

Diagnostic Reasoning Strategies for Means-End Models

Jan Eric Larsson

Department of Automatic Control
Lund University
Box 118, 22100 Lund, Sweden
E-mail: janeric@it.lth.se

This is a *preprint* of an article published in *Automatica* 1994, and it is made available as an electronic reprint by permission of IFAC. The full reference to the published article is:

Larsson, J. E., "Diagnostic Reasoning Strategies for Means-End Models," *Automatica*, vol. 30, no. 5, pp. 775-787, 1994.

Diagnostic Reasoning Strategies for Means-End Models*

JAN ERIC LARSSON†

Three model-based, diagnostic methods, for measurement validation, alarm analysis, and fault diagnosis, are presented. Multilevel flow models are used to describe the process model. Thus, the three methods represent model-based approaches to tasks normally solved by rule-based expert systems.

Key words: artificial intelligence; modeling; (diagnosis; expert systems; model-based reasoning).

Abstract—This paper describes three diagnostic methods for use with industrial processes. They are *measurement validation*, *alarm analysis*, and *fault diagnosis*. Measurement validation means consistency checking of sensor and measurement values using any redundancy of instrumentation. Alarm analysis is the analysis of multiple alarm situations to find which alarms are directly connected to primary faults and which alarms are consequential effects of the primary ones. Finally, fault diagnosis is a search for the causes of and remedies for faults. The three methods use *multilevel flow models*, (MFM), to describe the target process. They have been implemented in the programming tool G2, and successfully tested on simulations of two processes.

1. INTRODUCTION

INDUSTRIAL PROCESSES can be described and modeled in several ways. However, most model types contain little or no *means-end* information, and thus provide no good support in diagnostic reasoning tasks.

This paper utilizes one type of explicit means-end models, *multilevel flow models*, (MFM), as developed by Lind (1990 a). Lind has suggested a syntax for a formal language and given general ideas on how to use the MFM representation. The contributions of this paper are descriptions of three methods for diagnostic reasoning using MFM:

- Measurement validation
- Alarm analysis
- Fault diagnosis

All three methods use MFM as a common base, and the alarm analysis and fault diagnosis can be used together during execution.

The measurement validation algorithm takes a set of measured flow values and uses any available redundancy to check consistency.

* Submitted to *Automatica* on the 29th of October 1992. Revised May 19th 1993. To appear.

† Department of Automatic Control, Lund Institute of Technology, Box 118, S-221 00 Lund, Sweden.

A single erroneous flow measurement will be marked and corrected; if there are several conflicting values, the consistent subgroups of measurements will be marked but no flow value corrected. The alarm analysis algorithm takes as input a set of alarm states such as *normal*, *low flow*, *high flow*, *low volume*, and *high volume*. The method recognizes the primary alarms, while the other alarms are *either* primary *or* consequences of the primary ones. The fault diagnosis algorithm uses an MFM model to produce a “backward chaining” style of diagnosis. The system will look for faults, provide explanations, and give remedies.

2. AN OVERVIEW OF RELATED WORK

The main contributions of MFM has been made by Morten Lind and his group. Lind (1990 a) describes the basics of MFM, while Lind (1990 b) contains Lind’s suggestion for a diagnostic system. Lind has also treated real-time diagnosis, Lind (1990 c), and design of operator interfaces, (Lind 1989). Lind’s group has developed a graphical interface, see Osman (1990), a STRIPS planning system, see Norby Larsen (1990), and a system for alarm analysis and fault diagnosis, see Creutzfeldt (1990).

MFM has been used in operator interfaces for fault diagnosis, see Duncan and Prætorius (1989), for constructing COGSYS diagnostic systems, see Sassen *et al* (1991, 1992) and Sassen and Jaspers (1992), and for chemical fault diagnosis, see Walseth *et al* (1992).

For readings on the classical forms of measurement validation, i.e., *data reconciliation*, see for example Mah (1990), or the overviews of Isermann (1984) and Frank (1990, 1991, 1992).

Classical methods of alarm analysis is overviewed in Lees (1983), while Modarres and Cadman (1986) and Padalkar *et al* (1991) describes efforts based on functional models,

and Oyeleye (1989) and Finch (1989) present the qualitative reasoning system MIDAS, based on graph representations.

The premier example of fault diagnosis is the MYCIN system, see Shortliffe (1976). Chung and Modarres (1989), Padalkar *et al* (1991), and Allen and Rao (1980) all describe fault diagnosis using functional models. Dvorak and Kuipers (1991), Dvorak (1992) describes the MIMIC fault diagnosis system, based on the QSIM language for qualitative simulation, see Kuipers (1984, 1986, 1989). The Inc-Diagnose system of Ng (1991) also uses a QSIM representation.

The Diagnostic Model Processor, (DMP), performs fault diagnosis with quantitative models in the form of set of equations, see Petti *et al* (1990), Petti and Dhurjati (1991), and Petti (1992).

Vina and Hayes-Roth (1991) use a set of different models to build a real-time knowledge-based system for fault diagnosis using a blackboard architecture. Struss (1987, 1991, 1992) describes a fault diagnosis system using multiple representation of physical structure and function, while Mariño *et al* (1990) describes a fault diagnosis expert system with a general multiple-view representation.

