explanation module gives some explanations about the reached conclusions of the different KBS of the system (water line–KBS, supervisory–KBS, etc) and could give some required reports about the deductive processes.

Data Base management system controls the access of the several KBS, the CBRL module, the supervisory-KBS to the evolutionary (real-time) Data Base to warrant the consistency and reliability of the system.

Water line Data Collecting system periodically receives data from the on-line sensors of the plant and sends them to the Data Base through the Data Base management system. The *Interface module* provides interaction between the operator and the system through the visualization of a chart of the plant, asking and answering to the system inquiries, evolution of the system, detected situation of working, integration of process' data obtained off-line in the laboratory, etc. The *Actuator system* lets the system to modify the on-line working parameters of the plant (turbines on/off, recirculating flow, etc) with the supervision of the operator. The scheme of the implementation in Manresa's waste water treatment plant of last three subsystems is shown in Fig. 3.

Status of turbines, pumps, automatic grids, etc. are taken from the control panel with 6 PLC's (SISTEL 8512). The value of these 96 digital signals are transmitted through RS-422 to the computer for monitoring and control. It also receives 9 analogical signals (inflow, wasting flow, recirculationflow, biogas produced, DO-line-1, DO-line-2, pH, temperature-digester-1 and temperature-digester-2) converted by an AD/DA card. This computer has in real time the information of the plant, and is able to plot the evolution of these parameters for the last hour, day or week. It accumulates the running time for all the motors of the plant; activates different kind of alarms when something wrong is detected and switchs the turbines of the bioreactors. In a higher level, it is connected through an ethernet link to main computer (SUN Sparc station) where are running the other subsystems (Knowledge Based systems over G2 shell,etc.).

4. THE SUPERVISORY CYCLE

The system activates a new supervisory cycle at fixed intervals of time. Each cycle is formed of 4 phases: diagnose or evaluation phase, learning and reasoning phase, supervisory and communication phase and actuation phase.

Diagnose or evaluation phase: In a new cycle the Supervisory–KBS activates the water line–KBS to diagnose the state of the different subsystems of the plant (*i.e.*, microbiological identification). This means concurrent processing of all KBS involved.
For this purpose it is necessary to know some values for certain variables of the process. All this data can be extracted from the evolutionary Data Base, fitted either with the on-line sensors values coming from the data collecting systems or with some other features provided by the operator (like a laboratory analysis, qualitative observation, etc).

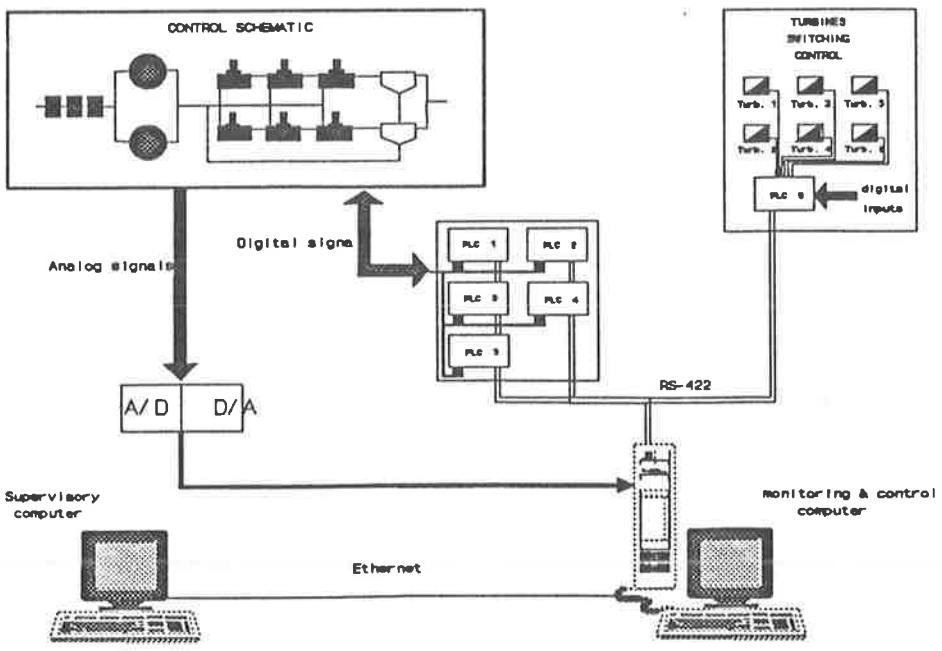


Fig. 3. Real time data collecting

Learning and reasoning phase: At the same time of the diagnose phase –in concurrent execution– the Case-Based Reasoning and Learning module (CBRL) is activated to retrieve similar cases recorded in the Case Base. Next, is updated the most similar one in order to adapt it to actual situation of the plant. For this task, needs to access to Data Base. The results are communicated to the Supervisory–KBS.

Supervisory and communication phase: The Supervisory–KBS combine all information from the several KBS and from the CBRL module to infer the actual global situation of the plant and the suggested actions to be taken. It sends this information to the operator through the User Interface module. The system can be inquired in some ways as asking for explanations, retrieving certain values, etc.

Actuation phase: If the normal situation has been detected then the automatic control is maintained. In another situations, The Supervisory–KBS waits for the operator's validation of actions to be taken to update the actual working state of the plant. If there are on-line actuators, the plant can be updated automatically through the Actuator system. If not, manual operation is required.

5. A CASE STUDY

A case study is now reported to show the real time interaction among the plant, the system and the operator. Situations considered by the system are: normal, rotation-band-crash, summer, winter, by-pass-bad-closed, toxic-loading, bulking, storm, rising, overloading, turbines-crash, overaeration, foaming, etc. and it has been chosen one of them.
The Diagnose or evaluation phase starts, and the value of Volatile Suspended Solids (VSS) at the effluent is high (>35 mg/l) nevertheless the quality of the inflow is normal. The system searchs the possible causes consulting the values recorded in the Data Base obtaining: Water temperature is high (18 °C), DO in bioreactor is lightly high (2.5 ppm) and the age of the sludge SRT is high (8.4 days). At this step, the Kbs is not able to identify the situation, and requires more information from the evolutive Data base: ammonia at the input of the plant is really high (52 mg/l) and ammonia at the output is low (9.6 mg/l). Then the KBS concludes that probably the situation could be nitrification (conversion from ammonia to nitrate in aerobic bioreactors) followed by denitrification (conversion from nitrate to nitrite in anoxic conditions as in clarifiers), and asks to the operator for nitrates values, that are measured off-line in laboratory.

At the same time, the system starts the second phase looking for most similar cases in CBRL. In this example it finds that a similar situation happened in the summer of 1991, when the fact of a big entrance of ammonia to the plant –wasted from a local industry– combined with high SRT and high temperatures, caused nitrogen bubbles in secondary clarifiers. This problem, known as *rising*, provokes a bad sedimentation of biomass with an increasing value of the VSS at the output.

With all this information and as a validation of the suspected working situation, the system consults the operator to know some features as the presence of bubbles when the V30 is done.

The conclusion provided by the Water Line KBS is: it is very possible that the situation was *rising*. Proposed actions: the first one is to increase the recirculation flow in order to reduce the amount of sludge in the clarifiers, and the second one is to increase the wasting flow to have a lower SRT in order to wash-out the nitrifiers of the plant.

6. CONCLUSIONS AND FUTURE WORK

A real time supervision AI architecture for wastewater treatment plants has been designed and is currently being developed. Main outstanding characteristics are real time and distribution: first feature provides an effective control system for a real process (wastewater treatment plants) and the second one provides concurrent execution.

This approach has several advantages that make it more powerful than other technologies applied to wastewater treatment plants: more efficiency through concurrent computation; modularity, reusability and extendibility (easy to export to any wastewater treatment plant with minor changes); learning from experience; a knowledge acquisition module; getting experience from different people with different backgrounds (Microbiologist, Chemical Eng., Expert operators, Control Eng., Computer Science AI researchers); quantitative and qualitative information, both used to real time plant supervision.

By the other hand, design and implementation of the system becomes more difficult and complex. In addition, a common sense (model-based) reasoning sometimes could be missed when neither the Supervisory–KBS nor the CBRL module are able to suggest any solution to an actual working situation of the plant.

With this system, more detailed knowledge about working situations has been obtained (e.g. section 5). Also the integration of signal treatment with symbolic knowledge results with an increasing performance of this wastewater treatment system.

As future work is needed to build the Sludge line-KBS and another modules of the architecture. It will be interesting to study the possibility of integrating a common sense reasoning module to the architecure. Some other new direction points to consider that Knowledge Bases would not be statical in future but dynamical ones. While CBRL is updating the Case Base to adapt it to new cases, it is not silly to consider that expert or control rules (Knowledge Bases) could be adapted accordingly.

ACKNOWLEDGEMENTS

This work has been supported by the "Junta de Sanejament de la Generalitat de Catalunya" and the Spanish CICyT projects ROB91-1139 and TIC91-1087-C03-01. Also the authors wish to acknowledge the cooperation of Ricard Tomàs, manager of the Manresa's wastewater treatment plant.

REFERENCES

Aamodt, A., and E. Plaza (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. *Research report IIIA 94-04*. Centre d'Estudis Avançats de Blanes (CEAB, CSIC).

Béjar, J., and U. Cortés (1992). LINNEO+: Herramienta para la adquisición de conocimiento y generación de reglas de clasificación en dominios poco estructurados.*IBERAMIA-92*, pp. 471-481. La Habana, Cuba.

Cammarata, S., D. McArthur, and R. Steeb (1983). Strategies of Cooperation in Distributed problem Solving. *Procc. 8th Int. Joint Conference on Artificial Intelligence*, 8-12 August Karlsruhe, West Germany, pp. 279-284.

Castillo, L., and G. Quintanilla (1991). Initial concepts in Distributed Artificial Intelligence. *Jornadas técnicas, Hacia los sistemas no convencionales. TEC-COMP.*

Coleman, J.J., S. Krovvidy, R. Scott Summers, and W.G. Wee (1991). An AI Approach for Wastewater treatment systems. *Journal of Applied Intelligence* vol. I, num. 3, pp. 247-261. Kluwer Academic Publishers.

Gómez, F., and B. Chandrasekaran (1981). Knowledge Organization and Distribution for Medical Diagnosis. In *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-11(1) pp. 34-42.

Huhns, M.N. (editor) (1987). *Distributed Artificial Intelligence*. Pitman Publishing/Morgan Kaufmann Publishers, San Mateo,CA.

Kirn, S., and J. Schneider (1992). STRICT: Selecting The "RIght" architeCTure. *Proc. of IEA/AIE-92*. Springer-Verlag. *Lecture Notes in Artificial Intelligence 604*, pp. 391-400.

Maeda, K. (1989). A knowledge-based system for the wastewater treatment plant. *Future Generation Computer Systems* 5, pp. 29-32. North Holland.

Metcalf & Eddy Inc. (1991). *Wastewater engineering: treatment/disposal/reuse*. Mc Graw-Hill, 3th edition.

Moreno, R., C. de Prada, J. Lafuente, M. Poch, and G. Montague (1992). Non-linear predictive control of dissolved oxygen in the activated sludge process. *ICCAFT 5 / IFAC-BIO 2 Conference*. Keystone (CO), USA.

Poch, M., J. Lafuente, P. Serra, U. Cortés, and M. Sànchez (1993). ISCWAP: A knowledge-based system for supervising activated sludge processes. Submitted to *Computers & Chemical Engineering*.

Robusté, J. (1990). Modelització i identificació del procés de fangs activats. Ph. D. Thesis. Departament de Química. Universitat Autònoma de Barcelona.

Roda, C., N. Jennings, and E.H. Mamdani (1990). A Cooperation Framework for Industrial Process Control. In *Procc. of the Int. Conf. on Cooperating Knowledge Base Systems*, Keele, England, pp. 54-57.

Sànchez, M., P. Serra, Ll. Belanche, and U. Cortés (1992). A knowledge-based system for the diagnosis of waste-water treatment plants. *Proc. ofIEA/AIE-92*. Springer-Verlag. *Lecture Notes in Artificial Intelligence* 604, pp. 324-336.

Serra, P., M. Sànchez, J. Lafuente, U.Cortés, and M. Poch (1994). DEPUR: a knowledge based tool for wastewater treatment plants. *Journal of Engineering Applications of Artificial Intelligence* 7 (1), pp. 23-30. Pergamon Press.

Sànchez, M. (1991). DEPUR: Aplicació dels sistemes basats en el coneixement al diagnòstic en plantes de tractament d'aigües residuals. Master Thesis. Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya.

Serra, P., J. Lafuente, R. Moreno, C. de Prada, and M. Poch (1993). Development of a real-time expert system for wastewater treatment plants control. *Control Eng. Practice* 1 (2) pp. 329-335. Pergamon Press.

Sriram, D. (1992). Real time AI systems. In *Inteligencia Artificial y control en tiempo real*, pp. 57-96. Instituto de Ingeniería del Conocimiento-Repsol S.A.

Steels, L. (1990). Components of expertise. *AI Mag.* 11 (2), pp.28-49.

A HYBRID EXPERT SYSTEM FOR VOLT/VAR CONTROL IN POWER SYSTEMS

F. PIGLIONE

Dipartimento di Ingegneria Elettrica Industriale, Politecnico di Torino, I-1029 Torino, Italy

Abstract. The paper presents a hybrid expert system, based on the joint use of a best-first search strategy and a neural network model of the power system voltage/reactive power relationship, which assists the system operator in the voltage profile control task. A Prolog program, based on the proposed strategy, has been developed. Many tests carried out on standard systems have shown promising results.

Key Words. Power system control, Expert systems, Neural nets, Voltage control, Prolog

1. INTRODUCTION

The voltage/reactive power (V/Q) control problem, once considered as a secondary issue which could be solved on a local basis using locally available compensation devices, is now a large scale problem. Indeed, the continuous increase of the load demand has brought transmission systems to operate near to their own security limits. Moreover, the growing complexity and interconnection of power networks has made the traditional local voltage regulation a very difficult task. The problem is not only concerning the planning studies, where full algorithmic methods are available, but particularly the on-line system control, when the control centre's operator must quickly face voltage limit violations caused by load deviations from their program values.

When a bus voltage exceeds its functional limits, the operator undertakes a compensation manoeuvre acting on the available voltage controllers (voltage regulated buses, shunt capacitor and reactor banks, load tap changing transformers, static VAR compensators). Of course, the selected control devices should have enough regulation capability within its own functional limits. Moreover new violations in other buses should not arise in consequence of the compensation action undertaken. Therefore voltage regulation task concerns the whole power system, but it still presents local peculiarities, because only the electrically nearest controllers can significantly influence the regulation process.

The problem approach is traditionally heuristic: the operator employs his own physical problem and specifical network knowledge to eliminate the detected violations, acting on the existing control devices. Control strategy is then checked by a full AC load-flow, to detect possible residual violations. However, the increasing complexity of the modern transmission systems made the human expert approach very difficult. Linear programming methods have been developed to systematically solve the V/Q control problem on the whole system, achieving also further purposes, as the minimisation of the branch losses caused by reactive power flows. Nevertheless these methods, if applied to real size systems, are quite time consuming and therefore they are used only for large and spread violations. On-line control, instead, requires fast decisions, often even taking into account several possible system configurations.

Applications of Artificial Intelligence techniques to power system problems have attracted large interest in recent years. The heuristic human approach that is employed in many power system problems to find fast approximate local solutions, avoiding the full algorithmic solution, is well suited to automatic implementation by Expert Systems (ES). Indeed an ES emulates the human expert specific knowledge and empirical rules by a solution search strategy in a knowledge base. Unlike numerical algorithms, ESs can explain their results to the user in terms of heuristic choices. Therefore they do not substitute the operator, but they work as advisors, making easier the human supervision of the decisional process.

In last years many ESs and knowledge-based methods have been proposed to assist the system operator in V/Q control task. Some authors (Liu and Tomsovic, 1986; Tweed and Weatherwax, 1988) employed general purpose ES shells to build production systems based on rules written in

natural-like language. Cheng *et al.* (1988) used instead appropriate heuristic search strategies, directly coded in a declarative language. Although the former approach seems very direct and simple, the latter is often more effective in terms of coding task and operating performances.

Neural Networks (NN) are another Artificial Intelligence domain which gained broad popularity in last years. Roughly speaking, whereas ESs efficiently carry out logic tasks, NNs emulate instead the intuition capability of the human brain. Therefore the integration of the peculiarities of both methods seems very attractive. In power system domain, Khosla and Dillon (1994) showed that many classical problems could be effectively faced by integration of ES and NN capabilities.

This paper presents a Hybrid Expert System (HES) for the V/Q control task. It uses a modified best-first search strategy to explore the state space of a linear model of the relationship among voltage control devices and bus voltages, searching for feasible solutions. The search is guided by an index, representing the V/Q violations amount for each state. The search strategy has been coded in Prolog, language highly oriented to state space searching. The relationship among power system load profile and linear model parameters is modelled by a NN, therefore avoiding direct computing for different operating points. In the following sections, after some detail about V/Q control problem, the search strategy and the NN-based model will be described, as well the realised program and some numerical results on test systems.

2. THE V/Q CONTROL PROBLEM

Power system variables (bus voltage magnitudes, line currents, active and reactive synchronous machine generation) are subject to functional limits. The well known Load-Flow (LF) equations show a typical decoupling between active (active power, voltage phase) and reactive (reactive power, voltage magnitude) variables which greatly simplifies the V/Q control problem. Therefore it is possible to regulate voltage profile and reactive generation without appreciably modify active power flows. From this viewpoint, the V/Q control distinguishes the involved variables in dependent variables (load bus voltages V_L, generator bus reactive powers Q_G) and control variables (generator voltages V_G, shunt capacitor values B_L and transformer set points t_{ij}). Both dependent and control variables are subject to physical constraints.

Because the relationship among V/Q control and dependent variables is hidden in LF equations and it is not available in closed form, control actions are evaluated by sensitivity factors, calculated using the system decoupled Jacobian matrix (Ilic-Spong and Zaborszky, 1987). These factors form a Linear Model (LM) of the V/Q relationship that can be described in matricial form:

$$\begin{bmatrix} \Delta V_L \\ \Delta Q_G \end{bmatrix} = \begin{bmatrix} S_{vg} & S_{vb} & S_{vt} \\ S_{qg} & S_{qb} & S_{qt} \end{bmatrix} \begin{bmatrix} \Delta V_G \\ \Delta B_L \\ \Delta t_{ij} \end{bmatrix}$$

where the sub vectors ΔV_L and ΔQ_G represent changes in dependent variables caused by variations of control variable sub vectors ΔV_G, ΔB_L and Δt_{ij}. The sub matrices S_{vg}, S_{vb}, S_{vt} and S_{qg}, S_{qb}, S_{qt} contain the sensitivity factors among control and dependent variables. The LM has the twofold purpose of calculating the compensation action magnitude and evaluating the consequences that every control variable variation produces on all dependent variables.

The V/Q control problem is then solved adjusting the control variable values inside their operating limits so that the constraint violations of dependent variables are eliminated or at least reduced. Moreover new violations must not arise in buses electrically near to the adjusted controllers. System operator carries out this task heuristically, developing a control strategy based on his own experience and then testing it by a full AC LF.

3. THE EXPERT SYSTEM APPROACH

If several dependent or control variables are near its own functional limits, a dead-lock condition could occur, where apparently every attempted compensation manoeuvre fails, since it causes new violations in other buses. Problem solution, if it exists, requires the assessment of several alternative hypotheses, and also an expert operator could not succeed in keeping track of all of them. In these cases an ES can assist the system operator, quickly assessing several control hypotheses and presenting its results in terms of heuristic choices.

The ES emulates the operator's behaviour, using a solution search strategy which takes the place of the empirical compensation rules and a LM representing the specific operator knowledge of the power system. Because the LM depends on the system operating point, it should be recalculated when the operating point changes.

NNs have been often employed for function approximation. In general terms, NNs can easily approximate complex relationships after a suitable learning session based on a set of input-output pairs obtained from the relationship itself. It is then possible to use a NN for approximate the

layer, that transmits input vector to each PE of the hidden layer. The hidden layer is formed by k competitive Kohonen PEs, which are fully connected to the m Grossberg PEs of the output layer. Of course, n and m are respectively the dimensions of input and output vectors.

When an input vector is presented to the input layer, Kohonen PEs compete and the PE having lower Euclidean distance with the input vector is selected. The selected PE output is then set to one, whereas the other outputs are set to zero. Therefore only connections from winning Kohonen PE toward the output layer are activated. The output vector components are just the weight values of these connections.

Learning procedure assigns connection weights of the Kohonen and Grossberg layers minimising the global prediction error on the TS samples. In this way, each Kohonen PE represents a subset of the input space where some TS samples are clustered. The corresponding output is then the centroid of sample output vectors belonging to the cluster. Learning is accomplished in an iterative way, repeatedly presenting TS samples until the global prediction error falls below a predefined threshold.

In the proposed application, the input vector is formed by reactive loads of the system load buses, whereas the output vector components are the sensitivity factors. This choice has proved to be the most suitable one to represent the system operating point for V/Q control purposes. Details on the developed NNMs will be presented in the following section.

5. NUMERIC RESULTS

A test program based on the proposed search strategy has been developed in a 486 Personal Computer environment. The Search Strategy module has been coded in PDC Prolog, a Prolog dialect with powerful man-machine interface facilities. The Search Strategy module interacts with the NNM, implemented by a C-coded program executing the CPN recall phase. A LF Fortran program executes power system simulation and check calculations. CPN training sessions have been carried out off-line by a NN development package. Many tests have been executed on the standard networks IEEE 14-bus and IEEE 30-bus. Summary results and some significant cases will be now presented.

In Table 1 main data concerning the two developed NNMs are summarised. For both systems, a 1000 sample TS has been generated uniformly varying in a random way all loads and generations for an amount of ± 30% with respect to their base values. Afterwards each load value has been again randomly varied adding up a noise equal to ± 50% of its amount, and balancing then the generation. For each sample case the corresponding sensitivity table is calculated. Finally, each pair of load and sensitivity factor vectors has been recorded in TS.

Table 1 Summary of developed NNM features

System	Inputs	Outputs	Hid. PE's	RMS error	max error
IEEE-14	8	74	5	$5.74\ 10^{-4}$	$3.34\ 10^{-2}$
IEEE-30	21	277	9	$8.14\ 10^{-4}$	$6.11\ 10^{-2}$

For the studied systems, Table 1 reports the number of input, output, and hidden PEs of the corresponding NNM. Global root mean square error and maximum sample error obtained in learning session are also reported.

In Table 2 and 3 are summarised two examples carried out on the studied systems. In columns are reported variable related bus number and type, variable values before and afterwards the HES control action, variable functional limits. Last column specifies the variable category, control or dependent (c/d). All values are measured in per unit on the basis of 100 MVA and 250 kV.

First example, developed on the IEEE-14 bus system, is shown in Table 1. The operating point corresponds to base case, which presents four voltage violations in buses 7, 9, 11, 12 and a reactive power violation in generator bus 2. The proposed control action moves voltage set points of generator buses 1, 2, and 8, and transformer taps, increasing also the reactive power generated by the capacitor bank in bus 9.

Table 2 .Example on IEEE-14 bus system

Var. #	Type	original	correct	max	min	c/d
4	V_L	1.0138	1.0263	1.05	0.95	d
5	V_L	1.0166	1.0243	1.05	0.95	d
7	V_L	1.0596	0.9881	1.05	0.95	d
9	V_L	1.0539	0.9938	1.05	0.95	d
10	V_L	1.0493	0.9994	1.05	0.95	d
11	V_L	1.0560	1.0304	1.05	0.95	d
12	V_L	1.0549	1.0504	1.05	0.95	d
13	V_L	1.0499	1.0409	1.05	0.95	d
14	V_L	1.0332	0.9947	1.05	0.95	d
1	Q_G	-0.1238	0.3473	0.50	-0.40	d
2	Q_G	0.5197	0.2010	0.50	-0.40	d
3	Q_G	0.2825	0.1547	0.40	0	d
6	Q_G	0.1501	0.1536	0.24	-0.06	d
8	Q_G	0.1880	0.1846	0.24	-0.06	d
1	V_G	1.060	1.090	1.10	0.90	c
2	V_G	1.045	1.055	1.10	0.90	c
3	V_G	1.010	1.010	1.10	0.90	c
6	V_G	1.070	1.070	1.10	0.90	c
8	V_G	1.090	1.020	1.10	0.90	c
9	B_L	0.190	0.230	0.30	0	c
4-9	T_{ij}	0.969	1.10	1.10	0.90	c
5-6	T_{ij}	0.932	0.90	1.10	0.90	c
4-7	T_{ij}	0.978	1.09	1.10	0.90	c

A second more difficult case is depicted in Table 3. Starting from the base case, active and reactive loads of the IEEE-30 bus system have been randomly increased of an amount of 100% of their base value. Low voltage violations arise in buses 18, 19, 20, 23, 26, 27, 29 and 30. Moreover two reactive power violations occur in buses 2 and 8. The HES control strategy solves all violations increasing generator voltage and capacitor bank set points and slightly adjusting transformer taps.

Table 3 Example on IEEE-30 bus system

Var. #	Type	original	correct	max	min	c/d
3	V_L	1.0008	1.0246	1.05	0.95	d
4	V_L	0.9900	1.0082	1.05	0.95	d
6	V_L	0.9930	1.0058	1.05	0.95	d
7	V_L	1.0000	1.0075	1.05	0.95	d
9	V_L	1.0128	1.0516	1.05	0.95	d
10	V_L	0.9975	1.0489	1.05	0.95	d
12	V_L	1.0063	1.0382	1.05	0.95	d
14	V_L	0.9614	1.0004	1.05	0.95	d
15	V_L	0.9597	1.0036	1.05	0.95	d
16	V_L	0.9901	1.0308	1.05	0.95	d
17	V_L	0.9898	1.0383	1.05	0.95	d
18	V_L	0.9180	0.9678	1.05	0.95	d
19	V_L	0.9334	0.9845	1.05	0.95	d
20	V_L	0.9482	0.9995	1.05	0.95	d
21	V_L	0.9876	1.0445	1.05	0.95	d
22	V_L	0.9858	1.0442	1.05	0.95	d
23	V_L	0.9409	1.0012	1.05	0.95	d
24	V_L	0.9556	1.0347	1.05	0.95	d
25	V_L	0.9628	1.0192	1.05	0.95	d
26	V_L	0.9440	1.0015	1.05	0.95	d
27	V_L	0.9434	0.9873	1.05	0.95	d
28	V_L	0.9783	0.9919	1.05	0.95	d
29	V_L	0.9217	0.9666	1.05	0.95	d
30	V_L	0.9091	0.9547	1.05	0.95	d
1	Q_G	-0.0482	0.8728	0.90	-0.40	d
2	Q_G	0.8732	0.1183	0.60	-0.20	d
5	Q_G	0.0747	-0.0415	0.63	-0.15	d
8	Q_G	0.8878	0.5067	0.57	-0.15	d
11	Q_G	0.3887	0.4568	0.46	-0.10	d
13	Q_G	0.5304	0.3676	0.57	-0.15	d
1	V_G	1.050	1.1	1.1	0.9	c
2	V_G	1.045	1.055	1.1	0.9	c
5	V_G	1.010	1.01	1.1	0.9	c
8	V_G	1.010	1.01	1.1	0.9	c
11	V_G	1.050	1.1	1.1	0.9	c
13	V_G	1.050	1.06	1.1	0.9	c
10	B_L	0.19	0.28	0.3	0	c
24	B_L	0.04	0.27	0.3	0	c
4-12	T_{ij}	0.932	0.922	1.1	0.9	c
6-10	T_{ij}	0.969	0.969	1.1	0.9	c
6-9	T_{ij}	0.978	0.988	1.1	0.9	c
27-28	T_{ij}	0.968	0.968	1.1	0.9	c

6. CONCLUSION

An hybrid expert system that assists the system operator in V/Q control task has been presented. The proposed method uses a modified best-first strategy to explore the state space of a linear model of the voltage/reactive power relationship, searching for feasible solutions. The linear model is based on sensitivity factors and it depends on the system operating point.

A neural network has been trained to model this relationship, so avoiding the time consuming calculation of sensitivity factors in on-line operation. For this purpose, a CounterPropagation network seemed to represent the best compromise between learning session speed and recall phase accuracy. Disregarding sensitivity factors below a predefined threshold, the sensitivity table becomes rather sparse and the neural network approach could be applied also to real size power systems.

The proposed search strategy, based on recursive calls and backtracking cycles, has been coded in Prolog, language highly suitable for these purposes. An experimental environment has been built, where the Prolog search strategy interacts with the C-coded CPN routine and a Fortran load-flow program for power system simulation and check calculations.

Several tests performed on many standard systems have shown encouraging results. The proposed search strategy is basically applicable to real size systems, but further work is still necessary in the user interface module, so that expert system results can be clearly explained to the operator in terms of heuristic choices.

7. REFERENCES

Bratko, I. (1986). *Prolog Programming for Artificial Intelligence*. Addison-Wesley, Reading

Cheng, S.J., O.P. Malik, and G.S. Hope (1988). An Expert System for Voltage and Reactive Power Control of a Power System. *IEEE Trans. on Power Systems*, **3**, 1449-1455.

Hecht-Nielsen, R. (1987). Counterpropagation networks. *Applied Optics*, **26**, 4979-4984

Ilic-Spong, M. and J. Zaborszky (1987). Fundamentals of volt/var analysis and control. In: *Reactive Power: Basics, Problems and Solutions*, IEEE tutorial course, 87EH0262-6-PWR.

Khosla, R. and T.S. Dillon (1994). A neuro-expert approach to power system problems. *J. Engineering Intelligent Systems*, **2**, 71-78.

Liu, C.C. and K. Tomsovic (1986). An expert system assisting decision making of reactive power/voltage control. *IEEE Trans. on Power Systems*, **1**, 195-201.

Tweed, E.D., and J. Weatherwax (1988). A Volt/Var dispatch knowledge based system for electric utility using Prolog. *Proc. Symp. on Expert Systems Appl. to Power Systems, Stockholm-Helsinki*, ref. 15-15.

NEURAL NETWORK MODEL FOR DISSOLVED OXYGEN CONTROL IN A BATCH FERMENTER

Geng-Ware Hwang, Jinn-Jong Wong, Shu-Chen Chang, Kin-Ching Young

Union Chemical Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan 300, R.O.C.

Abstract. This work is to study the application of neural network model in self-learning control for batch process. A neural network model is developed to simulate the empirical control strategies using by a control engineer. The process system status and time varying set-point values are treated as input variables of the model. The manipulated variables are used as output variables. A moving-average method is used to describe the batch process dynamics and to condense the past history of process responses. Both efficiency and effectiveness of our neural network model are validated by plentiful experimental data from an *E. Coli* batch fermenter.

Key Words. Neural nets; adaptive control; batch process control; biocontrol; computer control; dynamic response; fermentation process; learning system; model reduction; model reference control.

1. INTRODUCTION

Advance in artificial intelligence allows the neural network technique to provide a self-learning method in adaptive control. Neural network model is a computational model in resemblance to human brain. Its architecture (Hecht-Nielsen, 1990; Kosko, 1992) consists in three or more layers, each layer is formed by several nodes, also called neurons or processing elements. The first network layer and the last network layer of a neural network model stand for network computation inputs and outputs. In each network nodes, input signals are multiplied by connection weights, summed up and transformed by a linear or nonlinear function, then transfer the output signal to other nodes.

The adaptive part of a neural network is in its layer-to-layer connection weights. In the learning phase of neural network computation the connection weights are adjusted automatically to fit training examples. After learning, the learned and unlearned data are sent into the neural network to perform recall test. The advantages of neural network are its quantitative reasoning and adaptive learning abilities. There are a number of articles on control applying neural networks such as in Miller et al. (1990), and in IFAC Symposium (1992) because of these advantages.

This research is to study the application of neural network model for nonlinear batch process control. In general, biochemical processes are multivariate and highly nonlinear. A batch fermenter is difficult to control through conventional control techniques. So a neural network model is developed to simulate the empirical control strategies using by a control engineer. The process system dynamic status and time varying set-point values of controlled variables are treated as input variables of the proposed neural network model. Computed values of manipulated variables are used as output variables. A refined model with the

A second more difficult case is depicted in Table 3. Starting from the base case, active and reactive loads of the IEEE-30 bus system have been randomly increased of an amount of 100% of their base value. Low voltage violations arise in buses 18, 19, 20, 23, 26, 27, 29 and 30. Moreover two reactive power violations occur in buses 2 and 8. The HES control strategy solves all violations increasing generator voltage and capacitor bank set points and slightly adjusting transformer taps.

Table 3 Example on IEEE-30 bus system

Var. #	Type	original	correct	max	min	c/d
3	V_L	1.0008	1.0246	1.05	0.95	d
4	V_L	0.9900	1.0082	1.05	0.95	d
6	V_L	0.9930	1.0058	1.05	0.95	d
7	V_L	1.0000	1.0075	1.05	0.95	d
9	V_L	1.0128	1.0516	1.05	0.95	d
10	V_L	0.9975	1.0489	1.05	0.95	d
12	V_L	1.0063	1.0382	1.05	0.95	d
14	V_L	0.9614	1.0004	1.05	0.95	d
15	V_L	0.9597	1.0036	1.05	0.95	d
16	V_L	0.9901	1.0308	1.05	0.95	d
17	V_L	0.9898	1.0383	1.05	0.95	d
18	V_L	0.9180	0.9678	1.05	0.95	d
19	V_L	0.9334	0.9845	1.05	0.95	d
20	V_L	0.9482	0.9995	1.05	0.95	d
21	V_L	0.9876	1.0445	1.05	0.95	d
22	V_L	0.9858	1.0442	1.05	0.95	d
23	V_L	0.9409	1.0012	1.05	0.95	d
24	V_L	0.9556	1.0347	1.05	0.95	d
25	V_L	0.9628	1.0192	1.05	0.95	d
26	V_L	0.9440	1.0015	1.05	0.95	d
27	V_L	0.9434	0.9873	1.05	0.95	d
28	V_L	0.9783	0.9919	1.05	0.95	d
29	V_L	0.9217	0.9666	1.05	0.95	d
30	V_L	0.9091	0.9547	1.05	0.95	d
1	Q_G	-0.0482	0.8728	0.90	-0.40	d
2	Q_G	0.8732	0.1183	0.60	-0.20	d
5	Q_G	0.0747	-0.0415	0.63	-0.15	d
8	Q_G	0.8878	0.5067	0.57	-0.15	d
11	Q_G	0.3887	0.4568	0.46	-0.10	d
13	Q_G	0.5304	0.3676	0.57	-0.15	d
1	V_G	1.050	1.1	1.1	0.9	c
2	V_G	1.045	1.055	1.1	0.9	c
5	V_G	1.010	1.01	1.1	0.9	c
8	V_G	1.010	1.01	1.1	0.9	c
11	V_G	1.050	1.1	1.1	0.9	c
13	V_G	1.050	1.06	1.1	0.9	c
10	B_L	0.19	0.28	0.3	0	c
24	B_L	0.04	0.27	0.3	0	c
4-12	T_{ij}	0.932	0.922	1.1	0.9	c
6-10	T_{ij}	0.969	0.969	1.1	0.9	c
6-9	T_{ij}	0.978	0.988	1.1	0.9	c
27-28	T_{ij}	0.968	0.968	1.1	0.9	c

6. CONCLUSION

An hybrid expert system that assists the system operator in V/Q control task has been presented. The proposed method uses a modified best-first strategy to explore the state space of a linear model of the voltage/reactive power relationship, searching for feasible solutions. The linear model is based on sensitivity factors and it depends on the system operating point.

A neural network has been trained to model this relationship, so avoiding the time consuming calculation of sensitivity factors in on-line operation. For this purpose, a CounterPropagation network seemed to represent the best compromise between learning session speed and recall phase accuracy. Disregarding sensitivity factors below a predefined threshold, the sensitivity table becomes rather sparse and the neural network approach could be applied also to real size power systems.

The proposed search strategy, based on recursive calls and backtracking cycles, has been coded in Prolog, language highly suitable for these purposes. An experimental environment has been built, where the Prolog search strategy interacts with the C-coded CPN routine and a Fortran load-flow program for power system simulation and check calculations.

Several tests performed on many standard systems have shown encouraging results. The proposed search strategy is basically applicable to real size systems, but further work is still necessary in the user interface module, so that expert system results can be clearly explained to the operator in terms of heuristic choices.

7. REFERENCES

Bratko, I. (1986). *Prolog Programming for Artificial Intelligence*. Addison-Wesley, Reading

Cheng, S.J., O.P. Malik, and G.S. Hope (1988). An Expert System for Voltage and Reactive Power Control of a Power System. *IEEE Trans. on Power Systems*, **3**, 1449-1455.

Hecht-Nielsen, R. (1987). Counterpropagation networks. *Applied Optics*, **26**, 4979-4984

Ilic-Spong, M. and J. Zaborszky (1987). Fundamentals of volt/var analysis and control. In: *Reactive Power: Basics, Problems and Solutions*, IEEE tutorial course, 87EH0262-6-PWR.

Khosla, R. and T.S. Dillon (1994). A neuro-expert approach to power system problems. *J. Engineering Intelligent Systems*, **2**, 71-78.

Liu, C.C. and K. Tomsovic (1986). An expert system assisting decision making of reactive power/voltage control. *IEEE Trans. on Power Systems*, **1**, 195-201.

Tweed, E.D., and J. Weatherwax (1988). A Volt/Var dispatch knowledge based system for electric utility using Prolog. *Proc. Symp. on Expert Systems Appl. to Power Systems, Stockholm-Helsinki*, ref. 15-15.

NEURAL NETWORK MODEL FOR DISSOLVED OXYGEN CONTROL IN A BATCH FERMENTER

Geng-Ware Hwang, Jinn-Jong Wong, Shu-Chen Chang, Kin-Ching Young

Union Chemical Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan 300, R.O.C.

Abstract. This work is to study the application of neural network model in self-learning control for batch process. A neural network model is developed to simulate the empirical control strategies using by a control engineer. The process system status and time varying set-point values are treated as input variables of the model. The manipulated variables are used as output variables. A moving-average method is used to describe the batch process dynamics and to condense the past history of process responses. Both efficiency and effectiveness of our neural network model are validated by plentiful experimental data from an *E. Coli* batch fermenter.

Key Words. Neural nets; adaptive control; batch process control; biocontrol; computer control; dynamic response; fermentation process; learning system; model reduction; model reference control.

1. INTRODUCTION

Advance in artificial intelligence allows the neural network technique to provide a self-learning method in adaptive control. Neural network model is a computational model in resemblance to human brain. Its architecture (Hecht-Nielsen, 1990; Kosko, 1992) consists in three or more layers, each layer is formed by several nodes, also called neurons or processing elements. The first network layer and the last network layer of a neural network model stand for network computation inputs and outputs. In each network nodes, input signals are multiplied by connection weights, summed up and transformed by a linear or nonlinear function, then transfer the output signal to other nodes.

The adaptive part of a neural network is in its layer-to-layer connection weights. In the learning phase of neural network computation the connection weights are adjusted automatically to fit training examples. After learning, the learned and unlearned data are sent into the neural network to perform recall test. The advantages of neural network are its quantitative reasoning and adaptive learning abilities. There are a number of articles on control applying neural networks such as in Miller et al. (1990), and in IFAC Symposium (1992) because of these advantages.

This research is to study the application of neural network model for nonlinear batch process control. In general, biochemical processes are multivariate and highly nonlinear. A batch fermenter is difficult to control through conventional control techniques. So a neural network model is developed to simulate the empirical control strategies using by a control engineer. The process system dynamic status and time varying set-point values of controlled variables are treated as input variables of the proposed neural network model. Computed values of manipulated variables are used as output variables. A refined model with the

three-moving-average method is developed to describe the batch process dynamics and to condense the past history into representable average values. Both efficiency and effectiveness of the proposed neural network model are validated by *E. Coli* batch fermentation experiments.

2. NEURAL NETWORK MODEL

A neural network model is proposed to achieve adaptive control for batch fermentation process. Experimental data collected include the following process variables: time (t), optical density (OD), agitator speed (RPM), air flowrate (AF), dissolved oxygen concentration (DO), temperature, and pH. The optical density is an indirect measurement of cell density in our experiment. A dissolved oxygen control system is chosen as the demonstration example. The controlled variables are dissolved oxygen concentration and optical density. The manipulated variables are agitator speed and air flowrate. Two other variables, temperature and pH, are kept constant in our experiments. Glucose concentration, an important process variables, is unmeasured in our experiment and is considered as unmeasured disturbance.

2.1. Basic Model

The back-propagating network model is chosen to learn the operator's control strategy. The empirical control strategy is learned in an quantitative manner. Though the multivariate correlations of process variables are in a control engineer's mind, such nonlinear and implicit correlations can be learned by a neural network model through voluminous operating data examples. A basic model for dissolved oxygen control is shown in Fig. 1. This model can be applied to steady process conveniently. The basic model is not suitable to batch process because the control model must depend on time-varying process behavior. The model parameters (connection-weights) have to be time dependent since the process itself is changing with time. In order to make the model easy to be used, it is desirable to keep the control model in a stationary manner. In this sense, the past process responses should be involved in the inputs to the model to cancel out the time-varying process characteristic.

Fig. 1. A basic model for dissolved oxygen control

2.2. Improved Model

An improved learning model is proposed to meet the requirement of batch process. The batch fermentation process model is assumed to time-independent if the model was correlated to past time-series values. The recent manipulated variable values (AF and RPM) are determined by the new set-points (OD and DO) together with past time-series values (AF, RPM, OD, and DO). The improved model, considers the past process responses, is illustrated as in Fig. 2.

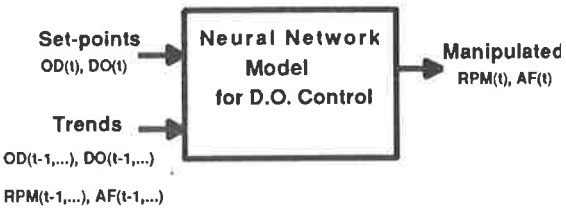


Fig. 2. An Improved model with trends for dissolved oxygen control

The past historical values of OD, DO, RPM, and AF has to be condensed to enhance the computing speed and memory saving for the improved model. The influences of correlated historical values may be slow in the growing phase of fermentation but fast in the starting phase. In a batch process, the process responses may be influenced by both long-lag and short-lag disturbances simultaneously. Therefore, it is difficult to estimate appropriate learning sampling time intervals and correlating periods of past time-series values for each process variables. The sampling time intervals for network learning has to be as large as possible so as to save the network learning time required for the whole batch duration. The small correlating periods of

past values can improve computing speed in each sampled point learning.

2.3. Refined Model

A convenient but heuristic method, the three-moving-average method, is developed here to overcome difficulties imparted by various lags (discussed in 2.2). Three representative moving averages with short-term, medium-term, and log-term averaging periods are computed to cover correlated historical values for each process variable. Each short-term moving-average value is computed within an averaging period of 0.001 of the batch cycle-time. The medium-term moving-average value is computed within an averaging period of 0.004 of the cycle-time while the long-term moving-average value; within 0.016 of the cycle time. To enhance the generality of the moving-average method, the batch cycle time is chosen to be the time basis. 0.001 for the short-term averaging period is suitable for rapid-response processes, for example, *E. Coli* fermentation. A value higher than 0.001 for short-term averaging period may be required for slow response processes. The refined model with moving averages is shown in Fig. 3.

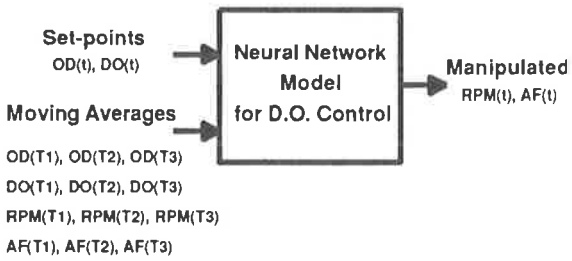


Fig. 3. A refined model with moving averages for dissolved oxygen control

3. APPLICATION RESULTS

E. Coli experiments are carried out in a fermenter in laboratory. Plentiful data are obtained from the on-line monitoring system to demonstrate the efficiency and effectiveness of our neural network model. A series of computer programs are written for on-line data collection, data transformation to neural network model input format, recall result charting, and numerical analysis for this application.

Table 1 Training data summary

Batch no.	No. of pts	Batch duration*	OD*			DO*			RPM*			AF*		
			Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
800606	2,562	0.88	0.17	0	0.3	0.23	0	1	0.82	0.05	0.91	0.45	0	0.85
800613	231	0.71	0.13	0	0.44	0.93	0.62	1	0.41	0	0.62	0.2	0	0.25
800614	346	0.28	0.39	0.23	0.51	0.93	0.91	0.93	0.48	0.07	0.68	0.05	0	0.25
800619	1,590	0.58	0.98	0.01	1	0	0	0.01	0	0	0	0	0	0
800620	2,008	1.09	0.38	0	1	0.58	0	1	0.7	0	1	0.3	0	1
800624	279	0.22	0.5	0	1	0.87	0	1	0	0	0.4	0	0	0
800625	528	0.95	0.04	0.01	0.08	0.48	0.3	0.76	0.28	0	1	0.15	0	1
800626	708	0.94	0.13	0	0.24	0.19	0	1	0.52	0.07	1	0.1	0	0.6
800627	147	0.22	0.13	0.04	0.19	0.17	0	1	0.47	0.08	0.94	0.15	0	0.85
801012	439	0.13	0.01	0	0.13	0.69	0.04	1	0.32	0.1	0.79	0.3	0.05	0.4
801023	1,806	0.47	0.27	0	0.44	0.4	0	1	0.67	0	1	0.25	0	0.5

* All values are normalized.

Some 10,600 data points in more than ten batches of experimental data are collected by on-line monitoring of the *E. Coli* fermenter. The experimental data collected for network training are summarized in Table 1. It has been noticed that, experimental data of two batches (800619 and 800624) are not in good control. But data of both batches are taken as disturbances to control strategy learning. Part of experimental data (Batches 800606 to 800627) are taken as learning examples while others (about 2,200 points) are treated as test data for alike-case reasoning. In order to check the alike-case applicability of our neural network model, part of the experimental data are retained and not included in the learning examples.

The commercial software *NeuralWorks Professional II* is used in neural network computation. The back-propagating network architecture is applied in this work. The most time-consuming stage of back-propagating network model computation, the learning stage of our network model, is 1,000 times faster than the actual process when runs on a 486 personal computer. The total operation duration for the 11 batches is about 65 hours. The learning phase of the refined network model takes only about 4 minutes.

After learning, part of the learned examples are recalled to test the recall power of the refined model. The recall result of Batches 800606, 800620, and 801023 are illustrated in Fig. 3, 4, and 5. The former two batches (800606 and 800620) are learned before recalling. But the Batch 801023 is not learned before recalling to test alike-case reasoning.

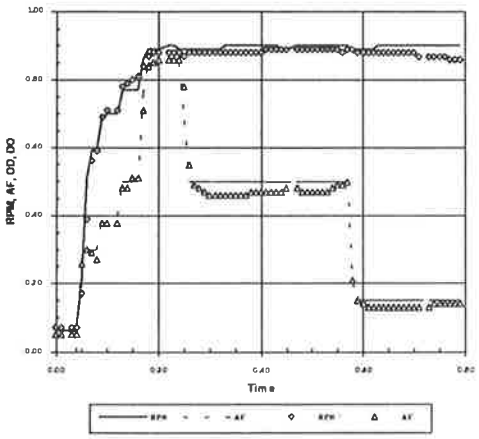


Fig. 4 Recall result of Batch 800606

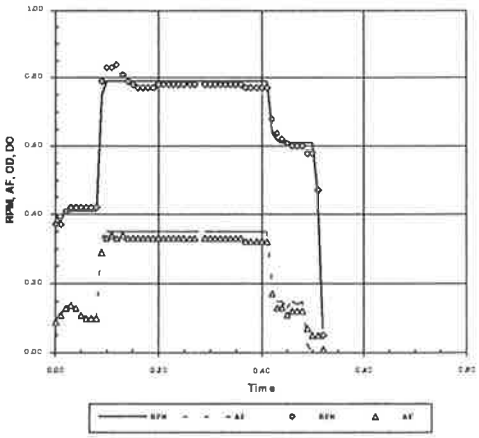


Fig. 5 Recall result of Batch 800620

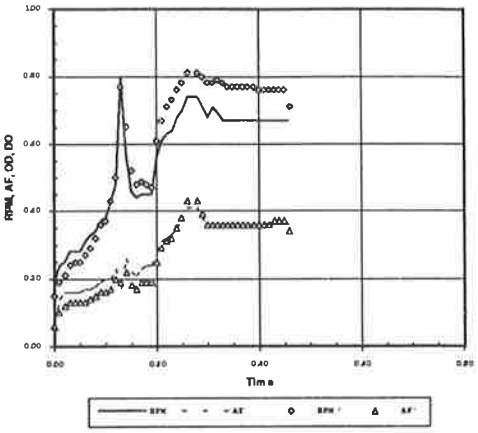


Fig. 6 Recall result of Batch 801023

A basic statistical analysis is carried out to comprehend the recall error distribution. Both learned and unlearned data examples are recall-tested. After statistical analysis of recall results, as shown in Table 2, the averages of RMS (root-mean-square) recall errors are about 2% for learned points and 4% for unlearned points.

Table 2. Recall result summary of the refined model with moving averages

Batch no.	Learned	RPM		AF		Average
		RMS error	Std. dev.	RMS error	Std. dev.	RMS error
800606	Y	0.021	0.017	0.030	0.027	0.025
800620	Y	0.020	0.020	0.022	0.019	0.021
801012	N	0.017	0.016	0.014	0.013	0.015
801023	N	0.070	0.051	0.027	0.021	0.048

As discussed in Section 2 ('Neural Network Model'), the basic model is not suitable to batch process. It is noticed that the recall error for the basic model is large. The recall results for the basic model are summarized in Table 3. The recall-error averages are about 15% for learned points and 15% for unlearned points.

Table 3. Recall result summary of the basic model

Batch no.	Learned	RPM		AF		Average
		RMS error	Std. dev.	RMS error	Std. dev.	RMS error
800606	Y	0.177	0.114	0.183	0.179	0.180
800620	Y	0.149	0.090	0.099	0.099	0.120
801012	N	0.132	0.109	0.173	0.146	0.152
801023	N	0.155	0.122	0.127	0.100	0.141

4. CONCLUSION

Owing to the quantitative reasoning and self-learning ability, this work apply successfully a neural network model to establish a fermenter control system so as to overcome the complexity of biochemical process control.

Considering the time-varying characteristics of batch process control, we combine the three-moving-average method and back propagating network model to form the neural-network control system architecture.

The proposed neural network model is tested more than ten batches on experimental data from an *E. Coli* fermenter. According to the application results, we think the accuracy and speed of the refined neural network model can achieve requirements of real-time industrial applications.

5. REFERENCES

Hecht-Nielsen, R. (1990). *Neurocomputing*, Chap. 2. Addison-Wesley, Reading, MA.

IFAC Symposium (1992): *Dynamics and Control of Chemical Reactors, Distillation Columns and Batch Processes*, Apr. 26-29. College Park, MD.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems*, Prentice-Hall, Englewood Cliffs, NJ.

Miller, III, W.T., R.S. Sutton, and P.J. Werbos, ed. (1990). *Neural Networks for Control*, MIT Press, Cambridge, MA.

RULE BASED INTERPOLATING CONTROL - FUZZY AND ITS ALTERNATIVES

Mikael Johansson

Department of Automatic Control, Lund Institute of Technology,
Box 118, S-221 00 Lund, Sweden, mikaelj@control.lth.se

Abstract. Design of control strategies based on heuristics is discussed. The most well-known example of this is fuzzy control. Fuzzy logic systems describe nonlinear mappings in terms of linguistic rules and an interpolative reasoning method. A useful tool for understanding advances in fuzzy control is proposed by considering the similarity between a fuzzy system and a look-up table with an interpolation method. This point-of-view also results in an alternative rule based paradigm consisting of crisp rules and an interpolative reasoning method. Stripping the colorful language, this is nothing but the classical function approximation approach.

Keywords. Fuzzy Control, Fuzzy Systems, Interpolation, Approximation Theory.

1. INTRODUCTION

Fuzzy logic was introduced in the 60's [Zadeh, 1965] as a way to formalize reasoning with imprecise or vague information. Soon, the first laboratory experiments with controllers based on fuzzy logic where presented [Mamdani, 1974]. An early industrial application was control of a cement kiln, where the fuzzy logic framework was used to mimic an experienced process operator's control actions [Holmblad and Ostergaard, 1982]. In the last few years, the area of fuzzy control has received renewed attention. This is partly because of its intuitive appeal to non-control engineers and partly due to some successful commercial applications. However, current hype and oversell has also resulted in an increasing amount of scepticism towards fuzzy control [Bernhard, 1992] [Elkan, 1993]. Another important reason for this scepticism is the lack of methods for comparing advances in fuzzy control with results established in the conventional control theory.

Special with fuzzy controllers is that the mapping from process observations to control action is described by a set of rules. By describing how rules and interpolation are combined in fuzzy systems, this paper aims at narrowing the gap between hype and scepticism. In principle, the rules specify the mapping for a few characteristic process conditions. For other process states, the control action is obtained by an interpolation carried out by the reasoning process. From this point-of-view, fuzzy systems is one of several possibilities to combine linguistic rules and an interpolative reasoning method.

This paper is organized as follows. First, control strategies based on rules and interpolative reasoning is discussed in general. The discussion is made more specific by describing how rules and interpolation is combined in fuzzy systems. A piecewise linearization of fuzzy system mappings is proposed as a useful tool for understanding fuzzy nonlinearities. As an alternative to fuzzy systems, the combination of linguistic rules and analytical interpolation is suggested. Finally, some work on adaptive fuzzy systems is reviewed in terms of interpolation systems.

2. RULE BASED INTERPOLATING CONTROL

If a mathematical process model is known, control theory provides a broad selection of design routines for computing a controller from specifications. In practice, however, there exists several cases when a valid or useful process model cannot be found. These problems are often solved by simply deciding a control strategy, possibly with a few adjustable parameters. The parameters are then tuned so that the controlled system behaves in the desired manner, see Fig. 1. One control structure often used in this way is the PID controller.

A somewhat different approach is based on the

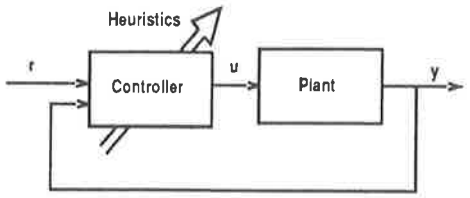


Figure 1. How can we control a plant for which no mathematical process model exists ?

following observation: Even if the plant behaviour fails to be adequately described by a mathematical process model, it may often be controlled by a skilled process operator. A straight-forward way to control such a process is to simply mimic the process operator's control actions. If the operator can state a set of rules that describes how to control the plant, a correct interpretation of these rules gives an automatic control close to the one performed by the human expert.

It is important to notice that in the same way as control theoretic approaches rely on the accuracy of a mathematical process model, expert control relies on adequate acquisition and interpretation of expert knowledge. Insight in this problem will be developed by noticing that "The problem of learning a mapping between an input and an output space is essentially ...equivalent to the problem of estimating the system that transforms inputs into outputs given a set of examples of input-output pairs. A classical framework for this problem is function approximation." [Poggio and Girosi, 1989].

Inspired by this, the following requirements for efficient design of rule based feedback controllers may be postulated:

1. The behaviour of the controller should be described by a set of linguistic rules. These rules contain an expert's control actions for some characteristic process conditions.
2. For cases where no exact control action is specified, the system should be able to generalize from knowledge of the control action for similar process conditions.
3. The linguistic description should have a transparent influence on the controller nonlinearity. This facilitates effective controller tuning.

The idea of mimicing a human expert's control actions has motivated several AI-based control approaches. One example is the BOXES algorithm used in [Sammut and Michie, 1991]. Here, the rules define constant control action on rectangular patches in controller state space. Thus, the mapping described by a set of rules is interpreted as a piecewise constant function. With a limited set of rules, the BOXES algorithm is only capable of representing a very restricted class of functions.

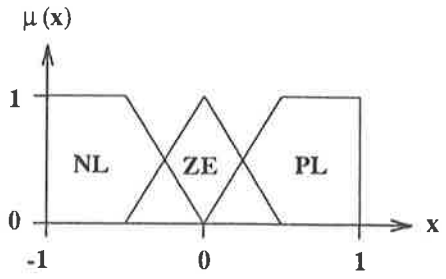


Figure 2. Membership functions defining linguistic values NL, ZE and PL used in the fuzzy PD controller.

Fuzzy systems, on the other hand, can theoretically represent every continuous function [Wang and Mendel, 1992]. In order to understand what can practically be achieved by fuzzy controllers, it is important to understand how rules and interpolation is combined in fuzzy systems. This understanding will be developed next.

3. FUZZY SYSTEMS...

In fuzzy controllers, the fuzzy system is used to perform a nonlinear mapping from the process observations to the control action. A fuzzy system consists of a set of rules, each stating the control action to be taken for a given process state, e.g.:

```
IF Error IS Zero THEN Control IS Zero
```

In order to interpret these rules by fuzzy logic and approximate reasoning, each linguistic statement is represented by a fuzzy set. For an introductory to the restricted fuzzy logic used in fuzzy controllers, see [Lee, 1990].

The main idea of fuzzy set theory is that statements like `Error IS Zero` are not just simply true or false but can be fulfilled to any degree. As a consequence, several rules of the fuzzy rule base may apply at the same time. This causes an interpolation effect. The nature of this interpolation is governed by the defuzzification strategy used. This article will primarily be concerned with product-sum inference, AND defined as a product and Center-of-Gravity defuzzification, hereafter referred to as product-sum reasoning.

Describing a function with fuzzy logic based rules is not very different from describing a nonlinearity by a look-up table with an interpolation procedure. Roughly speaking, the fuzzy sets and rules correspond to the table while the reasoning method corresponds to the interpolation. The similarity between the two approaches is especially transparent in the case of product-sum reasoning and piecewise linear input sets with full overlapping, see Fig.2. The relation between the fuzzy system parameters and the described nonlinearity can be described as follows [Johansson, 1994]:

- The input fuzzy sets partition the controller hypersurface into a set of interpolation intervals.
- The location of the output fuzzy sets determine the function values at the interval endpoints.
- The ratio between areas of the output fuzzy sets determine the slope of the nonlinearity at these endpoints.

The relation between the fuzzy system parameters and the nonlinearity will be illuminated by a small example:

EXAMPLE 1—A Fuzzy PD Nonlinearity

In a fuzzy controller of PD-type, the fuzzy system is used to perform a nonlinear mapping from the filtered control errors to the control action. One such mapping (with poor transient behaviour) can be described by the following nine linguistic rules:

		error		
		NL	ZE	PL
error rate	PL	ZE	PL	PL
	ZE	NL	ZE	PL
	NL	NL	NL	ZE

The linguistic values of each signal are defined by standard piecewise linear fuzzy sets with full overlapping, see Fig. 2. The influence of fuzzy set parameters on the nonlinearity is visualized stepwise. Fig. 3 shows how the fuzzy input sets partition input-space. Fig. 4 shows how the rule base and the output fuzzy sets determine the function values at some key-points. The interpolation carried out by the reasoning process results in the final nonlinearity of Fig. 5.

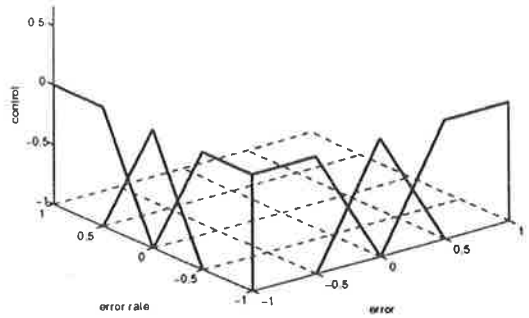


Figure 3. The fuzzy input sets partition controller state-space.

The analogy with function approximation gives good insight in practical possibilities and limitations of fuzzy system mappings. For the widely used class of fuzzy systems described above, the input fuzzy sets can be seen as basis functions for the interpolation. From this observation, it is clear

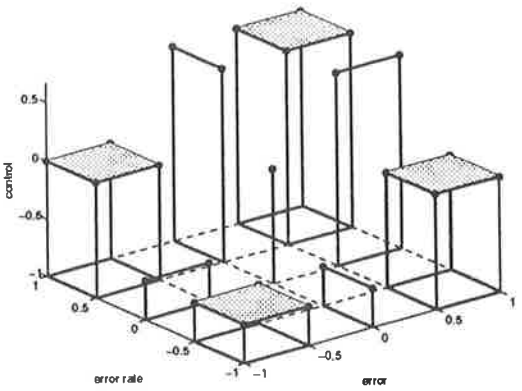


Figure 4. The rules specify the function values at some key-points.

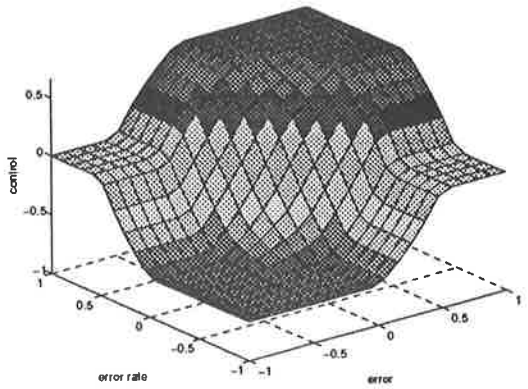


Figure 5. The reasoning process interpolates.

what happens if we use fuzzy sets of other shape or distribution.

It is interesting to notice that for this class of fuzzy systems, there exists an equivalent analytical description of the fuzzy system mapping in terms of a collection of first-order rational polynomials. The transformation from linguistic to analytical description of the nonlinearity can be used for fast implementation on standard hardware [Johansson, 1994].

4. ...LINEARIZATION ...

One reason for the current scepticism towards fuzzy control is the lack of methods for classical control engineers to relate the advances in fuzzy control to established ones in their own fields of work. A possible remedy for this would be a method to compute and visualize approximate fuzzy nonlinearities which requires only a basic knowledge of fuzzy systems. For this purpose, piecewise linearization of the fuzzy system mapping can be a very good tool. Piecewise linearization provides a method to translate a fuzzy system design to a look-up table. The approximate reasoning process is replaced with analytical linear interpolation in the table. The algorithm can be formulated as follows:

ALGORITHM 1— Piecewise Linearization.

1. For each input, create an input vector containing the modal points of the fuzzy sets of this input. The modal points of a fuzzy set are the points bounding the interval where the set is the only set with non-zero membership.
2. For each point in the Cartesian space of the input vectors, determine the corresponding output. Since only one rule apply at these points, the computation is trivial.
3. Use multi-linear interpolation to obtain a control surface.

The algorithm will be demonstrated on the fuzzy PD design of Example 1.

EXAMPLE 2—Fuzzy PD continued.

To make a piecewise linearization of the fuzzy PD design, Algorithm 1 is used:

1. Determine the modal points of the fuzzy sets of each input.

$$\mathcal{I}_1 = \mathcal{I}_2 = [\underbrace{-1}_{NL} \quad \underbrace{-1/2 \quad 0}_{ZE} \quad \underbrace{1/2 \quad 1}_{PL}]$$

2. Since Center-of-Gravity defuzzification is used in the original design, possible output values are the centroids of the fuzzy output sets, i.e.

$$\mathcal{O} = [\underbrace{-2/3}_{NL} \quad \underbrace{0}_{ZE} \quad \underbrace{2/3}_{PL}]$$

   For each point in Cartesian space $\mathcal{I}_1 \times \mathcal{I}_2$, determine the corresponding output. This step uses the rule base to obtain the following table mapping inputs to outputs:

$$\mathcal{T} = \begin{pmatrix} -2/3 & -2/3 & -2/3 & 0 & 0 \\ -2/3 & -2/3 & -2/3 & 0 & 0 \\ -2/3 & -2/3 & 0 & 2/3 & 2/3 \\ 0 & 0 & 2/3 & 2/3 & 2/3 \\ 0 & 0 & 2/3 & 2/3 & 2/3 \end{pmatrix}$$

3. Bi-linear interpolation is used in this table.

The resulting nonlinearity is shown in Fig.6.

If the input fuzzy sets are triangular or trapezoidal with full overlapping, the approach of piecewise linearization can be shown to give the exact mapping in case of (a) product-sum reasoning and output sets with equal area, (b) product-sum inference with singleton outputs and (c) Sugeno-type control with constant outputs. By requiring the output fuzzy sets to be rectangular with no overlap, case (a) can be extended to other

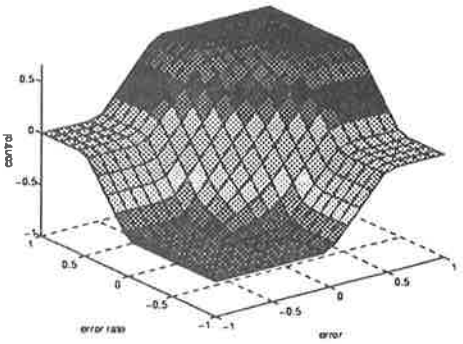


Figure 6. Nonlinearity defined by piecewise linearization.

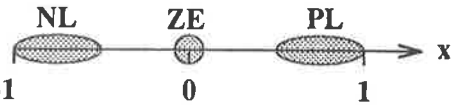


Figure 7. Alternative definition of linguistic values.

reasoning methods [Galichet and Foulloy, 1993]. Even if these requirements are not fulfilled, the piecewise linearization gives a good approximation of the fuzzy system mapping [Boverie *et al.*, 1993]. For further literature on piecewise linear fuzzy controllers, see [Meyer-Gramann, 1993].

5. ...AND THE ALTERNATIVES.

Looking back at the previous sections, it is natural to ask if it is motivated to use the complete machinery with fuzzy sets, fuzzy logic and approximate reasoning just to translate a set of linguistic rules to a nonlinear mapping. Moreover, a fuzzy controller has a very large number of adjustable parameters. Although the rules have a local influence on the nonlinearity, to what extent can these numerous parameters be tuned using heuristics?

The translation from linguistic rules to a look-up table can be made very directly and intuitively. One way of specifying the linguistic statements is simply to define the regions for which the statement is completely true. Other regions are left blank in order to indicate uncertainty or lack of exact knowledge, as in Fig. 7.

It is probable that the expert can point out the most characteristic features of the control strategy. Straight-forward analytical interpolation between the key-points defined by the rules may thus be a valid generalization of the stated knowledge. The investigations in [Johansson, 1993] indicated that the type of interpolation was of minor importance. It is then natural to choose an interpolation routine that is as simple as possible. Appropriate interpolation methods include multi-linear or cubic spline interpolation. This combination of linguistic rules and interpolative reasoning gives a transparent relation between the linguistic and

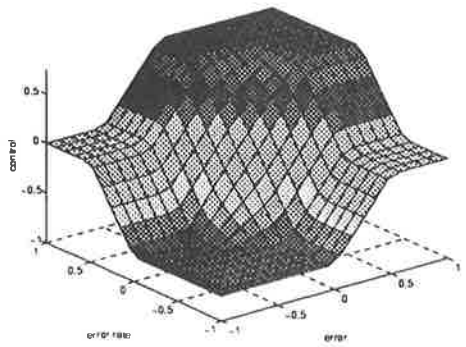


Figure 8. Nonlinearity defined by the rule based paradigm using analytical interpolation.

the analytical description of the nonlinearity. How this method can be used for a rule based PD design will be shown in the following example.

EXAMPLE 3—Fuzzy PD continued.

A nonlinear PD controller is designed on a heuristic basis and described by the nine linguistic rules from Example 1. The designer of these rules is then asked to specify for what inputs the control action, the control error and its derivative can be considered negative large, zero and positive large. The result of this query is shown in Fig. 7. These specifications are translated to a look-up table and combined with bi-linear interpolation. This design results in a nonlinearity in Fig. 8. Note the strong similarity with Fig. 6. The difference comes from the difference in interpretation of the linguistic values of the control variable.

6. NONLINEAR MODELERS

Much excellent work has been made in the field of adaptive control of linear processes. An additional tool of great use to a control engineer is a component for adaptive modeling of static nonlinearities. Such a component can be used for compensation of nonlinearities on the input or output of the process. This includes sensor linearization, adaptive nonlinear feed forward and nonlinear process identification. Several industrial applications of adaptive modelers are mentioned in [Heiss, 1994].

As shown previous in this paper, fuzzy systems are very similar to function approximators based on analytical interpolation methods. Adaptive fuzzy systems are in the same way related to adaptive function approximators based on analytical interpolation. This section will brief some work in adaptive nonlinear modeling and adaptive fuzzy systems. In essence, the same component has been developed under two different names and with two different areas of use: fuzzy systems model a complete control law while function approximators model some specific process characteristic. Further, the fuzzy systems approach provides an

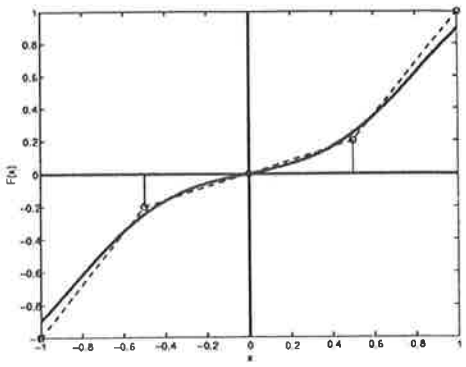


Figure 9. The unknown function (solid line) is approximated by a linear interpolation (dash-dotted line) using the table values ('o').

intuitive front-end to obtain the initial function approximation from a human expert.

Consider the problem of approximating an unknown function $\mathcal{F}(x)$ using a look-up table with an interpolation procedure. Typically, the table is represented by an array of inputs and corresponding output values. For inputs not explicitly stated in the table, the output is interpolated using some regularity condition of the unknown function. Often, the regularity condition is an assumption that the unknown function is continuous of order n. This results in spline interpolation of order $n+1$ [de Boor, 1978]. The problem of approximating an unknown function by a piecewise polynomial of order 1 is shown in Fig. 9.

The most straight-forward adaption method is to only adjust the function values in the look-up table. One adaptive nonlinear modeler based on this idea and linear interpolation was derived in [Åström, 1985]. Similar adaption is used in adaptive fuzzy systems, see e.g. [Wang, 1992].

Obviously, the accuracy of these approximations is highly dependent on the partitioning of the interpolation intervals. One way to allow for higher accuracy is to insert new control points at the input values for which the approximation error has its maximum [Kincaid and Cheney, 1991]. Kavli [Kavli, 1990] has developed a learning method that, given a set of input-output pairs, iteratively inserts control points nonuniformly to achieve the prescribed accuracy of the mapping. The parameters of the approximation are adapted on-line using a steepest descent method.

The insertion of new knots in a spline approximation corresponds to generation of new rules in a fuzzy design. Automatic rule generation is included in a fuzzy modeling algorithm presented in [Katayama *et al.*, 1994]. This algorithm uses a constrained optimization procedure to adjust both input and output fuzzy sets so that the fuzzy system mapping matches a set of input-output pairs. If the prescribed accuracy is not achieved by the optimization procedure, additional rules are inserted

to describe the region where the approximation error has its maximum. In the function approximation framework, this algorithm adjusts both the location of the control points and the corresponding function values. Further, new control points are inserted if necessary.

7. CONCLUSIONS

Fuzzy control was considered one of several possibilities to combine a rule based formalism with an interpolative reasoning method. Three reasonable requirements were postulated for effective rule based control design.

Insight in fuzzy systems were developed by noticing that describing a control strategy by a set of rules is equivalent to the function approximation problem. The most widely used classes of fuzzy systems where explained being very similar to interpolation systems based on splines. This analogy was later used to develop an understanding of advances in adaptive fuzzy systems and relate them to other techniques for modeling static nonlinearities.

For control engineers with no fuzzy systems background, piecewise linearization was proposed as a tool for understanding and visualizing nonlinearities of fuzzy controllers. As an alternative to fuzzy systems, an intuitive and simple way of mapping linguistic rules to a look-up table was described. Using analytical interpolation in this table results in systems with similar approximation capabilities as fuzzy systems.

8. REFERENCES

ÅSTRÖM, K. J. (1985): "The adaptive nonlinear modeller." Report TFRT-3178, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

BERNHARD, P. (1992): "Fuzzy control: Facts, japan, and europe." *ECC News Letter*, May.

BOVERIE, S., C. RAYMOND, and J. L. QUELLEC (1993): "Practical realization of fuzzy controllers comparison with conventional methods." In *Proceedings of the EUFIT '93*, pp. 149–155, Aachen.

DE BOOR, C. (1978): *A practical guide to splines.* Applied Math. Sciences Vol.27. Springer Verlag, New York.

ELKAN, C. (1993): "The paradoxical success of fuzzy logic." In *Proceedings of the AAAI.*

GALICHET, S. and L. FOULLOY (1993): "Fuzzy equivalence of classical controllers." In *Proceedings of the EUFIT '93*, volume I, pp. 1567–1573, Aachen.

HEISS, M. (1994): "Inverse passive learning of an input-output-map through update-spline-smoothing." *IEEE Transactions on Automatic Control*, **39:2**, pp. 259–268.

HOLMBLAD, L. and J. OSTERGAARD (1982): "Control of a cement kiln by fuzzy logic." In GUPTA and SANCHEZ, Eds., *Fuzzy Information and Decision Processes.* North-Holland, Amsterdam.

JOHANSSON, M. (1993): "Nonlinearities and interpolation in fuzzy control." Master thesis ISRN LUTFD2/TFRT--TFRT-5491--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

JOHANSSON, M. (1994): "The nonlinear nature of fuzzy control." In *Proceedings of the 4.e Dortmunder Fuzzy Tage.*, pp. 314–321.

KATAYAMA, R., Y. KAJITANI, and Y. NISHIDA (1994): "A self generating and tuning method for fuzzy modelling using interior penalty method and its application to knowledge acquisition of fuzzy controller." In KANDEL and LANGHOLZ, Eds., *Fuzzy Control Systems*, pp. 275–294. CRC Press.

KAVLI, T. (1990): "Nonuniformly partitioned piecewise linear representation of continuous learned mappings." In *Proceedings of the IEEE International Workshop on Intelligent Motion Control*, pp. 115–122, Istanbul, Turkey.

KINCAID, D. and W. CHENEY (1991): *Numerical Analysis.* Brooks/Cole Publishing Company, Pacific Groove, California.

LEE, C. C. (1990): "Fuzzy logic in control systems: Fuzzy logic controller part I and II." *IEEE Transactions on Man, Systems and Cybernetics*, **20:2**, pp. 404–435.

MAMDANI, E. (1974): "Application of fuzzy algorithm for control of simple dynamic plant." *Proc. IEE*, **No 121**, pp. 1585–1588.

MEYER-GRAMANN, K. (1993): "Easy implementation of fuzzy controller with a smooth control surface." In *Proceedings of the EUFIT'93*, volume 1, pp. 117–123, Aachen.

POGGIO, T. and F. GIROSI (1989): "A theory of networks for approximation and learning." Technical Report A.I. Memo No.1140, M.I.T.

SAMMUT, C. and D. MICHIE (1991): "Controlling a 'black box' simulation of a space craft." *AI Magazine*, **1:12**, pp. 56–63.

WANG, L. and J. MENDEL (1992): "Fuzzy basis functions, orthogonal least squares learning and universal approximators." *IEEE Transactions on Systems, Man and Cybernetics*, **3:5**.

WANG, L.-X. (1992): "Stable adaptive fuzzy control of nonlinear systems." In *Proceedings of the 31st Conference of Decision and Control*, pp. 2511–2516, Tuscon, Arizona.

ZADEH, L. (1965): "Fuzzy Sets." *Information and Control*, **No 8**, pp. 338–353.

A REAL-TIME EXPERT SYSTEM FOR PROCESS SUPERVISION AND ITS APPLICATION IN PULP INDUSTRY

J. BARKLUND*, G. BOHLIN** and P. MILDNER*

* *Uppsala University, Computing Science Department, Box 311, S-751 05 Uppsala, Sweden*
** *SCA Research AB, Box 3054, S-850 03 Sundsvall, Sweden*

Abstract. We have developed an expert system for continuous supervision of part of a pulp process. This is part of a prototype of KE 2000, a workstation for process industry developed by the Computing Science Department of Uppsala University and SCA Research AB. The system ran continuously in an operator control room at the SCA Östrand pulp plant from early 1991 to mid 1993. In this paper we evaluate the experiences gained so far.

Keywords. Alarm systems, Artificial intelligence, Expert systems, Knowledge Engineering, Man-machine systems, Process control, Pulp industry

1. BACKGROUND

The underlying motivation for the work described herein is to change the role of operators in process industry. Currently much of their time is spent passively monitoring instruments and screens. Our objective is to give the operators the opportunity to have a more active role in the supervision and maintenance of the process. Some examples of such activities are:

- Self-paced continuous education.
- Investigating unexpected process behaviour.
- Following up measures taken for improving process behaviour.
- Testing and further developing one's knowledge about the process.

In order to make such activities possible we must provide certain resources, such as:

- More sophisticated supervision systems to take over much of the routine monitoring, potentially also part of control.
- Course material, possibly, and maybe even preferably, computer-based.
- Tools for analysing process conditions and behaviour.
- Simulators, etc., for experimental activities.

We have, in collaboration with SCA Research AB, developed the KE 2000 concept. It is intended as a workstation for process industry, capable of providing certain services in a uniform environment. The current prototype for KE 2000 includes an expert system for process monitoring, software for self-studies, a simple simulator and standard tools such as spreadsheet and word processing programs.

2. WHY USING KBS TECHNIQUES

The role of the expert system in KE 2000 is not only that of automatic process supervision. Given appropriate tools, an expert system can also be used to test and deepen one's understanding of the problem domain. Briefly, the procedure is a repetition of the following steps.

- Formalising one's intuitive understanding of the process as rules in a knowledge base.
- Comparing the predictions of this knowledge base using (recorded, simulated, or live) process data, against those of other knowledge bases and human judgements.

Moreover, it is expected that a knowledge base can serve the purpose of preserving valuable knowledge that would otherwise be lost when experienced operators leave. We predict that studies of other operators' formalised knowledge will cause fruitful interaction and further advances. Of course such a repository of knowledge is also valuable when educating new operators.

To summarise, including an expert system component in KE 2000 is intended to give the following results.

- Automatic detection of routine process disturbances. A future possibility is to provide some means for automatic corrections as well.
- Advance warning of infrequent or serious process disturbances.
- Preservation of knowledge about the process.
- Preservation of discussion and stimulation of the operators' interest in the process.

- Being a tool for operators for formalising and verifying their own knowledge about the process.

Complete accomplishment of the first three goals for a non-trivial domain would have required substantial resources that were not available for developing the KE 2000 prototype. For the prototype it was not considered worthwhile to build a production quality knowledge base. As a consequence, the prototype was primarily intended to achieve the last two goals above.

3. OVERVIEW OF KE 2000

3.1 Requirements

In order to be useful in this kind of project, it was required that KE 2000 satisfy the following requirements.

- High reliability in terms of continuous operation.
- Extensibility.
- Ease of operation for all its intended users.

3.2 Services

Some examples of services to be provided by KE 2000 are:

- automatic process supervision,
- computer aided instruction,
- summary of the process state,
- simulation,
- statistical analysis of data,
- spreadsheets, and
- word processing.

3.3 Design Considerations

There existed no single tool that could offer the services mentioned above. It was therefore decided to let the main functions of KE 2000 be provided by distinct programs, through a consistent user interface. It was therefore possible to address each function separately.

4. THE EXPERT SYSTEM

4.1 Design Principles

For the expert subsystem there exist a number of tools in the market, most notably G2 (Moore *et al.* 1988), a further development of PICON (LISP Machines Inc. 1986). Some of our experiences with using PICON have been reported earlier (Hjort and Pavek 1988). The choice was between using G2 and developing our own tool for automatic supervision. Some circumstances influencing our decision were:

- The intended use of the expert system requires that operators are able to easily develop and modify knowledge bases. It is necessary that the knowledge language is simple and not in a foreign language.
- It was also required that operators be able to write and test their own knowledge bases. It must be possible to run and modify one or many alternative knowledge bases simultaneously with a distinguished "official" main knowledge base without risk of confusing their output, nor of inadvertently modifying the official one.
- For reasons listed above we preferred to use Macintosh workstations and the high memory requirements, etc., of G2 made this impossible. Currently G2 is not even available on the Macintosh.

Our choice was consequently to develop our own tool set for building the expert subsystem, according to the principles described below.

Modularity. Developing a tool set for expert systems from scratch is an enormous effort and it could not possibly have been done within the duration of the project. We therefore decided to use existing tools wherever possible and to link all subprograms, or modules, with a simple bytestream protocol.

This modular approach has several important advantages, for example:

- The development time for the tool set can be greatly reduced by enhancing commercially available tools with communication facilities instead of developing new tools.
- A module can easily be replaced by a new version. This can be done without any change at all to the other modules as long as the new version of the module follows the established protocols when communicating with the other modules. Provided that this is true the new version does not even have to be implemented using the same tool as the old version.
- This, in turn, means that the system can continuously be upgraded with new technology for each module as soon as it becomes available.
- Provided that the communication facilities support communication within the same machine as well as between distinct machines, it is possible to distribute the modules and thus the expert system over several machines. For a discussion of such distributed systems, see below.

Distributed system. There are several good reasons for choosing a communication system that allows modules on distinct machines to communicate. Some advantages with spreading modules over several machines are:

- The system can be made more robust, since most or all of the functionality of the system can be retained when one machine is stopped. In KE 2000, for example, it is very important that the expert system keeps running even if some other subsystem fails and possibly causes a machine to stop temporarily.
- A related issue is that of redundant functionality. In order to increase robustness it is possible to duplicate modules and let them run on distinct machines. In this way the failure of one machine need not affect at all the functionality of the whole system.
- More computation power can be added when necessary by adding new machines and moving modules to run on distinct machines.
- The user needs only see a machine that runs user interface modules. By duplicating these machines, many users at different locations can work with KE 2000 simultaneously and access the same information, without significantly decreasing performance.
- It is possible to monitor the system from a remote location.

Logic programming. We decided to base the rule language on Horn clauses because they have simple semantics and it is known how to run them efficiently on computers as exemplified by numerous Prolog implementations.

The Computing Science Department has long experience of Prolog implementation. We therefore chose to enhance the in-house Tricia Prolog system (1988) with communication facilities and use it for building the inference machine modules.

Interface to Existing Systems. KE 2000 can be used in environment with quite varying and heterogeneous existing instrumentation and computer systems. The presentation of conclusions, etc., should be uniform and independent of the data sources used. We have therefore decided to let the interface of the expert system to such equipment be a separate module in order to minimize the need for customisation.

4.2 Operation

The expert system is based on rules, each having a set of conditions and a set of conclusions. A conclusion of a rule may be used as a condition in another rule, but it may also be reported to alert the operators of a present or future process

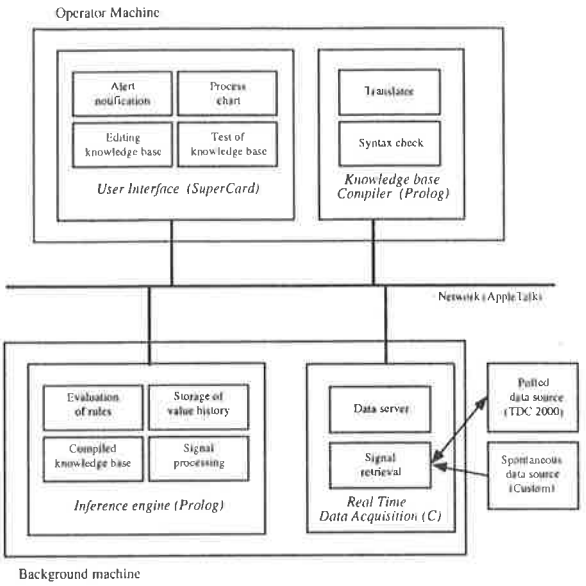


Fig. 1. Overview of the system

condition.[1] The rules are written in a high-level (natural) language and must be translated into a runnable version before being loaded into the inference machine. The installed prototype currently uses Swedish as the base language.

New data is continuously brought into the system. The inference machine is invoked periodically and computes the conclusions which are valid given the current set of data. This set of conclusions is compared with the previously valid set and any differences are sent to the user interface for notification.

4.3 Architectural Overview

As was noted before, the system consists of communicating modules, distributed on several computers. Fig. 1 is an overview of the system.

The software components have been built using the programming languages Prolog[2], Think C[3] and SuperCard[4]. More details are available elsewhere (Mildner and Barklund 1992).

Graphical User Interface. A user interface module provides the following functions.

[1]These process conditions typically indicate problems with the process but could in principle be any events important enough to report.

[2]Tricia Prolog follows de facto standards for Prolog, but has been extended with network communication. It was developed at the Computing Science Department of Uppsala University (Barklund and Millroth 1988). It is available by anonymous ftp from `ftp.csd.uu.se`.

[3]Think C by Symantec is a development system for C programs.

[4]SuperCard by Aldus is a specialised environment for building user interfaces. Our experience is that it is a useful tool for rapid development of prototypes but the resulting programs are too large and slow to be of production quality.

- Presenting to the user conclusions, and information about conclusions, reported by the inference engine.
- Displaying and providing interaction with a process schema that details the flow of substances, energy, etc., in the process.
- Displaying data from the process, such as values from sensors monitoring a part of the process.
- Displaying charts plotting current and past values of sensors and other data as a function of time.
- Providing an editor for the information in knowledge bases.
- Displaying an overview of the status of the process.

One user interface (presumably in the operator control room) is defined as the primary interface and is the only one allowed to perform certain operations, e.g., installing a new main knowledge base. Any number of observation-only user interface modules can be connected. It would also be possible to have specialised user interfaces for certain categories of users, e.g., a small alert-only interface.

Real Time Data Acquisition. A real time module is responsible for timely retrieval of data from the process being monitored. It periodically sends newly retrieved values to those inference engines subscribing to its services.

There are currently two kinds of data sources, those that the real time module has to poll for data, and those that are input-only. The latter are used for data provided by any external module with which the real time module cannot, or is not allowed to, have two-way communication.

Data items provided by the real time module (as opposed to named expressions, which are computed by an inference engine) are referred to as primitive data items.

Knowledge Bases. A knowledge base contains formal and informal knowledge about some part, or related parts, of a process. The formalised knowledge in a knowledge base includes rules, written in a subset of Swedish or some other natural language, which encode knowledge and beliefs about the process. The rules are translated into a runnable form by a knowledge base compiler and are subsequently given to an inference engine. The inference engine computes indications of "interesting" process states based on the rules. The language is described elsewhere (briefly by Mildner and Barklund (1992); extensively in Swedish by Barklund (1991))

Knowledge bases also contain informal knowledge which documents the process states, indications and rules in the knowledge base. The informal knowledge is not used directly by the expert system but is accessible to the user.

Knowledge Base Compilers. Our compiler takes as input a knowledge base with natural language syntax, parses, analyses, and outputs it in a form suitable as input to the inference engine (Mildner and Barklund 1992).

Inference Engines. An inference engine loads the runnable version of a knowledge base. It periodically receives data from the process and computes all current indications. Indications that have become valid, or have ceased to be valid, are reported to all "listening" user interfaces if the indications are so marked in the knowledge base.

An inference module also collects and provides the data needed for user interfaces to plot how primitive and derived values have changed over time.

A system contains more than one inference engine when several knowledge bases are used simultaneously. This is the case when testing a new knowledge base and potentially also when a knowledge base is split into parts, e.g., belonging to different parts of the supervised process.

4.4 Communication

The high level communication protocol is asynchronous (i.e., a module always proceeds immediately after sending a message, never waiting for a reply). This is more difficult to implement than synchronous communication but allows the operation of each module to be independent of the response time of any connected module (which might be virtually infinite if a module is unreachable).

The protocol is text-based, making it easy to monitor and debug the communication. A byte-stream protocol was chosen as underlying protocol also because it is a type of protocol available on most platforms and from most languages.

The messages sent and received by each module have been decided for each pair of communicating modules separately.

Communication in modular expert systems is discussed more thoroughly elsewhere (Wünsche 1990, Barklund *et al.* 1991, Barklund *et al.* 1992).

5. ARCHITECTURE OF THE PROTOTYPE

5.1 Platform

Our main criteria for choosing hardware were:

- Existence of a large software base and good program development tools.
- Ease of operation also for users with little or no previous experience of computers.
- Availability of software for rapid development of sophisticated user interfaces.
- Existence of national language support both in system and application software.
- Simplicity of network operation.
- Stable operation and good support.

Our earlier experiments used LISP Machines—workstations built to run symbolic processing efficiently. Our experience was that these machines were lacking in several of the criteria above.

We chose Macintosh computers because they were adequate according to our criteria and superior in some respects, notably the user interface, compared with the alternatives: PCs or UNIX workstations.

The absence of a pre-emptive multitasking operating system on the Macintosh computers was a source of difficulties. However, there were also advantages with having only Macintosh computers, for example, the AppleTalk network protocol provided on top of the EtherTalk network provided useful high level facilities.

5.2 Distributed System

Distributed systems were advocated above. In particular, it is advisable to run critical parts of the expert system on computers not affected by user operations (e.g., running computationally intensive or unsafe tools). We will call these computers background machines because the users have no need to physically access them. The operator machines can then be restarted at will, also with no effect on the background machines and thus on critical programs.

5.3 Configuration

The prototype system has one background machine (see Fig. 1) and can handle up to three operator machines.[5] Each of these machines is an Apple Macintosh II computer. The machines are connected by an Apple EtherTalk network. Using fast modems and bridging software, the network can include machines arbitrarily far apart.[6]

Background machines have no direct interaction with operators. An alternative would thus have been to use, e.g., a UNIX workstation as background machine while still having Macintosh operator machines.

6. CURRENT STATUS

The KE 2000 prototype has been installed at the SCA Östrand pulp plant. Engineers and operators at the plant have been developing a substantial knowledge base for the prototype. But at the end of 1993, the process and the basic process instrumentation were changed. The knowledge base in the KE 2000 expert system became obsolete and had to be disconnected. The expert system has not yet been restarted. One reason is that work force at the mill became significantly reduced.

We are now continuing our work within a project funded by the Swedish National Board for Technical and Industrial Development (NUTEK)[7]. In this project we will, among other things, develop more powerful languages for expressing various knowledge about an industrial, or other real-world, process. One of our approaches is to provide means for expressing, more directly, knowledge about causal connections between process entities.

7. EFFORT

KE 2000 is the latest result of a collaboration between the Computing Science Department of Uppsala University and SCA Research AB which has been ongoing for seven years. It is therefore difficult to assess exactly the total effort spent on KE 2000.

However, we estimate that we have spent what amounts to four person-years on developing the expert system part of KE 2000, including its user interface.

Unfortunately, due to factors external to this project, the engineers and operators at the SCA Östrand plant were not able to allocate enough time for developing a high-quality initial knowledge base. As a consequence, the output from the expert system is not as good as we expect that it could be.

[5] Three operator machines was sufficient for testing the prototype but a marginal further effort would be sufficient to handle an arbitrary number of operator machines.

[6] This facility was particularly useful during the development of the system. It has been possible to investigate problems with the system in Timrå and upgrading it from the Computing Science Department at Uppsala University, approximately 400 km away.

[7] Contract No. 93-3113, Intelligent Real-Time Systems.

8. EVALUATION

The system has proven very reliable in terms of up-time. The expert system have had virtually no unscheduled stops in the first eighteen months, yielding a up-time ratio close to 100%, clearly above the target of 98% for the prototype.

The KE 2000 prototype operator machine has been used eight hours per twenty-four hours (as shown by the activity logs maintained by the system). Possibly, some of this time has been spent using programs not directly related to the expert system. However, whenever the operator machine is running, the expert system user interface is active and visible on a separate screen. Therefore, if operators are using the machine, they will pay attention to output from the expert system.

It is important that the running knowledge base can be modified without stopping the expert system. This possibility has the added advantage of speeding up the process of experimenting with, and developing, a knowledge base. However, in the KE 2000 prototype, the rather slow knowledge base development subsystem turned out to be a source of irritation.

As noted above, the quality of the main knowledge base and consequently the output from the expert system has been unsatisfactory. The engineers and operators have had no problems in understanding the knowledge base and in using the language of the expert system, but developing and structuring an accurate knowledge base has proved difficult. It is important to realize that the process logic can be very complicated and not easy to catch in a few rules. The process operators and engineers should be supported by a knowledge engineer who is able who is able to introduce AI techniques. The knowledge engineer has to educate and advise the personnel, and promote the expert system techniques. This need not be a full time job in a plant. Possibly a process control engineer could be given the opportunity to study AI techniques, e.g., at university, and on part time introduce and maintain this new field in the production environment.

There is a need for continuous work with the rules of the knowledge base in order to adjust them to small or big process changes. The process engineers and the operators have to cooperate in a much more formal and basic manner than before. This can, of course, lead to many problems. The language used by engineers is not the same as the language of the operators, but rules are an acceptable language for all parties although not as precise as the process models of the engineers in mathematical form. A knowledge engineer could play a useful role also in this communication. The work with KE 2000 at the mill so far has shown that the engineers and the operators are getting closer to each other, but there is still a long way to go.

The system appears to have been successful in motivating operators to use new technology, as well as having been instrumental in providing a ground for discussions about the process. One problem is that some of the operators are not willing to participate in this kind of activity.

9. ACKNOWLEDGEMENTS

We wish to acknowledge the contributions of Lars Farm and Peter Sandström, and those of Torsten Palm, Sten-Åke Tärnlund and others at the Comp. Sci. Dept. of Uppsala Univ., to the work reported herein.

The development of the KE 2000 prototype was supported financially by the DUP group of the Swedish National Board for Technical and Industrial Development (NUTEK).

REFERENCES

Barklund, J. (1991). *KE2000 — Användarhandledning, Version 0.9.1.* Computing Science Dept., Uppsala Univ.

Barklund, J., A. Hamfelt and J. Wünsche (1991). A modular architecture for knowledge systems. In N. Cercone, F. Gardin and G. Valle (Eds.). 'Computational Intelligence, III'. Elsevier. Amsterdam. pp. 247–262.

Barklund, J., A. Hamfelt and J. Wünsche (1992). 'Building modular legal knowledge systems'. *Expert Systems with Applications* **4**, 343–353.

Barklund, J. and H. Millroth (1988). *The Basic Tricia Prolog Manual.* Computing Science Dept., Uppsala Univ.

Hjort, A. and P. Pavek (1988). Pulp Expert. In S.-G. Edlund (Ed.). 'Proc. Control Systems in the Pulp and Paper Industry'. pp. 65–68.

LISP Machines Inc. (1986). *PICON User's Guide.*

Mildner, P. and J. Barklund (1992). Design and implementation of a modular real time expert system. UPMAIL Technical Report 73. Computing Science Dept., Uppsala Univ.

Moore, R., L. B. Hawkinson, M. Levin, A. Hofman, M. David, B. Matthews and J. Allard (1988). The G2 real-time expert system. In 'Proc. Instrument Society of America Advances in Instrumentation'. Vol. 43. pp. 1625–1633.

Wünsche, J. (1990). Integrating Prolog and a multimedia environment. Master's thesis. Computing Science Dept., Uppsala Univ.

Experiences from development and operation of an operator guidance system for the blast furnace process

Carl Tivelius*, Rutger Gyllenram, Jan Olov Wikström***, Mats Hallin***

* SSAB Tunnplåt AB, S-951 88 Luleå, Sweden
**Kobolde AB, Källbacken 15, S-129 40 Hägersten, Sweden, kobolde@faksimil.se
***Mefos, S-951 88 Luleå, Sweden

Abstract. An operator guidance system has been developed and taken into operation at the No. 2 blast furnace at Swedish Steel, SSAB, in Luleå. The reasoning model expressed by the personnel and used in an early prototype had to be modified in order to make the reasoning easier to understand. The performance of the system is satisfying with suggestions of the same magnitude as the actions undertaken by the operators but often with better timing. Important issues when the system is further developed are establishing manual evaluation routines and establishing criteria for judging a suggestion on its merits.

Key Words. Expert systems; guidance systems; human factors; process control; blast furnace; steel industry

1. INTRODUCTION

The blast furnace is a countercurrent process where iron ore, slag formers, and coke is transported in skips and charged at the top of the furnace. Hot blast and pulverized coal is blown through nozzles in the lower part of the furnace. The ore is reduced to liquid iron and slag which is tapped through a tap hole near the bottom of the furnace. Figure 1 shows the principle of the process with the main reactions given. The thermal condition in the furnace is primarily determined by the ore to coke ratio and the efficiency with which the reduction work is taking place. These parameters may change without the operators knowledge with changes of furnace heat level defined as changes in hot metal quality as a result.

The time lag between a change in ore to coke ratio and a change in hot metal quality is approximately five hours and before the operator gets a new hot metal temperature and analysis it can take another one to two hours. This time lag, the complexity of the process and the large potential of savings that can be made if the process could be kept in a stable state has made the blast furnace process a topic for computer modelling since the mid sixties.

The development of knowledge based systems for blast furnace control started in Japan in the mid eighties which inspired activities at the Swedish Ironmasters Association. In 1988 a research project was initiated at the Royal Institute of Technology KTH, in Stockholm, Sweden, in collaboration with Åbo Akademi, Åbo, Finland. The project resulted in an off-line prototype which was evaluated at the No. 2 blast furnace at SSAB Tunnplåt AB in Luleå during spring 1991.

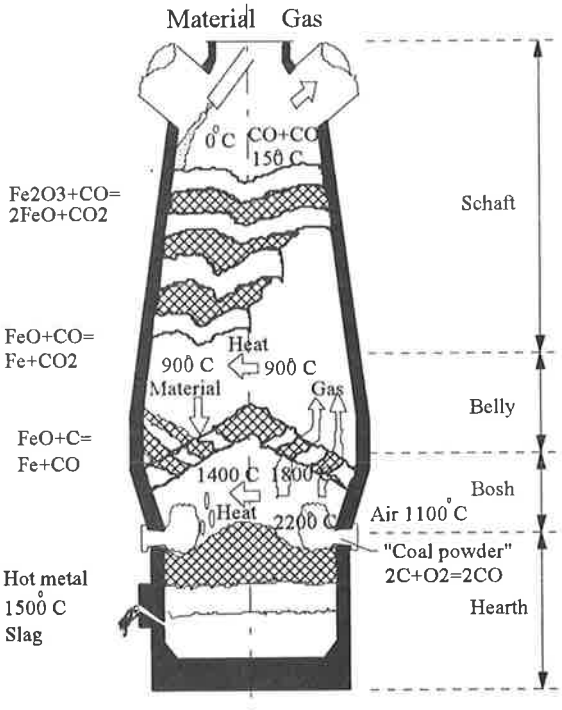


Fig. 1. The principle of the blast furnace process

After the evaluation a development project started at SSAB and an on-line system for hot metal and slag quality control named

MASMESTER was designed and programmed. This system analyses data from the process database every minute and if an action should be undertaken a suggestion is presented to the operator.

MASMESTER was taken into test operation in September 1993 and operation results are promising. The system and the operators seem to have the same opinion about what actions should be undertaken though the system is often a couple of hours ahead of the operators.

This paper aims at describing the MASMESTER reasoning mechanism, the system configuration and results from the first year of operation. The authors experiences from performing this kind of development in an industrial environment is presented.

2. PROCESS CONTROL

The blast furnace is operated as a continuous process where the amount of hot air supplied determines the production rate. The operational goal is to produce a correct amount of hot metal with lowest possible deviation in quality. Like in most metallurgical processes it is difficult to make measurements inside the reactor due to temperatures of 1000-2000 degrees C. Another problem is that the lagre amount of raw materials can not be analysed continuously since that would result in considerable costs. As a consequense of these and others conditions the hot metal quality varies with time even though the production is considered to be normal.

In daily operation this variation is continuously evaluated by the personnel. When operators decide to charge more coke or inject more coal, to correct the furnace heat level, the decision is mainly based on variations in the hot metal temperature and analysis i.e. carbon, silicon and sulphur content. If e.g. the temperature of the hot metal suddenly drops the operators must decide if this deviation should be considered to be temporary or not. In deciding about actions also the uncertainty of the temperature measurement must be taken into account. If this is the first indication of a lower heat level and the previous temperatures and anlyses indicate a historic normal heat level, no action is probably undertaken even though the deviation is not neglectable. The reason for this practice is the ability of the furnace to recover from a deviation in heat level without any operator action. The probability for the furnace to recover increases with a past high heat level and a high coke to ore ratio. But if the next piece of information recieved on the hot metal confirm a lower heat level the reason for a corrective action becomes stronger.

The core knowledge in blast furnace heat level control lies in grasping the complexity of this dynamic behaviour.

3. INITIAL DECISION MODEL

An initial decision model was developed based on interviews with blast furnace managers and researchers (Gyllenram *et al.*, 1991; Sandberg, 1992). Important input parameters, for decisions, were identified as the hot metal temperature (T) and the silicon analysis (Si). The output of the model was decided to be suggestions regarding changes of coke, extra skip of coke and moisture addition.

The model was rule based in a way which correlated well to how the personnel described the decision process. Although rules were used the model could be described as a matrix with T as ordinate and Si as abscissa. Only a few cells in the matrix contained suggestions. Surrounding cells were empty. If the input data hit an empty cell the suggestion was calculated by a linear interpolation using the values from the surrounding filled cells.

The problem of taking the process history into consideration was solved by filtering the input data with a first order filter. This smoothing of data also decreased the negative impact of uncertainty in data. The personnel also dealt with this problem by applying half of the action to begin with and later, when more data confirmed the initial judgement, applying the rest. This reasoning strategy was also adopted in the model.

The model was tested on real process data, off-line, and evaluated by the personnel. The performance was acceptable but on some occasions more data was used by the personnel than in the model. A number of other reasons, like problems with how to adjust the model for a desired output, explaining why input data resulted in a particular suggestion and how the history was considered, resulted in the decision to search for a new approach in the modelling.

4. MODIFIED DECISION MODEL

With the experiences from the initial model a new model was created (Tivelius *et al.*,1991). The model was based on the same knowledge but reformulated in order to overcome the discovered drawbacks. An extensive process evaluation indicated the possibility of introducing three reasoning modes, cold, normal and warm, when modelling the dynamics of the process.

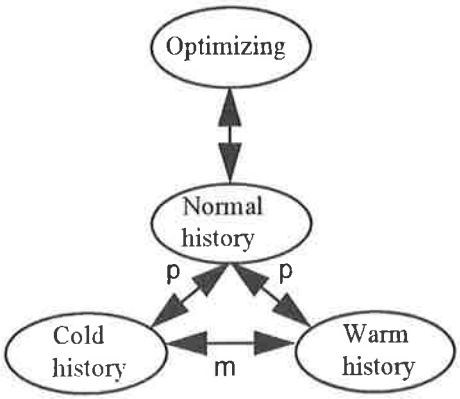


Fig. 2. Diagram showing the different reasoning modes of MASMESTER

Figure 2 describes the different reasoning modes of MASMESTER. In the model no restrictions are set on possible paths between the three modes. However, it is not likely in actual operation with a direct shift of the furnace's heat level e.g. from warm to cold. The furnace follows, according to experience, the arrows marked with a *p* but again the model may use the arrow marked with an *m*. In operation, at certain conditions, it is possible to decrease coke or coal powder although the quality of the hot metal is acceptable. This strategy falls in the optimization state which is not considered in the MASMESTER system.

The model consists of five parts, an initial group of rules for deciding the historic heat level giving the reasoning mode, three matrixes each representing one historic heat level and a group of rules for final adjustment of the proposal from the matrix.

The inital group of rules contains criterias for classification of the historic heat level as cold, normal or warm. Data used in this classification are hot metal temperature, silicon (Si), carbon (C) and sulphur (S) content.

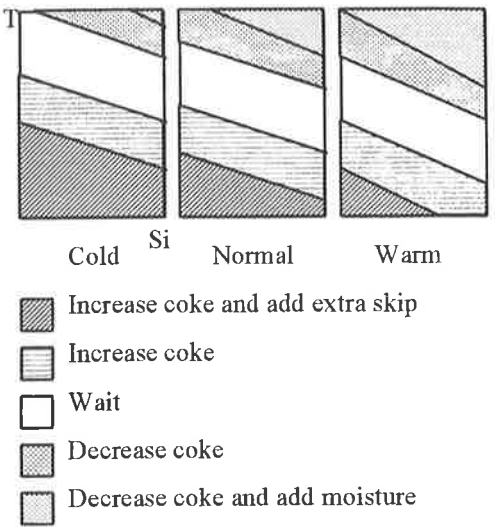


Fig. 3. Action matrixes with action profiles for different reasoning modes..

Figure 3 describes the three matrixes in a general way. As in the initial model the temperature (T) and the silicon analysis (Si) is used as heat level descriptors of the hot metal. The output of the model is represented in Figure 3 by the different grafic patterns. In every cell, 800 in every matrix, a suggestion on a corrective action is found. The type or magnitude of a suggestion for each combination of temperature and silicon differ between the three matrixes. Each matrix represents a different action profile suitable for a special heat level history. The straight forward idea in the model is that the latest data of the hot metal, T and Si, must be interpreted differently depending on the heat level history. Henceforth has the warm matrix a larger acceptance e.g. of a temperature drop than the normal matrix, see location of white areas. The underlying belief is that with a historic high heat level of the furnace, the furnace has a higher ability to recover from a temperature drop. As the heat level decreases the model will change matrix from warm to normal to cold and the corresponding suggestions will increase in magnitude.

A proposal from one of the matrixes is only preliminary. Since the rate of throughput of the furnace is 5-7 hours the proposal needs to be compared with and adjusted for already performed changes in operation. The result of this comparison is a final suggestion.

Although this was a new way of describing the reasoning mechanism it emmidiately won acceptance. A particular suggestion could be

explained without any calculations and the rules and the matrixes could quickly be altered.

5. THE FUNCTION OF MASMESTER

MASMESTER is an operator guidance system (OGS) with presentations of suggestions combined with background information. The system is integrated with the plant process relational database at SSAB. MASMESTER reads data every minute, evaluates this information, stores and presents the result of reasoning on a screen in the operator room. A number of important events such as changes in, coke, coal powder, extra skip of coke, recipe and burden calculations are detected since they are necessary in the decision making. An important feature is the modelling of the time lag associated with each event. By keeping track of each skip of charged material a time concept was introduced that is independent of different production rates.

MASMESTER is implemented on a Microvax 3100 and developed with an object orientated tool, a relational database and with C as 3GL. A character based user interface based on Windows technology has been used.

As the structure of MASMESTER is distributed, data source, reasoning and presentation at three different locations, disturbances are unevitable. A hierarchy of programs for supervision with inter process message passing has successfully been developed and applied.

6. RESULTS OF OPERATION

MASMESTER is today in continious operation and interaction with operators. A comparison of MASMESTER's suggestions and actions performed by the operators, over a period of three monts, is presented in Table 1. The base of the comparison is the performed corrective actions and the result of these. An action is considered as correct only if the change was favourable in a longer time perspective.

MASMESTER has as an average, during the 90 days, presented little less than two suggestions a day. In actual operation, regardless of the average value, many days may pass without any actions or proposals.

The suggestions from MASMESTER have been put into 4 categories. Category 1 holds the 94 best proposals from MASMESTER. Both size and direction were equal to the operators' choice but each proposal was given at a time when no action was taken. This time varies from minutes to several hours.

Performed changes of operation by the operators	84
Total number of advice from MASMESTER	174
Correct size, direction and earlier	94
Correct direction, earlier but sligtly too large	67
Preliminary direction correct, final direction wrong	11
Unrealistic	2

Table 1. Operational results from three months of operation.

In most cases the operators waited longer than MASMESTER, long enough to recieve a second or third proposal from the system, before deciding on a corrective action. Sometimes these advice were too large. Category 2 hold all 67 proposals of this kind.

In situations where the operators have decreased the coke to ore ratio, because of the hot metal being too warm, the matrix model has also suggested a decrease. If the actual performed decrease is larger than the suggested MASMESTER will as a final suggestion propose an increase. These increases were not necessary. Category 3 holds 11 suggestions of this type.

At two accasions the set point of the coal powder injection was drastically decreased. Since the operators knew this was only temporary no compensation was made. MASMESTER proposed an increase of the coke to compensate for the loss of coal powder. In a longer time perspective these suggestions would have been correct but not in a shorter time perspective.

An example of the operators' interaction with MASMESTER is shown in Figure 4. The system has come to the conclusion that the heat level has to be adjusted and a suggestion is proposed.

The suggestion is divided into three main parts. At the top part the preliminary tactic and proposal from the matrix model is presented. In the middle part it is possible to see if the system has detected any control actions and if these still

may effect the heat level. At the bottom is the final tactic and suggestion presented.

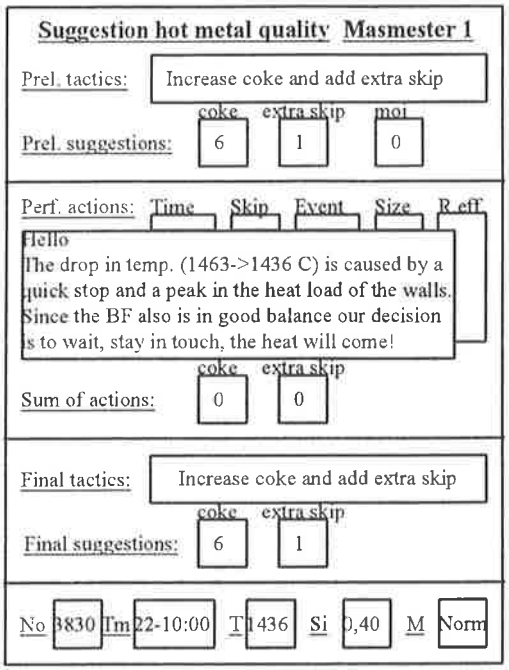


Fig. 4. An advice regarding heat level control with an operator comment in a pop-up window.

The operator has given a comment to this advice because he does not intend to follow the recommendation. Acccording to the rules in MASMESTER this deviation is too large and the furnace is not believed to recover. A few hours after this proposal was given the heat level returns to a normal level. The operator seems to have made the corrrect decision at this point. After some time, though, the heat level drops again. The evaluation classified this advice as slightly too large but correct regarding time and direction.

7. ACHIEVING ACCEPTANCE FROM THE ORGANIZATION

MASMESTER is a guidance system and not a closed loop control system. The operators and foremen are always fully responsible for all actions undertaken and as a consequence they will only accept and use a system that they either fully understand or is always correct. Since events of importance for the decisionmaking may occur which do not show in the database the system may err and therefore the first alternative mentioned above must be fulfilled.

The reasoning model in MASMESTER is based on the heuristic knowledge of the operators, foremen and managers at the blast furnace. They are all skilled craftsmen and the system development team has dealt with their main resource, their knowledge and experience of the process and the furnace equipment. Although the willingness of the personnel to participate in the development process and to share knowledge and ideas has been outstanding throughout the project the team has been forced to cope with some major problems concerning knowledge acquisition and system accptance.

The first problem is that when discussing the furnace operation it has been easy to confuse how things are actually done and how things should be done according to instructions or the opinion of the personnel. Reasoning often seem to start from some sort of ideal situation which is very seldom prevalent.

Another problem is that although a vast amount of data is available about the furnace only a very limited amount of this data is suitable for the day to day control of the furnace. This is due to the complexity of the process but nevertheless it is annoying for a craftsman to accept that measured values from the process can either be interpreted in so many ways that they cannot be used or do not give any additional information about the process or both.

An interesting example of the latter is the ad hoc attribution of phenomena or process states to certain data or patterns in data in order to explain why a suggested action is not undertaken. The main reason may be a reluctance to make changes and a wish for more indications that the action is really necessary.

An important problem when you are dealing with a system developed for decisionmaking under uncertainty is that there always will be occurances where a suggestion given by the system does not show out to be successful. Since MASMESTER seem to be more eager than the operators to perform changes it is reasonable to believe that future evaluations have to judge between costs for unnecessary actions and costs for late actions. The psychological dilemma is that a singel action that has to be nulled by a counteraction is easily remembered as an event but a practice of waiting a considerable longer time before the actions will not be remembered since the actions are correct when they at last are performed.

In the future development of MASMESTER effort is put into finding objective measures to evaluate the operation and to judge the suggestions and actions on their merits. The challenge now is to establish a control practice where a correct behaviour is to follow the suggestions unless hazardous and then improving the practice step by step.

This effort is supported by the comittment of SSAB to further development of quality assurance systems making control room practice a management issue. Another factor supporting further development is the possibility to delegate most of the decisionmaking about day to day heat level control from the foremen to the operators.

8. FURTHER R&D

This project is an example of an academic research project that has found its way out to the industry. Although MASMESTER today is a concern of SSAB the research and development activities regarding the use of information technology in process industry contiues in the project "Intelligent Alarm Management" at SSAB, Mefos and the University of Luleå with the SSAB blast furnaces as the target process. Examples of activities in this project are development of FDI-methods for asynchronous motors, metallurgical prediction models using neural nets and human factor studies.

A major experience that is carried from the MASMESTER project to Intelligent Alarm Management is the importance of industrial engagement in academic research projects. The reason is threefold. Learning about the methods used at the university and understanding how to use them in an industrial environment takes time as well as teaching the researchers about the industrial realities and providing them with data and expertness. Finally preparing the organization for a new technology and new concepts must start at about the same time as the research projects.

9. CONCLUSIONS

MASMESTER today is a system that can compete with a skilled operator or foreman in controlling the heat level of the furnace. Sufficient knowledge combined with the computer systems admirable ability to be alert even during the worst hours of the night shift makes MASMESTER stand a good chance of becoming an accepted and desired working tool.

With the introduction of MASMESTER several other operation guidance system projects followed at SSAB Luleå for other segments of the production route. No doubt the existence of a pilot project capable of breaking ice has a great importance to free creativity and promotes activities in an organization.

10. REFERENCES

Gyllenram, R. J. Sandberg, H. Saxén, L. Karilainen (1991). A prototype knowledge-based system for blast furnace operational guidance. AIME 1991 Iron making conference proceeding. Washington. U.S.A.

Sandberg, J. (1992). Expert system based blast furnace control. Lic. thesis. Dept. of Production Technology , Mining and Steel Industry, Royal Institute of Technology, Sweden.

Tivelius, C. and C. Petersohn (1991). Control of the blast furnace using expert system techniques - establishing and programming of decision rules for different states of the process. Master thesis. Dept. of Production Technology , Mining and Steel Industry, Royal Institute of Technology, Sweden. In Swedish.

Fuzzy Persistence in Process Protection

Howard P. Rosenof, PE
Manager, Process and Utilities Marketing
Gensym Corporation
CambridgePark Drive
Cambridge, Massachusetts 02140, USA

Abstract. This paper proposes "fuzzy persistence", whereby a fuzzy logic network used for process protection can impose a delay time which depends on the severity of the off-normal situation. This technique is familiar in electric circuit protection, in which a circuit breaker or fuse responds more quickly to clear a more severe fault. Here fuzzy persistence is implemented as an extension of a protection system for a batch chemical reactor.

Key Words. alarm systems; artificial intelligence; expert systems; fuzzy control; batch process control

1. INTRODUCTION

Basic protection of process units is provided by interlocks, the "lowest" level of control[1]. Interlocks typically serve to prevent the process unit from reaching an unsafe state, or when an unsafe state is reached, they alert plant personnel to this fact and attempt to return the unit to safe operation.

Interlock systems often deal in absolutes. A temperature measurement slightly below the threshold for alarming or emergency action, causes no response; but a value at or slightly above the threshold results in the maximum response possible. The success of such an arrangement is strongly dependent upon the proper selection of thresholds - if they are too loose dangerous conditions may fail to be detected, but if they are too tight, operators may simply disconnect the interlock system to prevent recurring false alarms or spurious automated safety actions. Further, in the case of batch process units, excursions representing a genuine approach to unsafe conditions must be distinguished from the routine variations inherent in batch processing.

[1]Rosenof & Ghosh, Batch Process Automation: Theory & Practice, Van Nostrand Reinhold Co., New York, NY, 1977

Fuzzy logic is proposed as a remedy for this problem. A statement like "The reactor is approaching an unsafe condition" or "The reactor has achieved an unsafe condition" may be assigned a truth value, by convention a number from 0.0 (completely false) to 1.0 (completely true). A complete reactor protection system would retain the simple and absolute interlocks discussed above, with thresholds set to avoid false alarming and actions, and would add fuzzy logic calculations to reliably warn of the impending action of such an interlock. In the discussion that follows it is assumed that the lowest-level interlocks, not otherwise considered here, are always active.

1.1. Batch Temperature Monitoring

Batch exothermic reactions may be subject to "thermal runaway", a condition in which heat evolves more rapidly than it can be removed by the reactor's cooling system. This is a positive feedback situation. The evolving heat raises the reaction rate, which further increases temperature, until the reactants are exhausted. Within limits the reactor may be self-regulating (increasing temperature leads to increased heat transfer to the cooling medium) but under some conditions the evolved heat can exceed maximum cooling capacity. Much of the reactor's interlock

arrangements are in place to predict and if possible prevent or suppress thermal runaway when this condition is not desired.
Several measurements are available for this, including a simple monitoring of the control signal attempting to set the position of the coolant control valve - a high demand for cooling implies that the limits of cooling are being approached. The most obvious measurement is temperature, of course. With simple interlock logic, when the temperature exceeds its threshold an alarm or emergency shutdown action is initiated.

However, temperature alone is not a reliable predictor of this condition. During the exothermic reaction a comparatively low temperature may be dangerous, but once the reaction is complete a higher temperature may be safely used. It may practical to include a "maximum temperature" specification in the recipe or formula for each phase of production, but this technique is subject to errors, errors that might not manifest themselves until a dangerous condition is reached.

Rate of change of temperature can indicate thermal runaway, since thermal runaway is accompanied by a rise in temperature. However this calculated variable if used alone can also lead to false results, such as by alarming on the increase in temperature that may be experienced by a cold reactor charged with warmed material. Therefore, when rate of change is used it is normally applied only when the reactor temperature is above a specified value. The calculated rate of change may also have to be adjusted to remove the effects of the reactor's response to deliberate changes in its temperature setpoint.

If the temperature is slightly below above the alarm threshold, no value of rate of change will initiate the alarm. When the temperature is slightly above the threshold, the alarm logic's sensitivity to rate of change is the same as when the temperature is well above.

This alarm logic can be improved with fuzzy logic. First the statements "the reactor is hot" and "the rate of change of reactor temperature is high" are represented as fuzzy logic statements, each with truth a value between 0.0 to 1.0 inclusive at any one time. The truth value of the first statement, for example, may be 0.0 for temperatures below 80 C and rise monotonically (typically, but not necessarily, linearly) to 1.0 at 120 C. In words, this means that the statement "the reactor is hot" is completely false through 80 C, is increasingly true (and decreasingly false) through 120 C, and completely true at and above 120 C. Similarly, the statement "the temperature of the reactor is increasing quickly" may be completely false below 0.5 C/min and completely true above 1.5 C/min.

The two signals are then combined to characterize reactor conditions in one value, and this value compared to a threshold. Fuzzy logic offers two possibilities - the fuzzy OR which, by convention, transmits the most true input to its output, and the fuzzy AND, which transmits the least true input to its output. (If the only possible input values are 1.0 and 0.0, these fuzzy gates act as conventional OR and AND gates.) The AND function is not desirable because failure of one input (for example, so that its output were stuck at "completely false") could prevent the logic network from achieving an alarm state. OR logic would work better, but is still not ideal, because any value of one input below the threshold would leave the network with the same sensitivity to the other input.

A better logic function would be one in which a relatively high (i.e., more true) value of one input would result in an increased sensitivity to the other input: If the temperature of the reactor is low a relatively high rate of change would be required to institute the alarm, but if the temperature is high, only a small rate of change would be required for alarming. The arithmetic average of the two variables would fulfill this function, but the author has suggested in a previous paper [2] that the Euclidean mean, the square root of the sum of the squares of the truth values, is a particularly good choice. Thus the function

$$C(T_1, T_2) = (1/\sqrt{2}) * \sqrt{(T_1^2) + (T_2^2)} \quad (1.0)$$

takes on values in the range of 0.0 to 1.0 when the truth values are within the same limits.

The threshold value, that is the value of C (T_1, T_2) which causes an alarm action to occur, should be set low enough so that if one of the input values fails at 0.0, the alarm action should occur when the other value is less than 1.0. The threshold value should be less than

---

[2] Rosenof, H.P. "A Prototype Fuzzy Controller for an On-Off Load", Engineering Society of Detroit International Programmable Controls Conference, Detroit, MI, April 8, 1992

$$C\ (0.0, 1.0) = (1/\sqrt{2}) = 0.707 \qquad (1.1)$$

Fig. 1 shows alarm regions for various thresholds, and Fig. 2 shows the threshold required for a given truth value for one input, when the other input is failed at 0.0. Combining two inputs, it is necessary to choose between the possibility of a false alarm or safety actuation if one of the inputs fails high, and the possible failure to take a safety action if one of the inputs fails low. The former is normally preferable.

1.2. Fuzzy Persistence

It is common to filter inputs to an alarm mechanism, so that the alarm does not respond to short-term changes which may be the result of electrical or mechanical noise rather than a genuine alarm condition. In discrete alarm systems, this filtering is often achieved by requiring persistence, that is requiring the alarm condition to be asserted for a specified time before the alarm is communicated to the operators.

"Fuzzy persistence" recognizes the severity of an alarm condition and imposes a relatively long persistence requirement if the condition is not severe. The persistence requirement decreases with increasing severity of the alarm condition.

Returning to the batch temperature monitoring example, assume that at a combined truth value of $C_H = 0.7$ the alarm or safety actuation should be immediate. This will permit a prompt response to an extreme high-temperature or high-rate condition if the other sensor channel fails low. At the other extreme, there should never be a response for $C_L < C\ (0.5,0.5)$ since this is a no-alarm condition. This corresponds to

$$C\ (0.5, 0.5) = 0.5 \qquad (1.2)$$

In use, such an alarm system would begin accumulating time T once C exceeded 0.5. When T exceeds the threshold corresponding to the current value of C, the alarm or safety actuation is initiated.

2. IMPLEMENTATION OF FUZZY PERSISTENCE

The three-term version of eq. 1.0 is

$$C\ (T_1, T_2, T_3) = (1/\sqrt{3}) * \sqrt{(T_1^2)+(T_2^2)+(T_3^2)} \qquad (2.0)$$

T_3 represents accumulated time, and is the fuzzy truth value of the statement "the possible fault condition has persisted for a significant duration". **T** is the accumulated time after which T_3 should equal 1.0 when $C\ (T_1, T_2)$ exceeds 0.5. The value of T_3 increases from 0.0, starting from the time that $C\ (T1, T2) \geq 0.5$, up to 1.0, after **T** seconds. Thus

$$T_3 = \min\ (1.0, t/\mathbf{T}) \qquad (2.1)$$

where t is the accumulated time during which $C\ (T_1, T_2)$ has exceeded 0.5.

With $T_3 = 1.0$ and $C\ (T_1, T_2) = C\ (0.0, 1.0) = 0.7$ (to avoid a failure to function in case one sensor channel fails low),

$$C\ (0.0, 1.0, 1.0) = (1/\sqrt{3}) * \sqrt{(0.0^2)+(1.0^2)+(1.0^2)} = \sqrt{2}/\sqrt{3} = 0.82 \qquad (2.2)$$

To determine the time delay associated with given values of T_1 and T_2, it is necessary to solve eq. 2.0 for T_3 when $C\ (T_1, T_2, T_3) = \sqrt{2}/\sqrt{3}$

$$(1/\sqrt{3}) * \sqrt{(T_1^2)+(T_2^2)+(T_3^2)} = \sqrt{2}/\sqrt{3} \qquad (2.3)$$

or

$$T_3 = \sqrt{\frac{2}{T_1^2+T_2^2}} \qquad (2.4)$$

Solving eq. 2.4 for $t/\mathbf{T}$ when $T_1 = 1.0$ and $T_2 = 0.0$ yields

$$t/\mathbf{T} = \sqrt{2} \text{ or } t = \mathbf{T}\sqrt{2}. \qquad (2.5)$$

Therefore if it is desired that a delay of 14.14 seconds be experienced when $T_1 = 1.0$ and $T_2 = 0.0$ then a delay of 10 seconds will be experienced when $T_1 = T_2 = 1.0$. This relationship is shown in Fig. 3.

The relationship can be exaggerated by substituting for eq. 2.1

$$T_3 = \max\ (1.0, (t/\mathbf{T})^n) \qquad (2.6)$$

where n>1. This will result in smaller values of T_3 when t is small. For n=2, Eq. 2.4 becomes

$$T_3 = \sqrt[4]{\frac{2}{T_1^2+T_2^2}} \qquad (2.7)$$

For example, eq. 2.1 yields $T_3 = 0.5$ for $t/\mathbf{T} = 0.5$, but for n=2 in eq. 2.6, $T_3 = 0.25$. This relationship is shown in Fig. 4. Between Figs. 3 and 4, twice as much time is required to actuate the alarm condition when $T_1 = T_2 = 0.5$.

3. CONCLUSION

Fuzzy persistence is proposed as a means to reduce false alarming while responding more quickly to a severe problem. Fuzzy persistence is not proposed to replace conventional safety interlocks; rather to work with them. Time sensitivity can be changed by manipulating a term in the equation with which persistence is calculated.

4. ACKNOWLEDGMENT

The author gratefully acknowledges the assistance of Dr. Gregory M. Stanley of Gensym Corporation in preparing this paper.

Fig 1: Net truth values in accordance with the Euclidean mean

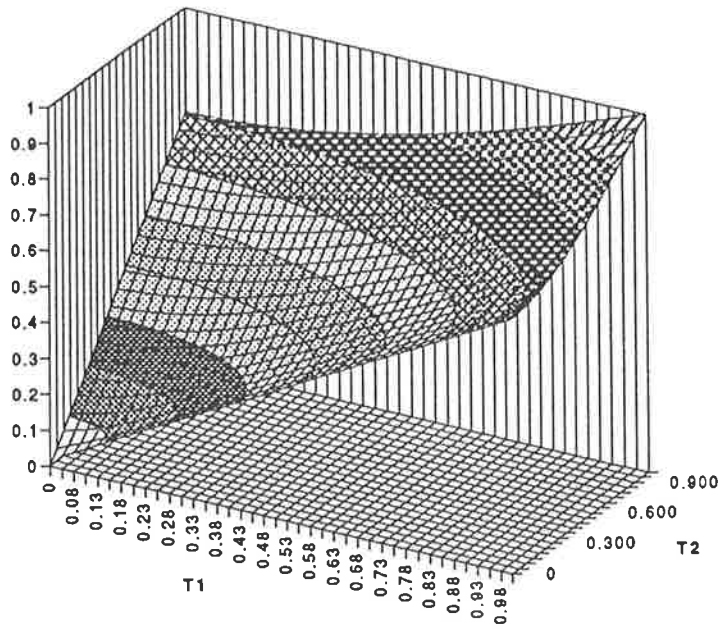


Fig. 2: Threshold value for actuation as a function of the truth value of the functioning input, when the other input has failed low

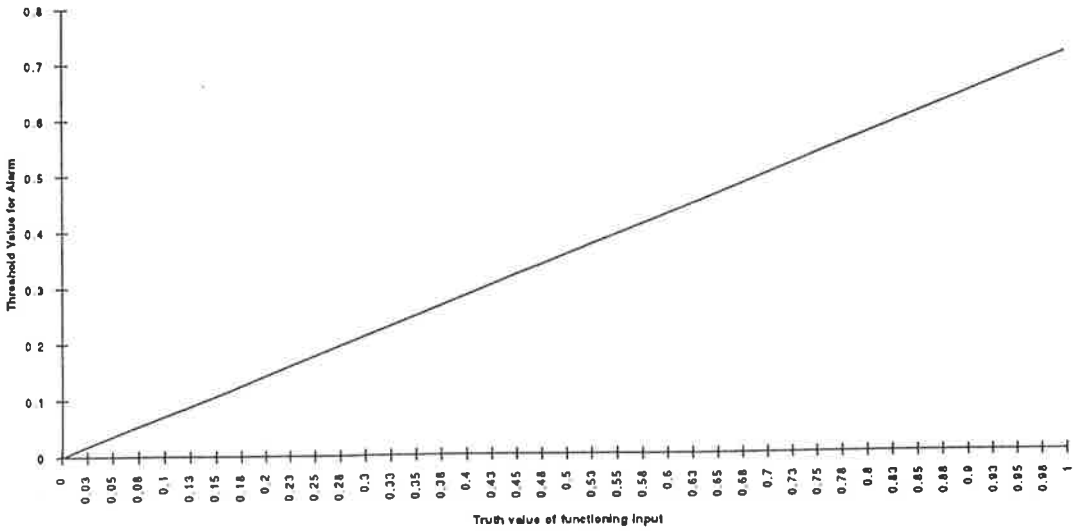


Fig. 3: Normalized time required to initiate alarm action

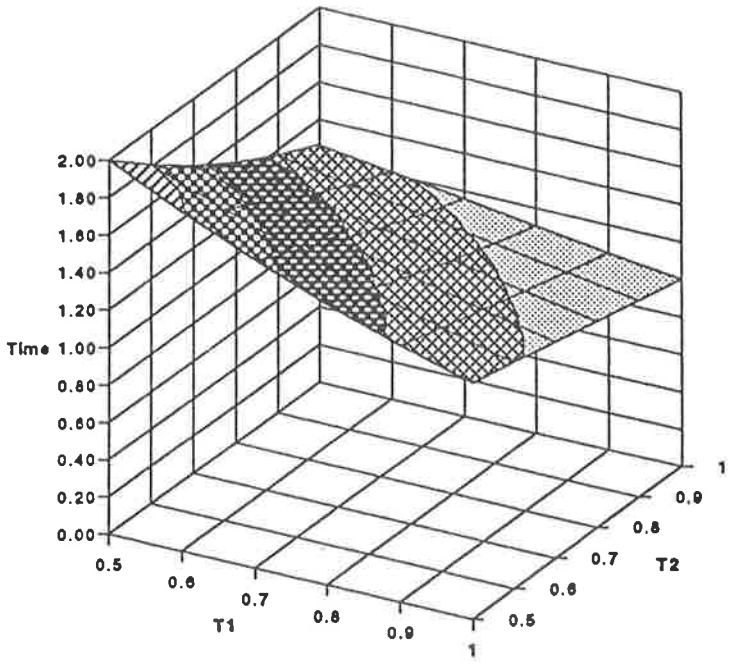
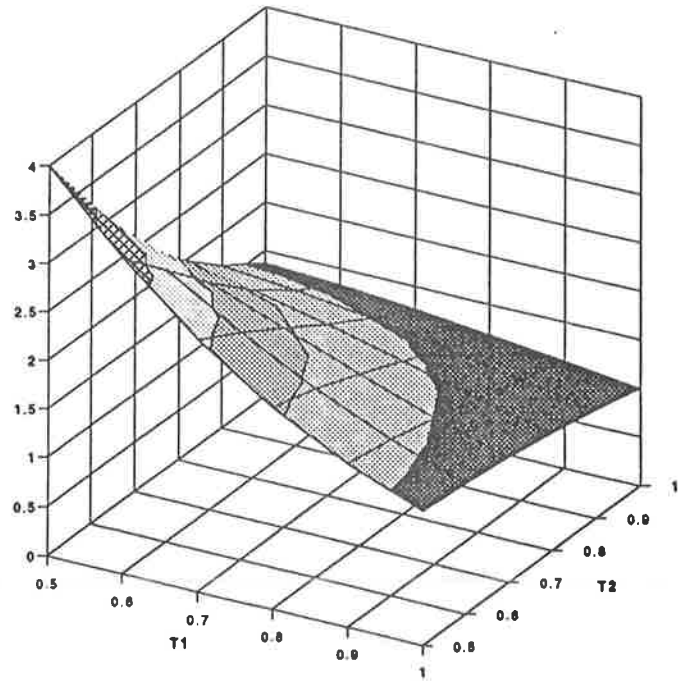


Fig. 4: Modification of Fig. 3 to emphasize time dependence



QUALITATIVE FAULT DETECTION BASED ON LOGICAL PROGRAMMING APPLIED TO A VARIABLE AIR VOLUME AIR HANDLING UNIT

L. Fornera*, A.S. Glass‡, P. Gruber‡ and J. Tödtli‡

* via Case Conti 3, CH-6616 Losone, Switzerland.
‡ Landis & Gyr, CH-6301 Zug, Switzerland

Abstract. A qualitative approach for detecting faults or sub-optimal operation in a class of heating, ventilation & air-conditioning systems is presented. In particular, qualitative models are used to describe the steady-state operation of a controlled central air-handling plant. Included among the variants of the underlying fault-detection approach considered is one using a model-based predictor in a logical programming environment. The predictions of the qualitative steady-state models are compared with numerical simulations of the dynamical system behaviour.

Key Words. heating ventilation and air-conditioning (HVAC), central air-handling unit, fault detection and diagnosis, qualitative modelling, logical programming.

1. INTRODUCTION

We consider the feasibility of a qualitative approach for diagnosing a class of faults in a variable air-volume air-handling system. This paper reports the results of some preliminary investigations making use of logical programming methods (Fornera *et al.*, 1993).

The system considered is a simplified version of the I.E.A. Annex 25 [1] reference air-handling system described by Kelly (1993). We deal with some illustrative faults of the central air-handling plant which result in deterioration of operation as distinct from actual failure — indeed some of the faults chosen reflect situations known to occur frequently in practice.

Our investigations comprise simulation of the air-handling system, with and without faults, an analysis of the faults considered in terms of qualitative models, and testing the models' predictions within a PROLOG logical programming environment.

In what follows, we begin in §2 by describing the system modelled, as well as the simulation methods employed. In §3 the relevant features of the steady-state behaviour of the CAHP system is discussed. Section 4 deals with the detectors themselves, including some relevant examples. Finally, we present our conclusions and outlook for future work in Sectionb 5. The results of the quantitative simulations are included in the appendices.

2. DESCRIPTION OF THE AIR-HANDLING SYSTEM

2.1 The reference system

Our investigations have concentrated on the central air-handling plant ("**CAHP**") of a simplified system based on the I.E.A. Annex 25 reference system described by Kelly (1993). The reference system – with a slightly different notation from that used by Kelly – is depicted in Figure 1.

The central unit comprises a bypass mixer and a heating coil followed by a cooling coil in a single air channel. The bypass dampers are controlled to provide a mixture of outside air and recirculated air; the amount of outside air may vary between 20% of the maximum airflow through the CAHP and 100%.

1 I.E.A. is an abbreviation for the International Energy Agency. Annex 25 is part of the I.E.A. programme, "Energy Conservation in Buildings and Community Systems"; and is specifically concerned with "Real-time simulation of HVAC systems for building optimisation, fault detection and diagnosis".

The air for each of the three zones is processed through a variable air volume ("**VAV**") box containing a damper and a re-heating coil. The airflow to each zone as regulated by the damper may vary between 40% and 100% of the specified maximum. In point of fact, the maximum flow from the CAHP is not sufficient to provide maximum flows through all three zones simultaneously, so limiting effects could occur.

Figure 1 ***The I.E.A. Annex 25 standard air-handling system***

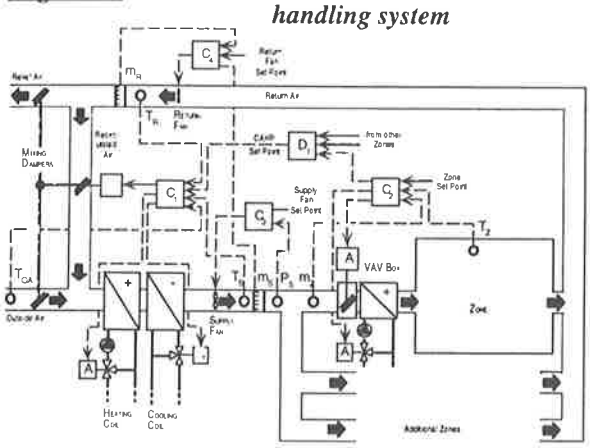


Figure 2 ***The Simplified System***

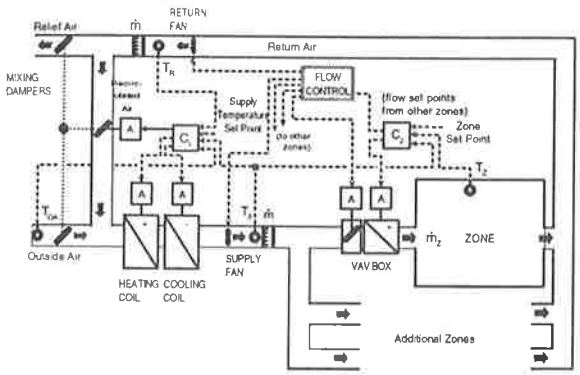


Table 1 **Notation used**

Quantity	*Symbol [Units]*	*Corresp. Symbol in Kelly (1993)*
Outdoor air temperature	T_{OA} [°C]	T1
Supply air temperature	T_S [°C]	T5
Return air temperature	T_R [°C]	T2
Zone temperature	T_Z [°C]	T6
Supply airflow rate	$\dot{m}_S$ [kg/s]	F1 (vol. flow rate)
Return airflow rate	$\dot{m}_R$ [kg/s]	F2 (vol. flow rate)
Airflow rate to zone	$\dot{m}_Z$ [kg/s]	VP (vol. flow rate)
Pressure in supply duct	P_S [kg/s]	SP

Five controllers share the task of regulating the fans, dampers, heating coils and cooling coils so as to attain the required temperatures and airflows in each of the three zones. Referring to Figure 1, Controller C_1, which mainly regulates the CAHP, attempts to maintain the supply air temperature T_S at its set point by operating the preheating coil, dampers and cooling coil in sequence. Controller C_2, which regulates a single zone, attempts to maintain the zone temperature T_Z at its set point by operating in sequence the damper and re-heating coil in the VAV box. Controllers C_3 and C_4 regulate the airflow through the CAHP: C_3 attempts to maintain the pressure P_S in the main supply air duct at its set point, and C_4 controls the return fan airflow rate so as to ensure that the difference $\dot{m}_R - \dot{m}_S$ between the return and supply airflow rates is a fixed, positive amount.

Controller C_2 actually consists of two cascaded controllers: the main zone controller attempts to maintain the zone temperature T_Z at its set point by acting in sequence on the reheater coil and a secondary controller, to which it supplies the zone air-flow set point. The secondary controller attempts to maintain the zone air flow at that set-point value.

The discriminator D_1 receives information from the various zone controllers C_2 and determines the highest supply air temperature set-point in a fixed range (13.9°C–18.3°C) compatible with ensuring that the demands of all the zones can be met.

2.2 The simplified system

The system that we have modelled thus far differs from the reference system in a number of points. First and foremost, for reasons of tractability, neither the component models nor the control strategies have yet been implemented in full detail. Besides that, there are some minor differences in the details of the control strategies used.

The modified system is illustrated in Figure 2. It differs from the reference system with regard to the following points:

1. All heating coils and cooling cools omit the actual physical heat exchange between the air and the hot water or coolant. The effect on the air of a prescribed heat transfer rate is modelled.
2. No reference to pressure is made in this model. Simulation models for pressure-dependent airflows have been developed, but have not yet been integrated in the present system.
3. The difference between supply and return airflows is assumed to be 0 ($\dot{m} = \dot{m}_R = \dot{m}_S$). We neglect air losses through open windows, leakage, etc. The total airflow through the CAHP is assumed to be the sum of the flows through the various zones.

$$\dot{m} = \dot{m}_{Z_1} + \dot{m}_{Z_2} + \dot{m}_{Z_3} \qquad (1)$$

4. Controllers C_3 and C_4 have been replaced by an idealized "flow control" which ensures that the flows $\dot{m}_{Z_1}$, $\dot{m}_{Z_2}$ and $\dot{m}_{Z_3}$ meet the requirements set by the zone controller, subject to the limitation that the resulting total airflow $\dot{m}$ through the CAHP does not exceed a specified maximum. Whenever the total demand exceeds this limit, the maximum available flow is shared among the three zones in proportion to the requirements set by their respective controllers.

5. Discriminator D_1 has been omitted. At the moment, the correct supply temperature set point is assumed to be fixed. It is planned to add the discriminator to future versions of the simulated system.

6. Controller C_2, the zone controller, does not include the zone airflow among its inputs. It is assumed that the zone airflow control operates *ideally* so as to maintain the zone airflow at its prescribed value $\dot{m}_{Z_1}$ as described in Point 4 above.. In its current implementation, however, it includes an economy mode feature not prescribed in the reference system, and therefore takes the supply air temperature into account.

7. Controller C_1, the CAHP controller, functions similarly to its reference system counterpart except for the fact that the supply air temperature set point is presumed constant (as mentioned in 5, above) and except for the differences in control strategy mentioned below.

8. When Controller C_1 operates the dampers in the bypass mixer, the minimal proportion of outside air is 20% of the *current total flow*, whereas the reference system sets a minimum of 20% of the maximum flow. The reference system specification will be implemented in future versions.

9. In our implementation of the economy control, reversing the direction of operating the bypass dampers occurs when the outside air temperature T_{OA} matches the *return air* temperature T_R rather than the fixed value of 22.2°C as proposed in the reference system specifications.

2.3 Simulation

For our preliminary investigations, the layout shown in Figure 2 was simulated on an Apple® Macintosh® IIfx computer using the SIMULINK™ applications software package[2]. The simulation program was used to investigate the quantitative dynamic behaviour of the system under the various fault conditions discussed in the following sections.

In addition, a MATLAB™ program was developed to investigate the steady-state behaviour of the system. Such investigations are extremely useful in assessing whether the specified design conditions can be met. In our case, pending a working simulation model of the discriminator D_1, such analyses were helpful in deciding what supply temperature set-points would ensure correct operation of the system in given situations.

3. THE STEADY-STATE BEHAVIOUR OF THE SYSTEM

We consider the operating regimes of the CAHP controller C_1 under steady-state conditions. In order to optimize energy consumption, the controller attempts to mix the outside air and the return air so as to ensure that the temperature T_M of the mixed air is as close as possible to the supply air set-point temperature. If the fraction of outside air in the mix is given by χ, the temperature of the mixed air is given by

$$T_M = \chi T_{OA} + (1-\chi)T_R, \qquad (2)$$

where χ is constrained between a specified minimum χ_{min} and maximum χ_{max}.

$$0 < \chi_{min} \le \chi \le \chi_{max} \le 1 \qquad (3)$$

In our case, $\chi_{min.} = 0.2$ and $\chi_{max} = 1.0$.

Since it is the temperature *difference* between T_M and T_S that is crucial, the various situations that may occur can be visualized in terms of a graphical representation with axes $T_{OA} - T_S$ and $T_R - T_S$ as depicted in Figure 3.

Combinations of T_{OA} and T_R yielding a given mixed air temperature correspond to straight lines in this representation. Combinations in which $T_M = T_S$ are straight lines through the *origin*. However, in view of the constraints imposed on χ, only those lines with slopes between

$$\frac{\chi_{max}}{1-\chi_{max}} \quad \text{and} \quad \frac{\chi_{min}}{1-\chi_{min}} \qquad (4)$$

correspond to situations in which the supply temperature can be achieved by operating just the dampers. Otherwise, depending on whether or not

[2] Program for the simulation of dynamic systems; one of the optional facilities offered with the MATLAB™ software package from The MathWorks, Inc.

$T_{OA} \leq T_R$ and $T_S < T_R$, the optimal damper setting will correspond to χ_{min} or χ_{max}.

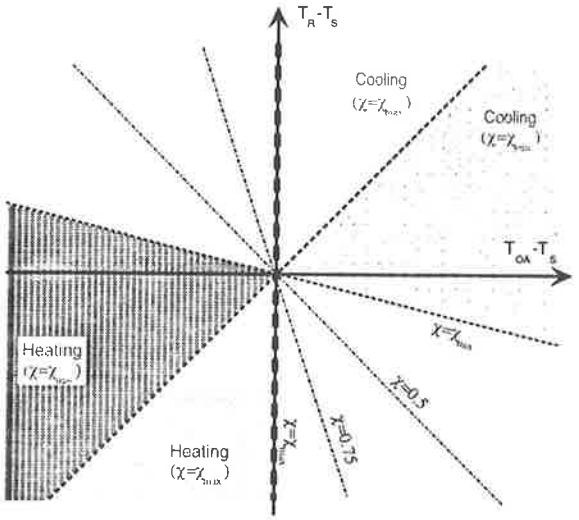


Figure 3 Graphical representation of CAHP controller operating regimes in terms of steady-state temperature conditions.

The various regimes are depicted in the diagram. The line $T_{OA} = T_R$ determines when χ should be switched from χ_{min} to χ_{max} or vice versa. The white regions are those in which it is only necessary to operate the dampers to achieve the desired effect. The shaded regions in the upper right sectors require cooling, with $\chi = \chi_{max}$, if $T_{OA} \leq T_R$, or $\chi = \chi_{min}$ otherwise. Similarly, the shaded regions in the lower left sectors require heating, with $\chi = \chi_{min}$, if $T_{OA} \leq T_R$, or $\chi = \chi_{max}$ otherwise. We summarize the six operating regimes:

1. $T_{OA} \leq T_R$ and T_{OA} comparatively low: dampers set for minimal outside air and controller operates heating coil.
2. $T_{OA} \leq T_R$ and $T_M = T_S$ can be achieved within the operating range of the bypass dampers: heating and cooling switched off and controller operates dampers.
3. $T_{OA} \leq T_R$ and T_{OA} comparatively high: dampers set for maximal outside air and controller operates cooling coil.
4. $T_{OA} > T_R$ and T_{OA} comparatively high: dampers set for minimal outside air and controller operates cooling coil.
5. $T_{OA} > T_R$ and $T_M = T_S$ can be achieved within the operating range of the bypass dampers: heating and cooling switched off and controller operates dampers.
6. $T_{OA} > T_R$ and T_{OA} comparatively low: dampers set for maximal outside air and controller operates heating coil.

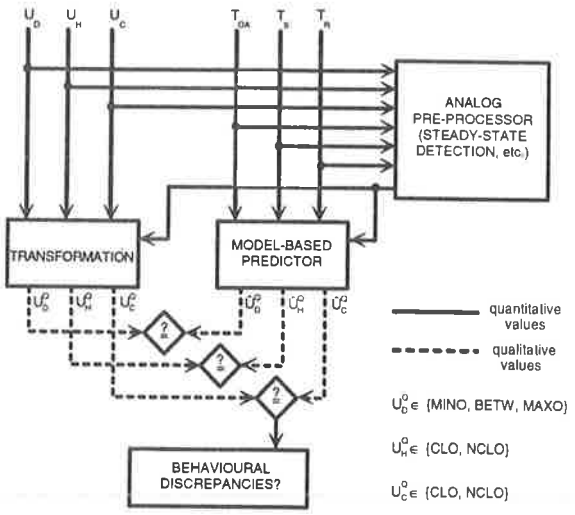
However, Cases 5 & 6 above can normally be ruled out. The VAV boxes are not equipped to cool the only cooling effects in the zones are thermal losses when either the outside air is cold or the walls are cold. Thus, in steady-state situations, we would only expect $T_R < T_Z$ to occur if the outside temperature is sufficiently *cold* that $T_{OA} < T_R$. In fact, barring transient effects (taking the complete system into account), no temperature states may be expected to be found anywhere in the lower right quadrant of Figure 3, so that the main cases of interest are Cases 1 to 4, which correspond to the four operating regimes prescribed for Controller C_1 in the reference system (Kelly, 1993).

4. DESIGN OF QUALITATIVE FAULT DETECTORS OF THE CENTRAL AIR HANDLING PLANT

4.1 General structure of the fault detector

The task of a fault detector is to recognize those changes of transient or steady-state behaviour of the air handling system which arise from faults. The types we describe operate on the central air-handling plant and are fault *detectors* in the strict sense: they only detect the *presence* of faults, without attempting to diagnose possible causes. They are particularly simple illustrative examples and are not capable of *localizing* or *diagnosing* faults. We consider three qualitative model-based fault detectors.

Figure 4 ***A qualitative model-based fault detector***



The overall structure of such a qualitative model-based fault detector is shown in Figure 4. This conforms to the structure of the so-called general diagnostic engine ("GDE") (de Kleer and Williams,

1987, Dexter and Glass, 1993). The fault detector models we consider are also related to the generic fault detection and diagnosis ("FDD") scheme proposed by Rossi and Braun (1993).

From the central air handling plant the observed values of the temperatures T_{OA}, T_R, T_S and the control variables U_H, U_D, U_C are obtained and fed into the detector. There they are used in the steady state detector which initializes both the transformation of the quantitative control values U_H and U_D, U_C to qualitative values U_H^Q, U_D^Q and U_C^Q and, at the same time, the generation of the corresponding predicted values $\hat{U}_H^Q$, $\hat{U}_D^Q$ and $\hat{U}_C^Q$.

The qualitative values of the outputs of the transformation and predictor blocks are chosen from the following sets:

$$\begin{cases} U_H^Q \in \{\text{CLO}, \text{NCLO}\}, \\ U_D^Q \in \{\text{MINO}, \text{BETW}, \text{MAXO}\}, \\ U_C^Q \in \{\text{CLO}, \text{NCLO}\}. \end{cases} \tag{5}$$

$$\begin{cases} \hat{U}_H^Q \in \{\text{CLO}, \text{NCLO}\}, \\ \hat{U}_D^Q \in \{\text{MINO}, \text{BETW}, \text{MAXO}\}, \\ \hat{U}_C^Q \in \{\text{CLO}, \text{NCLO}\}. \end{cases} \tag{6}$$

This corresponds to the idea of defining qualitative values in terms of so-called landmarks (Kuipers, 1984, 1985, 1986; Dexter and Glass, 1993), the landmarks in this case being provided by the extreme settings of the various actuators in the CAHP.

The predictor block is based on a qualitative prediction model of the plant to be supervised. The steady state behaviour of the central air handling plant is used as prediction model with the temperatures T_{OA}, T_R and T_S as inputs and the predicted control variables $\hat{U}_H^Q$, $\hat{U}_D^Q$ and $\hat{U}_C^Q$ as the outputs. The outputs of the transformation block and the predictor are then compared and checked for behavioural discrepancies as can be seen in Figure 4. The differences among the three detectors presented here concern the *predictor*.

4.2 Fault Detector 1

The first type of fault detector uses the graphical representation of Figure 3 as the prediction model. For each temperature triple T_{OA}, T_R, T_S a point in the plane of Figure 3 can be determined whose location fully determines the qualitative outputs $\hat{U}_H^Q$, $\hat{U}_D^Q$ and $\hat{U}_C^Q$ of the predictor. Figure 5 shows how the plane of the graphical representation of Figure 3 can be partitioned into regions according to the predicted actuator settings. Each region is labelled with the qualitative predicted actuator settings that correspond to the correct behaviour of the plant. The predicted actuator settings are also shown on some of the the lines separating the regions. On the line $T_R = T_{OA}$, however, the predicted values of $\hat{U}_D^Q$ are ambigouous.

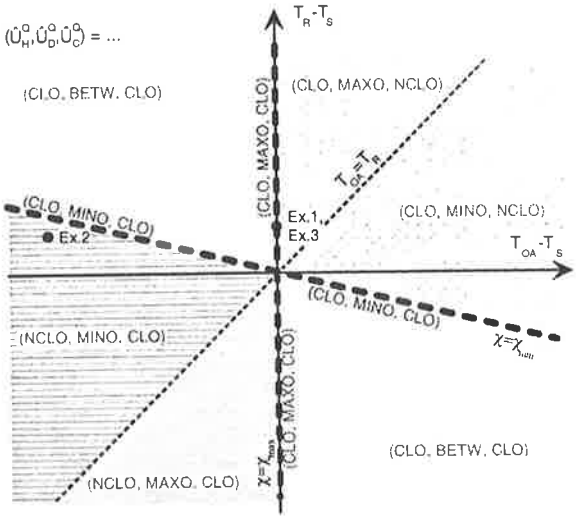


Figure 5 Graphical representation of predicted qualitative actuator settings in terms of steady-state temperature conditions.

Three examples illustrate how Fault Detector 1 works. The corresponding steady-state temperatures are shown in Figure 5.

Example 1

A steady state with the following values is detected:

$$T_{OA} = T_S, \quad T_{OA} < T_R, \quad \textit{fault: none} \tag{7}$$

It follows that (as indeed the simulations confirm)

$$U_H = 0, \quad U_D = 1, \quad U_C = 0 \tag{8}$$

The transformation block generates the following qualitative control values:

$$U_H^Q = \text{CLO}, \quad U_D^Q = \text{MAXO}, \quad U_C^Q = \text{CLO} \tag{9}$$

The steady state temperature, which are fed into the predictor lead to the points labelled "Ex.1" in Figure 5. This produces the qualitative prediction values:

$$\hat{U}_H^Q = \text{CLO}, \quad \hat{U}_D^Q = \text{MAXO}, \quad \hat{U}_C^Q = \text{CLO} \tag{10}$$

No discrepancy is found, so no fault is detected, which is indeed correct.

Example 2

A steady state with the following values is detected:

$$T_{OA} < T_S, \quad T_{OA} < T_R, \quad \textit{fault: cooling valve cannot close if } U_C = 0 \tag{11}$$

e.g.:

$$T_{OA} = -15°C;\ T_S = 18°C; \qquad T_R = 20°C \qquad (12)$$

Simulation yields the result that

$$U_H = 0.593, \quad U_D = 0.2 \text{ (min.)}, \quad U_C = 0 \qquad (13)$$

The transformation block now generates

$$U_H^Q = \text{NCLO}, \quad U_D^Q = \text{MINO}, \quad U_C^Q = \text{CLO} \qquad (14)$$

The steady state temperatures, which are fed into the predictor lead to the points labelled "Ex.2" in Figure 5. The prediction is

$$\hat{U}_H^Q = \text{NCLO}, \quad \hat{U}_D^Q = \text{MINO}, \quad \hat{U}_C^Q = \text{CLO} \qquad (15)$$

No discrepancy is found, so no fault is detected, although one is in fact present. Thus, in this particular operating steady state, the fault described cannot be detected. The air is unnecessarily heated up and cooled down, which results in wasted energy. In the simulation of the above example *without* the fault, the steady-state heating valve setting turned out to be $U_H = 0.514$.

Example 3

The same steady state is detected as in Example 1, but this time the fault of Example 2 occurs, namely the cooling valve cannot close completely if $U_C = 0$.

$$T_{OA} = T_S, \qquad T_{OA} < T_R \qquad \textit{fault: cooling valve cannot close if } U_C = 0 \qquad (16)$$

Simulation yields the values

$$U_H = 0, \qquad U_D = 0.916, \quad U_C = 0 \qquad (17)$$

The transformation block generates the following qualitative control values:

$$U_H^Q = \text{CLO}, \quad U_D^Q = \text{BETW}, \quad U_C^Q = \text{CLO} \qquad (18)$$

The location of this steady state in the diagrams of Figure 5 is the same as in Example 1. Consequently, the predictor produces the following qualitative predicted control values:

$$\hat{U}_H^Q = \text{CLO}, \quad \hat{U}_D^Q = \text{MAXO}, \quad \hat{U}_C^Q = \text{CLO} \qquad (19)$$

In this case the first predicted value differs from U_D^Q. That means a fault has now been detected.

If in Figure 3 the separation line representing the minimal damper position is missing or not known exactly, a **reduced** version of Fault Detector 1 can be derived, which is simpler. The reduced version is also able to detect the fault of Examples 2 and 3. This remark applies especially to the situation occurring in the reference system, in which the minimal outside air flow is absolute (20% of the maximum flow) rather than proportional. In this case, the slope of the line corresponding to χ_{min} will depend on the instantaneous total airflow through the CAHP.

Figure 6 **PROLOG program[3] for Fault Detector 2**

```
airhandler(AirhandleState,Ud,Uh,Uc,D_as,D_rs,D_ra) :-
    sum(D_rm,D_ms,D_rs),
    sum(D_am,D_ms,D_as),
    sum(D_mh,D_hs,D_ms),
    damper(DampState,Ud,D_am,D_rm),
    heater(HeatState,Uh,D_mh),
    cooler(CoolState,Uc,D_hs),
    controller(ContrState,Ud,Uh,Uc,D_ra),
    sum(D_ra,D_as,D_rs),
    airhandlcond(AirhandleState,DampState,HeatState,
        CoolState,ContrState).

%airhandlcond(AirhandleState,DampState,HeatState,
    CoolState,ContrState). airhandlcond(ok,ok,ok,ok,ok).

%damper (DampState,Ud,D_am,D_rm)
damper(ok,maxo,zero,Temp) :- anyTemp(Temp).
damper(ok,betw,neg,pos).
damper(ok,mino,neg,pos).
damper(ok,betw,zero,zero).
damper(ok,mino,zero,zero).
damper(ok,betw,pos,neg).
damper(ok,mino,pos,neg).

%heater(HeatState,Uh,D_mh).
heater(ok,clo,zero).
heater(ok,nclo,neg).

%cooler(CoolState,Uc,D_hs).
cooler(ok,clo,zero).
cooler(ok,nclo,pos).

%controller(ContrState,Ud,Uh,Uc,D_ra)
controller(ok,mino,nclo,clo,pos).
controller(ok,DampPos,nclo,clo,zero) :- anyDampPos(DampPos).
controller(ok,maxo,nclo,clo,neg).
controller(ok,maxo,clo,nclo,pos).
controller(ok,DampPos,clo,nclo,zero) :- anyDampPos(DampPos).
controller(ok,mino, clo,nclo,neg).
controller(ok,DampPos,clo,clo,Temp) :- anyDampPos(DampPos),
    anyTemp(Temp).

%rest
sum(X,X,X) :- anyTemp(X).
sum(X,zero,X) :- anyTemp(X).
sum(zero,X,X) :- anyTemp(X).
sum(neg,pos,Temp) :- anyTemp(Temp).
sum(pos,neg,Temp) :- anyTemp(Temp).

anyTemp(neg).
anyTemp(zero).
anyTemp(pos).

anyDampPos(mino).
anyDampPos(betw).
anyDampPos(maxo).
```

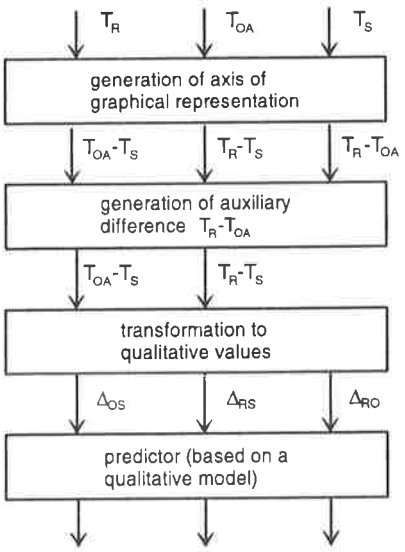
3 Note that the order of the controller outputs in this program is (U_D, U_H, U_C), whereas throughout the remainder of this paper we have shown the outputs in the order corresponding to the controller sequence, namely (U_H, U_D, U_C).

4.3 Fault detector 2

The design of Fault Detector 2 starts by describing the qualitative behaviour of the *components* of the central air handling plant: damper, cooler, heater and controller. Then the interaction of these components among themselves is described by defining the *structure* of the system incorporating them. The resulting qualitative prediction model is represented by logical clauses.

The form of the clauses is such that it can be executed as a PROLOG program, which is shown in Figure 6. It is structured as follows: in the first part of the code the structure of the air-handling unit is programmed – i.e. the way in which the components are linked. The second part deals with the components: damper, heater, cooler and controller. Finally, there are some auxiliary statements needed for the logical clauses.

Figure 7 **Structure of the predictor for Fault Detector 2**



The overall structure of the predictor is shown in Figure 7. It makes use of the auxiliary temperature differences

$$\Delta_{OS} = T_{OA} - T_S; \qquad \Delta_{RS} = T_R - T_S;$$

$$\Delta_{RO} = T_R - T_{OA}. \qquad (20)$$

The qualitative predictor consists of the qualitative prediction model and the PROLOG interpreter.

In the qualitative predictor block a prediction-type question such as

$$?- \text{airhandler}\left(\text{ok},\ \hat{U}_H^Q, \hat{U}_D^Q, \hat{U}_C^Q,\ 0, \text{pos}, \text{pos}\right) \qquad (21)$$

is posed to the PROLOG program. The predicted controller states compatible with fault-free operation are shown in Figure 8, based the graphical representation of Figure 3. It differs from Figure 5 in that the plane is subdivided into regions according to whether the *auxiliary* temperature differences Δ_{OS}, Δ_{RS} and Δ_{RO} are postive, negative or zero. As a result, three different qualitative controller states may be predicted for steady-state temperatures in the upper left or lower right quadrants.

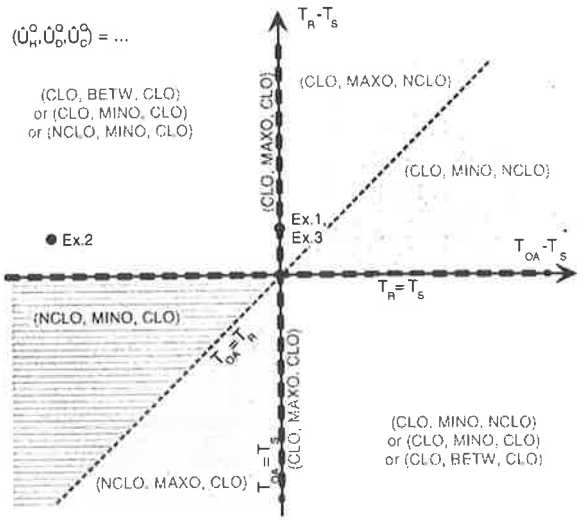


Figure 8 Graphical representation of predicted qualitative actuator settings in terms of *qualitative* steady-state temperature conditions (classified in terms of the signs of Δ_{OS}, Δ_{RS} and Δ_{RO}).

We illustrate the operation of Fault Detector 2 using the same three examples as for Fault Detector 1.

Example 1

The steady state is as in Example 1. The following qualitative controller values are observed:

$$U_H^Q = \text{CLO}, \quad U_D^Q = \text{MAXO}, \quad U_C^Q = \text{CLO} \qquad (22)$$

From the temperature measurements the qualitative values of the three temperature differences are calculated as:

$$\Delta_{OS} = 0, \quad \Delta_{RS} = \text{pos}, \quad \Delta_{RO} = \text{pos} \qquad (23)$$

With the assumption that the behaviour of the air handler is OK, the following predictor type question can be asked:

$$? = \text{airhandler}\left(\underbrace{\text{ok}}_{\text{assumption}},\ \underbrace{\hat{U}_H^Q, \hat{U}_D^Q, \hat{U}_C^Q}_{\text{predicted result}},\ \underbrace{0, \text{pos}, \text{pos}}_{\text{measurements}}\right) \qquad (24)$$

The result of the PROLOG program is (see appendix)

$$\hat{U}_H^Q = \text{CLO}, \quad \hat{U}_D^Q = \text{MAXO}, \quad \hat{U}_C^Q = \text{CLO} \qquad (25)$$

So again there are no discrepancies. Therefore no fault is detected, which is correct.

Example 2'

The steady state is as in Example 2. The observed qualitative controller values are the same:

$$U_H^Q = \text{NCLO}, \quad U_D^Q = \text{MINO}, \quad U_C^Q = \text{CLO} \qquad (26)$$

The qualitative temperature differences are:

$$\Delta_{OS} = \text{neg}, \quad \Delta_{RS} = \text{pos}, \quad \Delta_{RO} = \text{pos} \qquad (27)$$

The question posed now is

$$? = \text{airhandler}\left(\underbrace{\text{ok}}_{\text{assumption}}, \underbrace{\hat{U}_H^Q, \hat{U}_D^Q, \hat{U}_C^Q}_{\text{predicted result}}, \underbrace{\text{neg, pos, pos}}_{\text{measurements}}\right) \qquad (28)$$

The result of the PROLOG program is (see appendix)

$$\hat{U}_H^Q = \text{NCLO}, \quad \hat{U}_D^Q = \text{MINO}, \quad \hat{U}_C^Q = \text{CLO} \qquad (27)$$

Again the fault remains undetected, because no discrepancies are observed.

Example 3'

The steady state is as in Example 3. The same question is asked with the same input arguments generating the same predicted qualitative control values. The check for the behavioural discrepancy leads to the same result as in Example 2: the fault detector has detected that there is a fault.

Modelling the prediction model in this way has been motivated by work of Bratko *et al* (1989) and Burkhard (1992). But in contrast to the latter's work, the plant has been modelled here in a way that is independent of the particular sizing of the plant being monitored. This is important, since the adaptation of the fault detector (or FDD-system) to the plant should be as simple as possible and, furthermore, be based on easily available information.

4.4 Fault detector 3

Fault detector 3 is the same as Fault Detector 2 but the predictor part is in a compiled form, i.e. the predictor is in a form which, for a given input (Δ_{OS}, Δ_{RS}, Δ_{RO}), outputs the corresponding possible controller states ($\hat{U}_H^Q$, $\hat{U}_D^Q$, $\hat{U}_C^Q$). Such a form is, for example, the decision table shown in Table 2.

This table has been derived by posing for each possible input to the predictor a predictor-type question to the PROLOG program. For two input cases this leads to situations with three possible predictions (see Table).

Examples:

If we consider the same examples as for Fault Detector 2 we obtain the same results.

Table 2 **Predictor part of Fault Detector 3, realized as a decision table (positions marked in brackets are irrelevant)**

Δ_{OS}	Δ_{RS}	Δ_{RO}	Prediction No.	$\hat{U}_H^Q$	$\hat{U}_D^Q$	$\hat{U}_C^Q$
pos	pos	pos	1	CLO	MAXO	NCLO
pos	pos	0	1	CLO	MAXO or BETW or MINO	NCLO
pos	pos	neg	1	CLO	MINO	NCLO
pos	0	(neg)	1	CLO	MINO	NCLO
pos	neg	(neg)	1 2 3	CLO CLO CLO	MINO MINO BETW	NCLO CLO CLO
0	neg	(neg)	1	CLO	MAXO	CLO
neg	neg	neg	1	NCLO	MAXO	CLO
neg	neg	0	1	NCLO	MAXO or BETW or MINO	CLO
neg	neg	pos	1	NCLO	MINO	CLO
neg	0	(pos)	1	NCLO	MINO	CLO
neg	pos	(pos)	1 2 3	NCLO CLO CLO	MINO MINO BETW	CLO CLO CLO
0	pos	(pos)	1	CLO	MAXO	CLO
0	0	(0)	1	CLO	MAXO or BETW or MINO	CLO

4.5 Comparison of the three fault detectors

1) As mentioned before, all three fault detectors have the structure of a GDE and all distinguish the same qualitative values for the control variables U_H^Q, U_D^Q, U_C^Q. Their differences lies in the predictors.

2) Fault Detectors 2 and 3 are equivalent in their overall behaviour. This means that they give the same output in the same situation. Fault detector 1 however is not equivalent to Fault Detectors 2 and 3. It generates different answers in some situations. This can be seen by comparing the graphical representation of the predictor of Fault Detector 2 or 3 with the graphical representation of the predictor of Fault Detector 1. The comparison shows that the predictor of Fault

Detector 2 and 3 corresponds to the **reduced** version of Fault Detector 1. But although Fault Detector 1 is not equivalent with Fault Detector 2 and 3, there is no contradiction in their behaviour. This verifies our derivations of the fault detectors, at least to some extent.

3) The difference between Fault Detectors 2 and 3 lies in the predictor. The predictor of Fault Detector 2 has to run a PROLOG program in each detected steady state of the central air handling plant. In contrast to this, the predictor of Fault Detector 3 has only to look up some values in a decision table. The predictor of Fault Detector 2 is model based and the model is a deep model in the sense that it is composed of component models and a description on how the components are linked together. In contrary the predictor of Fault Detector 3 does not explicitly contain a deep model even if it is derived originally from a deep model of the plant.

5. CONCLUSIONS AND OUTLOOK

As is evident from the examples discussed, it is possible to define a fault detector for a central air-handling plant based solely on *qualitative* observable features. The results of the predictions using this qualitative model are in agreement with the quantitative simulations carried out.

As the examples show, qualitative tests require only a minimal knowledge of the system parameters. On the other hand, as expected (Dexter and Glass, 1993), the qualitative tests investigated here were not always able to discern faults that quantitative methods might have identified. Moreover, a potential limitation of the method tried is that the particular steady states of the system that lend themselves to such diagnosis may occur very infrequently in *normal operation*. One way of overcoming such a problem would be to devise active tests – say during the time when a building is unoccupied – to set up the conditions that allow for such fault detection.

This work is being continued by implementing a steady-state detector in the simulation program being developed, and linking the diagnostic tests with actual simulations (Glass *et al.*, 1994). It is also planned to implement the types of *active* diagnostic tests mentioned in the previous paragraph. As the research work progresses, the methods developed for the CAHP will be extended to the full reference air-handling system and to the actual localization and diagnosis of faults [e.g. using an assumption-based truth maintenance system (Gelle, 1993)].

The fault detectors considered here detect faults by analysing one steady state of the plant (or a subsystem of the plant). We shall consider qualitative fault detectors which also detect faults by comparing two (or more) different steady states together or by analysing transients. Knowing that a function is monotonic, for example, can be exploited in many cases [cf. work by Koch (1992)].

6. ACKNOWLEDGEMENT

The investigations described in this article have been partially funded by grants from the Swiss Federal Energy Office (Bundesamt für Energiewirtschaft) and the Swiss National Foundation (Schweizerische Nationalfonds).

7. REFERENCES

Bratko, I., Mozetic, I., and Lavrac, N. (1989). *KARDIO: A study in Deep and Qualitative Knowledge for Expert Systems*, M.I.T. Press, Cambridge, U.S.A..

Burkhard, E. (1992). "Wissensbasierte Modellierung einer Heizungs-Lüftungs-Klima-Anlage" Diploma thesis, Lehrstuhl für Logik und Informatik, ETH Zürich, September, 1992.

Dexter, A.L. and Glass A.S (1993). "The use of qualitative models in fault detection and diagnosis", §4.2 in *Building Optimisation and Fault Diagnosis System Concept*, J. Hyvärinen and R. Kohonen (eds.), Technical Research Centre of Finland (VTT), Laboratory of Heating and Ventilation, P.O. Box 206 (Lämpömiehenkuja 3), FIN-02151 Espoo, Finland (ISBN 952-9601-16-6).

de Kleer, J. and Williams, B.(1987). "Diagnosing Multiple Faults", *Artificial Intelligence*, **32**(1), 97-130.

Fornera, L., Glass, A.S., Gruber, P. and Tödtli, J. (1993). "Fault Detection Based on Logical Programming Applied to the VAV Reference System: First Trials", *I.E.A. Annex 25* Working Paper AN25/CH/050493/1.0, Tokyo meeting, 14-16.4.1993.

Gelle, E. (1993). "Diagnostic system for a heating-cooling unit", diploma thesis, Laboratoire d'Intelligence Artificielle, École Polytechnique Fédérale de Lausanne, March, 1993..

Glass, A.S., Gruber, P. , Roos,"M. and Tödtli, J. (1994). "Preliminary Evaluation of a Qualitative Model-Based Fault Detector for a Central Air-Handling Unit", *1994 IEEE Conference on Control Applications*, Glasgow, August 24-26, 1994.

Kelly, G.(1993) "Air conditioning unit", §5.3 in *Building Optimisation and Fault Diagnosis System Concept*, J. Hyvärinen and R. Kohonen (eds.), Technical Research Centre of Finland

(VTT), Laboratory of Heating and Ventilation, P.O. Box 206 (Lämpömiehenkuja 3), FIN-02151 Espoo, Finland (ISBN 952-9601-16-6).

Koch, G.(1992). "Knowledge-Based Coordination of Qualitative Online Diagnostic Tests", *IFAC Symposium on Intelligent Components and Instruments for Control Applications*, May 20-22, 1992, Malaga, Spain.

Kuipers, B (1984). "Commonsense reasoning about causality: Deriving behaviour from structure", *Artificial Intelligence*, **24**(1-3), 169-203.

Kuipers, B. (1985). "The Limits of Qualitative Simulation", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, William Kaufman, Los Altos, USA.

Kuipers, B (1986). "Qualitative Simulation", *Artificial Intelligence*, **29**(3), 289-388.

Rossi, T. and Braun, J., "Classification of Fault Detection and Diagnostic Methods" (1993). §2 in *Building Optimisation and Fault Diagnosis System Concept*, J. Hyvärinen and R. Kohonen (eds.), Technical Research Centre of Finland (VTT), Laboratory of Heating and Ventilation, P.O. Box 206 (Lämpömiehenkuja 3), FIN-02151 Espoo, Finland (ISBN 952-9601-16-6).

ANNEX I

PROLOG Program Pertaining to the Decision Table of Fault Detector 3 (cf. Table 2)

```
:- airhandler(ok, Ud, Uh, Uc, pos, pos, pos)
3 times        Ud=maxo,Uh=clo,Uc=nclo

:- airhandler(AirhandleState, Ud, Uh, Uc, pos, pos, zero)
3 times        AirhandleState=ok,Ud=maxo,Uh=clo,Uc=nclo
3 times        AirhandleState=ok,Ud=betw,Uh=clo,Uc=nclo
3 times        AirhandleState=ok,Ud=mino,Uh=clo,Uc=nclo

:- airhandler(ok, Ud, Uh, Uc, pos, pos, neg)
3 times        Ud=mino,Uh=clo,Uc=nclo

:- airhandler(ok, Ud, Uh, Uc, pos, zero, neg)
1 time         Ud=mino,Uh=clo,Uc=nclo

:- airhandler(ok, Ud, Uh, Uc, pos, neg, neg)
3 times        Ud=betw,Uh=clo,Uc=clo
3 times        Ud=mino,Uh=clo,Uc=clo
1 time         Ud=mino,Uh=clo,Uc=nclo

:- airhandler(ok, Ud, Uh, Uc, zero, neg, neg)
9 times        Ud=maxo,Uh=clo,Uc=clo

:- airhandler(ok, Ud, Uh, Uc, neg, neg, neg)
3 times        Ud=maxo,Uh=nclo,Uc=clo

:- airhandler(ok, Ud, Uh, Uc, neg, neg, zero)
3 times        Ud=maxo,Uh=nclo,Uc=clo
3 times        Ud=betw,Uh=nclo,Uc=clo
3 times        Ud=mino,Uh=nclo,Uc=clo

:- airhandler(ok, Ud, Uh, Uc, neg, neg, pos)
3 times        Ud=mino,Uh=nclo,Uc=clo

:- airhandler(ok, Ud, Uh, Uc, neg, zero, pos)
1 time         Ud=mino,Uh=nclo,Uc=clo

:- airhandler(ok, Ud, Uh, Uc, neg, pos, pos)
3 times        Ud=betw,Uh=clo,Uc=clo
3 times        Ud=mino,Uh=clo,Uc=clo
1 time         Ud=mino,Uh=nclo,Uc=clo

:- airhandler(ok, Ud, Uh, Uc, zero, pos, pos)
9 times        Ud=maxo,Uh=clo,Uc=clo

:- airhandler(ok, Ud, Uh, Uc, zero, zero, zero)
infinite times Ud=maxo,Uh=clo,Uc=clo
infinite times Ud=betw,Uh=clo,Uc=clo
infinite times Ud=mino,Uh=clo,Uc=clo
```

ANNEX II

Simulation of the simplified reference Air-Handling System

The dynamic behaviour of the simplified reference system was simulated numerically using the SIMULINK™ package using the physical parameters specified by Kelly (1993). In each simulation the outside air temperature T_{OA} was constant and the initial temperatures of the zones was 15°C. The system was allowed to run for a simulated time of 30 min with the zone controllers acting to try and achieve a set-point temperature of 20°C and the CAHP controller acting to try and achieve the supply-air temperatures noted below.

Simulations for 1/2 hour, requiring about 10-15 min. to reach steady state in both the central air-handling plant and the zones. In all cases, zone temperatures acceptably close to 20°C were achieved within approximately 10 min and the zone controller outputs attained an acceptable equilibrium within 15 to 20 min. The supply-air set-point temperatures were, as a rule, attained much faster.

Example 1

Relevant inputs:

Outside air temperature	T_{OA}	14.5°C
Supply air set point temperature	T_S^{SP}	14.5°C
Zone set point temperature	$T_{Z_k}^{SP}$	20°C

Faults: *none.*

Values of relevant quantities attaining steady state.

Return air temperature	T_R	20.001°C
Supply air temperature	T_S	14.500°C
Controller outputs	U_H	0.000
	U_D	1.000
	U_C	0.000

Figure 9 Simulation of the CAHP in Example 1, showing T_{OA}, T_S & T_R (upper plot) and U_H, U_D, U_C & $\dot{m}/\dot{m}_{max}$ (lower plot).

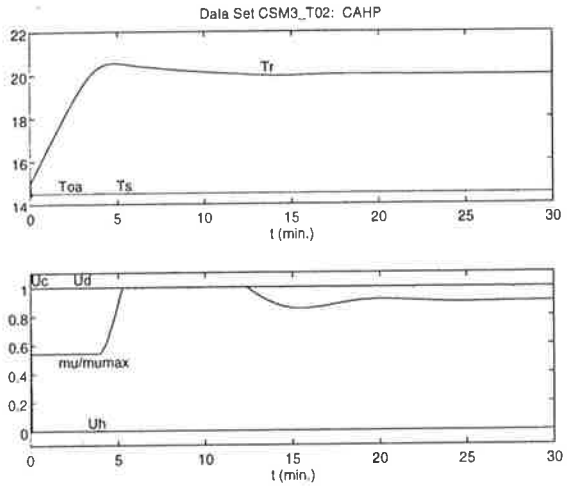
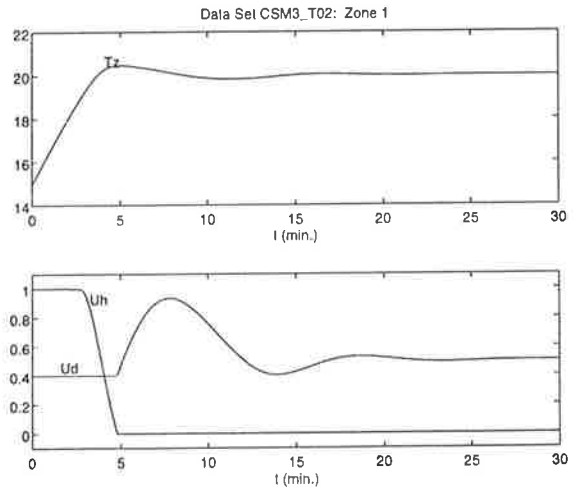


Figure 10 Simulation of one of the zones in Example 1, showing T_Z (upper plot) and the VAV controller outputs U_H & U_C (lower plot).



Example 2

Relevant inputs:

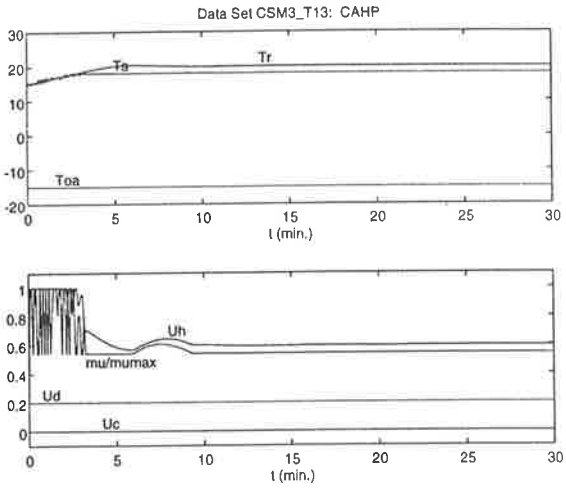
Outside air temperature	T_{OA}	-15°C
Supply air set point temperature	T_S^{SP}	18°C
Zone set point temperature	$T_{Z_k}^{SP}$	20°C

Fault: *cooling valve cannot close completely* - cooling coil operates at a minimum of 2.5% of its 40600 kW capacity.

Values of relevant quantities attaining steady state:

Return air temperature	T_R	c. 20°C
Supply air temperature	T_S	c. 18°C
Controller outputs	U_H	0.593
	U_D	0.200
	U_C	0.000

Figure 11 Simulation of the CAHP in Example 2, showing T_{OA}, T_S & T_R (upper plot) and U_H, U_D, U_C & $\dot{m}/\dot{m}_{max}$ (lower plot).



Example 3

Relevant inputs:

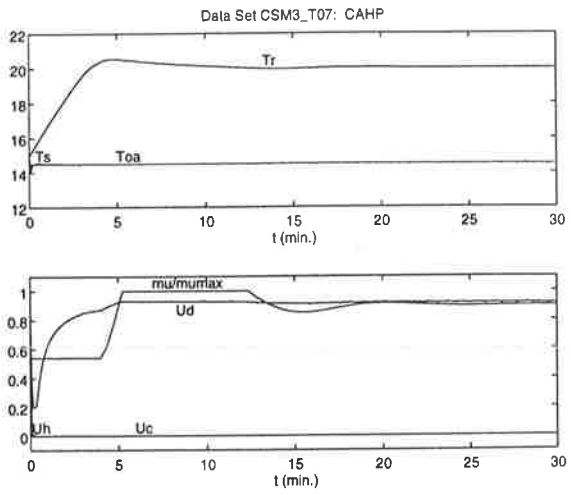
Outside air temperature	T_{OA}	14.5°C
Supply air set point temperature	T_S^{SP}	14.5°C
Zone set point temperature	$T_{Z_k}^{SP}$	20°C

Fault: *cooling valve cannot close completely* - cooling coil operates at a minimum of 2.5% of its 40600 kW capacity.

Values of relevant quantities attaining steady state.

Return air temperature	T_R	20.001°C
Supply air temperature	T_S	c. 14.5°C
Controller outputs	U_H	0.000
	U_D	0.916
	U_C	0.000

Figure 12 Simulation of the CAHP in Example 3, showing T_{OA}, T_S & T_R (upper plot) and U_H, U_D, U_C & $\dot{m}/\dot{m}_{max}$ (lower plot).



PROCESS DIAGNOSIS IMMUNE FROM SENSOR FAULT BY SELF-ORGANIZATION

Y. ISHIDA* and F. MIZESSYN**
*Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma, Nara, 630-01 Japan
**Lafarge Coppee Recherche, BP 15, St. Quentin Fallavier, 38291 La Verpilliere Cedex, France

Abstract. We discuss modification and extension of the immune net model to apply it to on-line real-time diagnosis of processing plants. The immune net models the recognition capability emergent from cooperative recognition of interconnected units. We also implemented a system on which we can build an immune net when the relations among process variables are given. Further, this approach is compared with that by the knowledge-based systems and that by the neural networks.

Key Words. Process diagnosis, immune net, self-diagnosis model

1. INTRODUCTION

Information processing of biological systems often gives insight for treating with complex systems. The immune network model, presented in this paper, focuses on the feature of mutual recognition [1], based on the analogy of the *idiotypic network* by Jerne (1973). That is, system level recognition is attained as an emergent phenomenon from unit level interactions.

For this immune network model, learning algorithm similar to those of neural network has been developed by Ishida (1990), Ishida and Mizessyn (1992).
In this paper, we will show that this algorithm with modifications can be applied to sensor diagnosis in process instrumentation systems. We also discuss the extension for the process fault detection.

Section 2 presents some results obtained for immune net, which are required in the diagnosis discussed in the following section. Section 3 modifies the immune net model for process diagnosis. Section 4 presents the application to sensor fault diagnosis of processes. The extension to process fault detection is discussed in section 5.

[1]Other than these two models, acquiring diversity by combinatorial generation may potentially give insight to building new information models.

2. MUTUAL RECOGNITION NETWORK MODEL

Consider the following problem. Many suspects said that some other suspects are (or aren't) criminal. The problem is to find the criminal based on the testimony. If a suspect is the criminal, he/she may lie. However, if a suspect is innocent he/she will tell the truth.

We can find such mutual testing models in fault diagnosis theory. Diagnostic algorithms for distributed diagnostic models have been studied, mainly from a viewpoint of attaining fault tolerance in distributed systems or distributed processing. The immune net model stated in this section is based on a distributed diagnostic model. There are many algorithms for this problem. We presented an algorithm by self-organization (Ishida 1990).

Preparata *et al* (1967) proved that if every suspect is tested by more than t other suspects, then the criminals can be identified with the presence of less than t criminals.

2.1. Parallel Processing in Mutual Recognition Network

We use the following notation. eij indicates the test on the unit uj done by the unit ui. Thus, the test outcome Tij of the test eij is defined:

- Tij=-1 when testing unit i is fault-free and tested unit j is faulty,
- Tij=1 when both units are fault-free,
- Tij=-1/1 otherwise.

In order to give a self-organizing capability for each unit, we suppose each unit has a simple information processing capability other than testing other units. That is, each unit decides its reliability by observing opinions of other units adjacent to the unit.

- $dri(t)/dt = \sum jTijRj + \sum jTjiRj - 1/2 \sum(Tij \neq 0)(Tij+1)$ for i=1,...,n (1)
- $Ri(t) = 1/(1+\exp(-ri(t)))$ for i=1,...,n

We have shown that this model will converge by the following energy function of the system in the same manner as that of Hopfield Net Algorithm.

2.2. The Function of a Unit

Focusing on the processing of one unit may help to understand the processing of immune network model. Since the immune network model is a homogeneous parallel processing model, all the processing done by each unit are the same. It should be noted that the reliability of one unit is evaluated not only from the opinions of other units testing the unit, but from the opinions of what the unit said to the other tested units. The former corresponds to the first term of the right hand side of the model (1) and the latter to those of the second and third terms. We call the latter *reflection effect*. The *reflection effect* is somewhat similar to the situation that if one say bad criticism on the highly evaluated person, it affects one's own credit not the person one criticized.

3. MODIFICATIONS ON THE MUTUAL RECOGNITION MODEL

So far, we have assumed that the test done by fault-free unit is always reliable. That is, if Ri = 1 then Tij = 1 implies that Rj = 1 and that Tij = -1 implies that Rj = -1. In existing systems, however, this is not always the case. As will be seen in the sensor diagnosis application, test may not be reliable even if it is done by fault-free unit due to the sensitivity of test. Formally said, even if Ri = 1 and Tij = 1, it may be the case that Rj = 0 because of the incompleteness of test. The test outcome, then, is defined as:

- Tij=-1 when sensor i is fault-free and sensor j is faulty,
- Tij=-1/1 otherwise.

It should be noted that the diagnosis with this incomplete test becomes more difficult than that with complete test, since the information obtained from test results is less (*i.e.*, even if we know the testing unit ui is fault-free and test Tij = 1, we cannot say that the tested unit uj is fault-free.). Further, with incomplete case $E \leq 0$ holds. This means, it is not possible to get any evidence from the test results Ti for believing some units fault-free, although the evidence of believing some units' faulty can be obtained. Thus, the initial values of the reliabilities are set to be 1, otherwise there is no chance for the unit to be evaluated as fault-free.

The immune net model under incomplete test becomes simpler form:

$dri(t)/dt = \sum(Tij \neq 0 \text{ or } Tji \neq 0)(Tij+Tji-1)Rj(t)$ for i=1,...,n (2)

We can know the solution of the above model converges on the consistent diagnosis, by calculating the time derivative of the Liapunov function.

Further modification is made on the above models for the sensor diagnosis application. Both (1) and (2) are designed to give clear diagnosis, faulty or fault-free, not in ambiguous state between them. That is, the reliability hardly stays around the intermediate values near 0.5. In some situations, however, the information of ambiguous state is also necessary rather than making them black or white.

The following model keeps the information of ambiguous state of sensor reliability.

$$dri(t)/dt = \sum (Tij \neq 0 \text{ or } Tji \neq 0)(Tij+Tji-1)Rj(t) - ri(t) \text{ for } i=1,...,n \quad (3)$$

We call the models complete, incomplete, modified incomplete respectively by the models (1), (2) and (3).

We have noted that the reasonable candidate for the initial value of R is (1, ... 1), since this is always near to the correct diagnosis assuming the number of faulty units is less than that of fault-free units. In reality, multiple fault does not occur at the same time. It often happens that after one unit falls into abnormal then another unit becomes faulty and so on. Thus, another practical way to determine the initial value R is to use the last value of R before new fault occurs. We used this strategy in the sensor network simulation in section 4.

4. APPLICATION TO SENSOR SELF-DIAGNOSIS

We applied these models to instrumentation systems of processing plant. In this section, self-detecting sensory network for processing plants has been proposed using the redundancy among process values. Temperature, pressure, and flow measured independently often have such interrelation, and hence interval correspondence between them, *e.g.*, between high temperature and high pressure.

If the process value calculated from other process value is not pinpoint, and only plausible interval is obtained, then we must say that the test here is incomplete. In that case, the model with incomplete test should be adopted. As we will discuss in the following, the network construction is done by the inequalities.

Example 1

Consider an example of a heat exchanger of the condenser type. Two flows *i.e.*, the flow of the shell side and flow of the tube side exchange the heat. Steam enters from the inlet of the shell side, then it is condensed, and finally being cooled by the flow of the tube side. The flowing object of the tube side then comes out being heated by the flow of the shell side.

Now, we consider the inequalities among temperatures of these two flows.

- Tho + 20 > Thi > Tho
- Tli + 10 > Tlo > Tli
- Tli + 50 > Thi > Tli
- Tlo + 40 > Tho > Tlo

where

- Thi : temperature at inlet of shell side,
- Tho : temperature at outlet of shell side,
- Tli : temperature at inlet of tube side,
- Tho: temperature at outlet of tube side.

Using these relations among sensors, sensor network will be constructed. For example, when the relation Tho + 20 > Thi > Tho does not hold, then either Thi or Tho can be faulty.
This means that the sensors measuring values Thi, Tho are testing each other through this relation. The tests done by these inequalities are incomplete, for even if the tested sensor is faulty it will not be recognized unless the tested sensor value fails to satisfy the inequality. Thus, the model under incomplete test should be used for this case.

Fig. 1 shows sensor network created by the Immune Net Generator[2]. Dark node indicates that the sensor is diagnosed as faulty and gray node as doubtful condition. Other than mutual links shown in Fig. 1, Immune Net Generator allows simple link (one way link), self link (test of one node comparing its value with some reference value) and multiple link (test of more than three nodes). Default tests for them are

[2]The Immune Net Generator is implemented on Macintosh in colaboration with Mr. K. Otuska at SMI.

$|x_i - x_j| < e$, $|x_i - M| < e$, and $|\sum x_i - M| < e$ respectively where x_i represents process values, M represents reference value and e some error constant.

Fig. 2 shows the time evolution of the sensor Thi and its reliability output from the Immune Net Generator, computed by the complete model. Fig. 3 also shows the time evolution and the reliability computed by the modified incomplete model. All the four sensors start in normal condition. After a while, the sensor Thi stuck to the HIGH position (140) for a while, then goes back normal condition (120) again. The other sensors remain constant:Tho (115), Tli (75) and Tlo (80).

It can be seen that the complete model always tries to make clear the black and white, while the modified incomplete one can keep the doubtful situation. The modified incomplete model can reflect the situation when the sensor becomes back to the normal condition.

It has been known that the immune network model works well for more complex sensor network in most of cases. We will show the example of a cement plant in section 6.

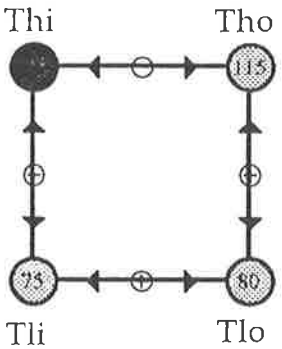


Fig. 1 Sensor Net of the Heat Exchanger Example

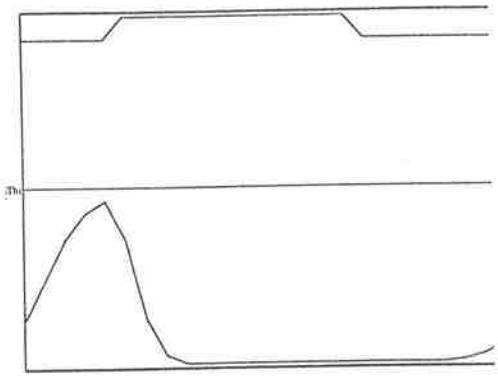


Fig. 2 Time Evolution of the sensor value Thi and that of its reliability computed by complete model. The sensor Thi stuck to the HIGH position (140) for a while, then goes back normal condition (120) again. The other sensors remain constant:Tho (115), Tli (75) and Tlo (80).

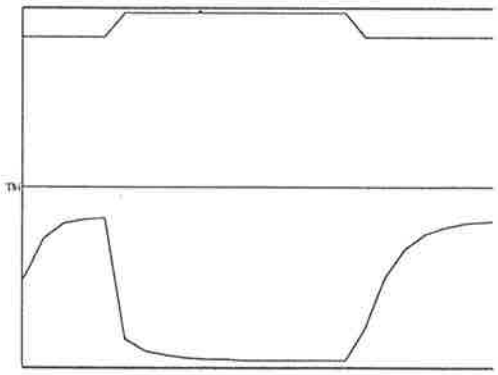


Fig. 3 Time Evolution of the sensor value Thi and that of its reliability computed by modified incomplete model. The sensor Thi stuck to the HIGH position (140) for a while, then goes back normal condition (120) again. The other sensors remain constant:Tho (115), Tli (75) and Tlo (80).

5. PROCESS FAULT DETECTION BY HIERARCHICAL IMMUNE NETWORK

So far, we have discussed only sensor fault identification. We will discuss how a process diagnosis including fault detection of the process can be carried out by more elaborate hierarchical sensor network. The hierarchical sensor network can be constructed by introducing the *virtual sensors.*

Example 2
As an illustrative example, consider the pipelines of confluence of two in-flows

(whose flow rate and temperature are Fi1, Fi2 and Ti1, Ti2 respectively) join into one out-flow (whose flow rate and temperature are Fo and Tio respectively). We have the following relation among sensor values by the mass and energy balance.

Balance Equations:
- Fi1 + Fi2 = Fo
- Fi1 Ti1 + Fi2 Ti2 = Fo To

Further, we suppose the following relations among the sensor values hold in a normal condition of the process by the feature of the process and the control structure.

Upstream Condition of the Process:
- Fi1 > Fi2
- Ti1 > Ti2

The first relation Fi1 > Fi2 means that if this relation is violated, there are three possibilities for that: (1) sensor Fi1 is faulty, (2) Fi2 is faulty or (3) the upstream process is abnormal.

Process fault disabling the relation Fi1 > Fi2, in our immune network framework, would disable the testing relation between Fi1 and Fi2. This type of process fault can be detected by considering the virtual sensor inserted between Fi1 and Fi2. The reliability of the virtual sensor is actually reliability of the relation Fi1 > Fi2 assured by the normal condition of the process, and it can be calculated by the measure of inconsistency between Fi1 and Fi2:

$$- Tij\ Ri(Rj\ -1/2(Tij + 1)) - TjiRj(Ri\ -1/2(Tji + 1))$$

for the complete model (1).

This measure of inconsistency is evaluated high when the values of Ri, Rj and Tij are inconsistent; *e.g.*, Ri = 1, Rj=1 and Tij =-1.

Example 3

By the balance equations and upstream conditions we can deduce the following relations:

- Ti1 Fi1 > Ti2 Fi2
- Ti1 Fi1 < To Fo
- Ti2 Fi2 < To Fo

Thus, when we use these relations as sensor network we must consider the *virtual sensors* Ti1 Fi1, Ti2 Fi2 and To Fo. Although these sensors do not physically exist, the information of multiplied value, not each sensor value, is used in the sensor network of these relations. Hence, we call this type of virtual sensor the *combined virtual sensors*. The faulty of the *combined virtual sensor* is interpreted as faulty of one of them.

In case that more than three sensors appear in one relation as in Fi1 + Fi2 = Fo. If we need construct a plain (as opposed to the hierarchical) sensor network for some reasons such as simplicity of model, then these relations as Fi1 + Fi2 = Fo can be deduced to several two sensor relations as Fi1 < Fo and Fi2 < Fo. But in the hierarchical view of sensor network, we can regard the relation Fi1 + Fi2 = Fo that the virtual sensor Fi1 + Fi2 watching the other sensor Fo and *vice versa*.

This type of the virtual sensor can be attained by the multiple link facilitated in Immune Net Generator.

The main characteristic of the immune network approach to process diagnosis is that this method admits only relative relation between process values without referring to the absolute value of the process values, hence does not suffer from the shifting of all the process values (which occurs depending on the load to the process or the change of environment such as seasonal change). However, it is possible for the immune net to involve the heuristics which refer to absolute values of the process values by the self link of the Immune Net Generator.

6. APPLICATION TO THE INDUSTRIAL PROCESS PLANT

Although we have applied the immune net model more complex model of a steel process, we show the application to a

cement process. In many cement plants, sensors are used in a severe environment with dust. Process diagnosis including sensor fault is needed. The immune net model allows the process diagnosis including sensor fault by taking consistency among many sensor values into account.

Example 4

Fig. 4 shows a sensor network generated by the Immune Net Generator for the preheating process of a cement plant. Twenty-two sensors (thermometers) are involved. In constructing sensor net, not only process knowledge but the experimental knowledge among sensor values is used to obtain enough relations for diagnosis.

The performance of the diagnosis depends on the quality of the relations involved; the diagnosability depends on the number of distinct tests, and the reliability of the diagnosis depends on the reliability of the involved tests.

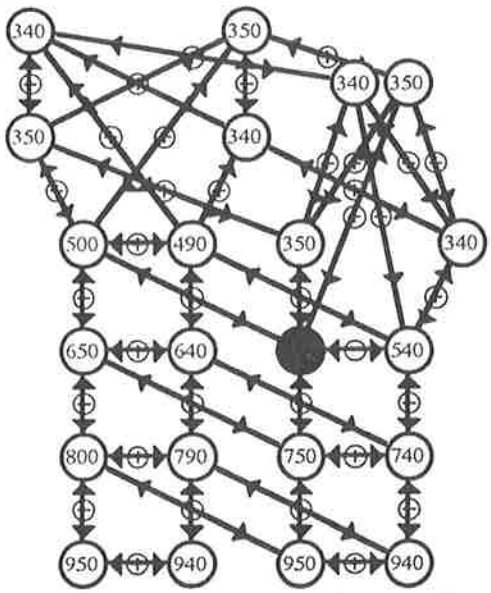


Fig. 4 Sensor Net of the Cement Plant Generated by the Immune Net Generator. The Value Inside the Node is the Sensor Value at Some Snapshot time. The Node with Dark Color Indicates the Sensor Diagnosed to be Faulty.

7. CONCLUSIONS

The immune net model focuses the idea of idiotypic network proposed in immunology. We presented several modifications on the immune net model to apply the self-organizing algorithm for sensor network in the instrumentation systems of processing plants.

Further, we developed a more elaborate hierarchical immune net model by introducing the virtual sensors. The hierarchical immune net model can detect not only sensor fault but process fault.

By the simulation for several existing processes such as cement process, the capability of self-diagnosis and self-elimination of the information of faulty sensor is demonstrated.

8. REFERENCES

Jerne, N.K. (1973). The Immune System. *Sci. Am.* Vol. 229, No.1, pp. 52-60.

Ishida, Y. (1990). Fully distributed diagnosis by PDP learning algorithm: Towards immune network PDP model. *Proc. of IJCNN90*, San Diego, USA, 777-782.

Ishida, Y. and Mizessyn, F. (1992). Learning Algorithms on Immune Network Model. *Proc of IJCNN92*, Beijing, China, I, 33-38.

Mizessyn, F. and Ishida, Y. (1993). Immune Network for Cement Plants. *Proc. of ISADS93*, Kawasaki, Japan, 282-288.

Preparata, F.P. , Metze, G. and Chien, R.T. (1967). On the connection assignment problem of diagnosable sysems," *IEEE Trans. Comp.*, EC-16, 848-854.

Rumelhart, D.E. and McClelland, J.L. (1986). *Parallel distributed processing.* MIT Press, USA., 1986.

Hopfield, J.J. (1982). Neural network and physical systems with emergent collective computational abilities. *Proc. Natl. Sci.* USA, Vol. 81, 3088-3092.

AN INTELLIGENT ALARM HANDLING TOOL

J. KRISTENSSON*, G. BENGTSSON*, J. BARKLUND**, P. MILDNER**

*First Control Systems AB, Klockartorpsg 14, S-723 44 Västerås, Sweden
**Uppsala University, Computing Science Department, Box 311, S-751 05 Uppsala, Sweden

Abstract. In production processes it is of vital importance to be able to predict consequences such as quality problems and shutdowns. First Navigator is a knowledge-based system tool consisting of a visual process database and an easily configurable "alarm-shell". The maintenance personnel will use decision trees for common problems, fault reports etc. to extract knowledge into rules. A specially designed expert system will listen to and filter time-tagged events. When the operating personnel asks for advice, the expert system will predict consequences with time estimates in the process.

Key Words. Alarm systems; artificial intelligence; knowledge engineering; prediction; system failure and recovery; maintenance engineering; process control

1. INTRODUCTION

A shutdown in a process, e.g., a continuous casting machine for billets, a paper machine or a steel mill, is usually very costly. Today, when a fault condition is encountered, the operating personnel use recorded data and measurements in combination with searching in control equipment and process documentation to arrive at some action to be taken. This paper will describe techniques to acquire and maintain knowledge in various processes in order to foresee problems, make rapid fault-diagnoses and take correct and accurate actions. The implementation of these functions is done using a new knowledge-based system (KBS) environment, First Navigator (FN), explained in more detail in next section. Briefly, FN is a graphical interface to a process database with an intelligent alarm handler. The database consists of data about the process and the control instrumentation together with documentation, used by the operator to diagnose faults and take correct actions. The objects and their attributes in this database are needed when writing the rules for the alarm handler.

The intelligent alarm handling tool is a specially designed expert system listening to and filtering time-tagged events from the control system. The operating personnel can consult the expert system to predict consequences in the process, suggest actions to be taken and explain causes of fault conditions. The expert system used in the KBS is a dedicated tool for reasoning about time tagged events, constructed by J. Barklund and P. Mildner at Uppsala University based on experience they have gained in a project with a real time expert system described by Barklund and Mildner (1994). The expert system was developed as part of the project "Intelligent Real Time Systems", supported by the Swedish National Board for Industrial andTechnical Development (NUTEK).

At the time this paper was written, the user experience was limited to a field test including the graphical database part of FN. Process information from a small chemical process was inserted into the database and a graphical model was automatically created. Another project to test the intelligent alarm handling at a paper mill is planned.

2. FIRST NAVIGATOR

FN is a software prototype developed during the project "Knowledge-based operator tools", as a part of a development program for the process industry supported by Swedish National Board For Industrial and Technical Development (NUTEK). Based on the rapid prototyping software tool, HyperCard, FN is running on Apples Macintosh computers. It can be seen as a graphical interface between the

operator and the process database, also connected to the control system as can be seen in Fig. 1.

FN consists of two parts:

- integrated factory documentation
- intelligent alarm handling

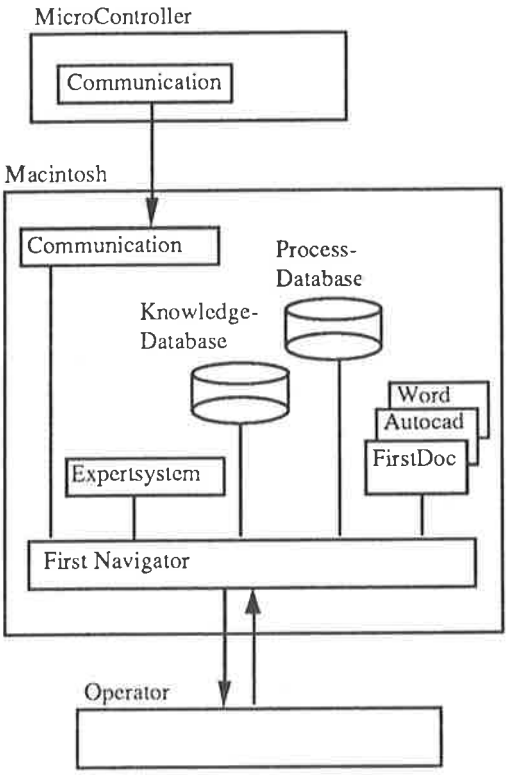


Fig. 1. Basic structure

2.1. Visual Database

In every process instrumentation project, a vast number of documents have to be constructed. What kind of documents are needed is more or less standardised. The instrumentation can be seen as a model consisting of three different representations:

- process equipment,
- control equipment
- application programs.

FN has a skeleton ready to accommodate data for any kind of process. Just enough data has to be inserted by the user for the tool to be able to automatically generate a visual model of the three representations. Process equipment, e.g., pumps, actuators, sensors and motors, vital for running the process, are all represented in this database. Each process object consists of several attributes or signals. Detailed information, such as circuit-drawings, functional descriptions, etc., can be connected to appropriate places in this model.

The user can easily navigate in the visual database. From schematic process-pictures, various kinds of documents (drawings, connection tables etc.) can be shown in windows.

2.2. Configurable "Alarm-shell"

The integrated factory documentation is a prerequisite for the alarm handling tool. At the same time the process is documented, a framework of process-objects is created to which knowledge can be connected. The connection to the control equipment is already done in defining the process database. A cross-reference table defines connections between the process-signals (knowledge-base) and their place in a specific signal package (control equipment).

Rules are written in a rule-editor using the objects and their attributes present in the database. A typical rule will have a main condition with an event leading to a state or to an other event, with a time estimate (see Fig. 2). The rule can also have one or more secondary conditions.

Rule 2 **Motor A Down**
If *Motor A Temp High is true*
Then *Motor A Down is true*
After *20 min*
Provided *Process-part B Running is true*
And *Fan C On is false*

Fig. 2. A typical rule

In the rule above there are three process objects involved; Motor A, Process part B and Fan C. These objects, together with their signals, have been inserted into the process database at an earlier stage. In writing the rule, the user has access to all this data through pop-up menus. When the user is satisfied, the rule is examined and the correct syntax for alerts, states and rules are generated automatically. All rules are saved in a knowledge-base which will be loaded when the intelligent alarm handling is started.

3. KNOWLEDGE ACQUISITION

Knowledge about the process is gathered in many different ways. From the beginning, only the obvious process connections will constitute the knowledge-base. The operating and maintenance personnel will have the key responsibility in developing the knowledge-base. Fault trees and fault-reports are two important sources. In a popup-menu in the event list, an item lets the user save the

current event to a fault report and make a short note about the fault condition. The maintenance personnel will look at these reports and determine if they should be translated into new rules and become part of the knowledge-base. In this way, new knowledge is acquired in a top down fashion.

4. DATA-COLLECTION THROUGH EVENTS

The expert system works with events. All calculations of variations in a process variable over time is done in the control system, which has a powerful module library, and capacity to perform these tasks. In the control system, all data related to a certain object are grouped together in a signal package (see Fig. 3). This definition consists of an unique object id, a name and a number of signals.

NAME:	MOTOR A
ID:	O1
SIG 1:	TEMP HIGH
SIG 2:	DOWN
SIG 3:	OIL LOW

Fig. 3. Definition of an object

When a process signal in such an object crosses any of its limits, the whole signal package for this object is transmitted by the control system to the KBS (see Fig. 4). This event consists of the unique object id, a name, a time-tag, a trig and status of the signals. The trig is information about which signal caused the event.

ID:	O1
TIME:	08:33:24
TRIG:	1
SIG 1:	0
SIG 2:	0
SIG 3:	1

Fig. 4. Event from the control-system

An event is generated and displayed in an event list and the expert system will save the event in a history memory. The expert system is operator driven i.e. stays idle until the operating personnel asks for advice. This division of tasks between the expert system and the control system is natural and imitates the way the operating personnel analyse the process (see Fig. 1).

5. PREDICTING CONSEQUENCES IN THE PROCESS

One of the main points in the intelligent alarm handling tool is to predict consequences of events in the process, with time estimates. In the main connection of every rule, there is a time-interval defined, which says how long a certain state is valid. There are different kinds of time-intervals:

- if A then B *after* x min
- if A then B *within* x min
- if A then B *between* x and y min

The expert system will put all rules with valid alerts on the agenda and try to trig them, provided that all their conditions are satisfied. If the interval for a certain alert has expired, the rules with this alert cannot be put on the agenda. Each time the expert system is asked what consequences could be expected, all possible consequences based on the event of interest are calculated.

5.1. Example: Continuous Casting of Steel Billets

A steel plant has four objects; Motor A, Fan B, Fan C and the product, the steel Billet. Both Fan B and Fan C provide Motor A with cool air. These objects are defined in the database with their associated signals according to Fig. 6. Fan B and Fan C have two associated signals, On and Down. The signal On indicates order to run and Down means out of order. If On is false, the fan is off. If both signals are true, the fan has a fault. In the same way, Motor A has three associated signals; Temp High, Down and Oil Low. The last object is the product, the steel Billet, with two associated signals, Break and Thin. If the signal Break becomes true, the billet casting machine is shutdown.

NAME:	MOTOR A	FAN B	FAN C	BILLET
ID:	O1	O2	O3	O4
SIG 1:	TEMP HIGH	ON	ON	BREAK
SIG 2:	DOWN	DOWN	DOWN	THIN
SIG 3:	OIL LOW			

Fig. 6. Definitions of process-objects

The current knowledge-base consists of the three rules shown in Fig. 7. Rule 1 reads, if Fan C is out of order and Fan B is off then the temperature of Motor A will go high after 5 min. The two other rules can be interpreted similarly.

Rule 1 **Motor A Hot**
If *Fan C Down is true*
Then *Motor A Temp High is true*
After *5 min*
Provided *Fan B ON is false*

Rule 2 **Motor A Down**
If *Motor A Temp High is true*
Then *Motor A Down is true*
After *1 min*

Rule 3 **Steel Billet Break**
If *Motor A Down is true*
Then *Billet Break is true*
After *0 min*

Fig. 7. Rules in the knowledge-base

When the process is running, events containing these objects are transmitted from the control system (see Fig. 8). At a certain point, an event containing Fan C Down is True is transmitted. The operator sees the event in the event list and wants to know what consequences can be predicted. According to rule 1, Motor A will run hot after 5 min, according to rule 2 Motor A will go down after 6 min and at the same time, according to rule 3, the steel Billet will break. The user interface is shown in Fig. 9. To get further advice about actions to be taken, the user can press the action button and the expert system will provide him with a list of actions. The why-button will list all rules activated in the prediction.

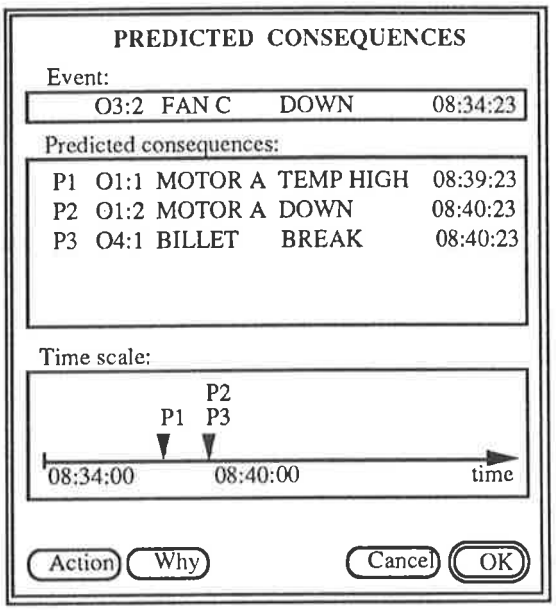


Fig. 9. Dialogue connected to consequence inquiry

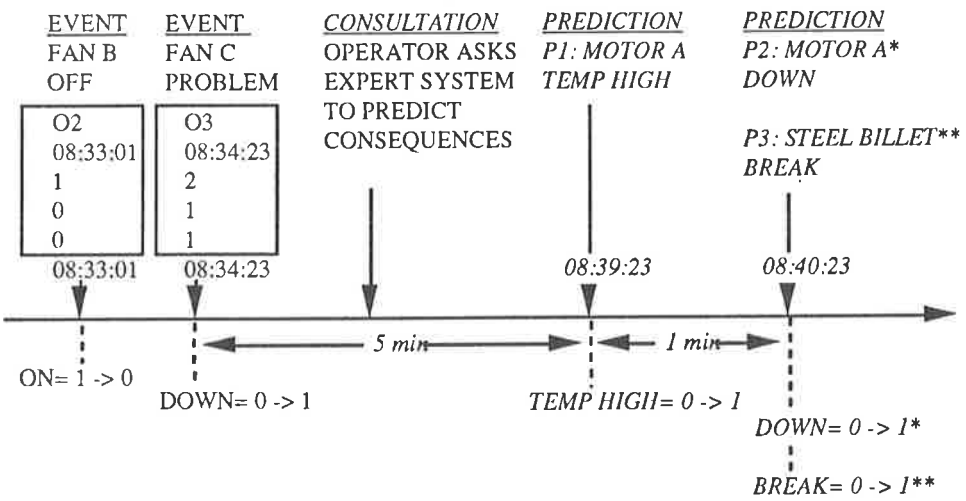


Fig. 8. Time-diagram over predicted consequences

6. CONCLUSIONS

The use of KBS such as the one described will lead to better and more consistent actions being taken by operating and maintenance personnel when problems arise in the process. Provided that there is an understanding of the causal relationship in the supervised process, the intelligent alarm handling tool will provide accurate predictions and explanations, with estimates of time delays. Because of the integration of the alarm handling tool with the reference part of FN, the operating personnel can directly associate the predicted and otherwise mentioned events with parts of the process and actual control equipment.

In order to build up the knowledge-base without having to describe the whole process in every detail, events are used. A basic structure is introduced, where the control system is calculating, comparing and analysing process variables in order to create events. These events are transmitted to the KBS. Imitating the operator, the KBS reason about these events to predict consequences. The KBS integrates a visual database of the process information, an expert-system, a knowledge-base and a communication driver to the control system. In this environment, the user can easily configure an intelligent alarm handler for various processes.
A KBS-tool as FN will lead to lower costs for maintenance and increased productivity due to fewer unplanned process shutdowns.

7. REFERENCES

Barklund, J. & Mildner, P. (1994). "A Real-Time Expert System for Process Supervision in Pulp Industry", Proc. IFAC.1994.

Årzén, K-E. (1990). "Knowledge-based control systems" Proc. Acc'90.