The DICE system, Jager (1990), Krijgsman *et al* (1990, 1991), and Krijgsman and Jager (1992), is a real-time expert system for control systems. It is based on a blackboard architecture and represents its knowledge with production rules. The RIGAS system, see Crespo *et al* (1991, 1992), is another real-time, blackboard architecture. The project described in Årzén (1989, 1990, 1993) and Årzén *et al* (1990) defines a general architecture for a real-time knowledge-based control system.

The methods described in this article has been presented in earlier papers; measurement validation in Larsson (1992 a), alarm analysis in Larsson (1991), and fault diagnosis in Larsson (1992 b). They have all been thoroughly described in the Doctor's thesis Larsson (1992 c), which is the basis for this article. Larsson (1992 d) contains more detailed descriptions of the implementation of the algorithms.

The advantages of the methods presented here are that they are based on a crisp and explicit description of means and ends, which is exactly the information needed in diagnostic reasoning, and that the graphical structure of MFM enables them to be very efficiently implemented, in comparison to the methods

mentioned above.

3. THE BASIC CONCEPTS OF MFM

An MFM model is a normative description of a system, a representation of what it has been designed to do, how it should do it, and with what it should do it. Thus, the three basic concept types of MFM are:

- Goals
- Functions
- Physical components

The *goals* are the objectives or purposes of the system, i.e., the ends that the constructors and operators want the system to reach. The *functions* are the means by which the goals are obtained, i.e., the powers or capabilities of the system. The physical *components* are what the system is constructed from, the equipment of which it consists.

The goals, functions, and components depend on each other in specific ways. Thus, in MFM there are different types of relations, that can be used to connect the objects:

- Achieve relations
- Condition relations
- Realize relations

An *achieve* relation connects a set of functions to a goal, and it signifies that these functions are used to obtain that goal. A *condition* relation connects a goal to a function, and signifies that the goal must be fulfilled in order for the function to be available. A *realize* relation connects a physical component to a function, and signifies that the component is used to realize or implement the function.

4. GOALS

The concept of a goal is central to MFM. Three different types of goals are recognized:

- Production goals
- Safety goals
- Economy goals

A production goal is used to express that to enable production, some specific process variable should be kept within a specified interval. A safety goal is used to express that for reasons of safe operation, some specific process variable should be kept above or below some value, or inside or outside an interval. An economy goal is used to express considerations of overall process optimization.

5. FUNCTIONS

The second important concept of MFM is that of a *function*. MFM describes the functional structure of a system as a set of flow structures. The levels are connected via achieve and condition relations, and the flow structures consist of connected flow functions. The types of flow structures currently treated in MFM are:

- Mass flows
- Energy flows
- Information flows

There are also several *function types*, which are treated in MFM. These are shown in Fig. 1. A detailed description of the different flow functions can be found in Larsson (1992 c). The given description of MFM is based on Lind (1990 a), while the following is original work.

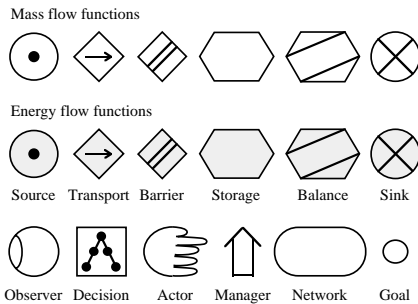


Figure 1. The symbols for the different MFM objects.

6. AN EXAMPLE OF A FLOW MODEL

The following example will be used to explain the basic concepts of MFM. The target process consists of a plate heat exchanger, see Fig. 2.

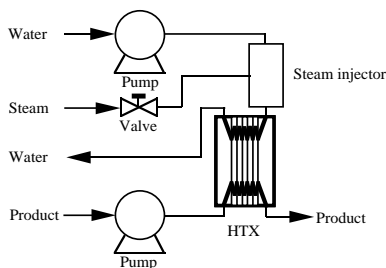


Figure 2. A heat exchanger system. The flowsheet shows how water is pumped through a steam injector, where it is heated with steam, and through a plate heat exchanger, where it heats the product.

This simple system serves quite well to explain the concepts of MFM. The primary goal is

to heat the product to a certain temperature, but a brief analysis shows that there are two subgoals also: having water and product available, i.e., bringing the media to the heat exchanger:

- G1: Heat product to certain temperature
- G2: Bring product to heat exchanger
- G3: Bring water to heat exchanger

The given example process is rather small, but there are many functions present:

- F1: Provide product
- F2: Transport product
- F3: Transfer thermal energy between media
- F4: Provide thermal energy
- F5: Transport thermal energy
- F6: Transport water
- F7: Provide water
- F8: Provide stream
- F9: Transport steam
- F10: Mix water and steam

The third type of objects are the physical components. Note that the product and water tanks do not actually appear in Fig. 2:

- C1: Product tank
- C2: Product pump
- C3: Heat exchanger
- C4: Water tank
- C5: Water pump
- C6: Steam system
- C7: Steam valve
- C8: Steam injector

These are the sets of goals, functions, and components. However, the relations between these objects are as important. First, the goal G1 is superior to G2 and G3, i.e., the latter are subgoals of G1. Thus, there is a *goal hierarchy*, formed by goal-subgoal relations. There are also relations between goals, functions, and components. For example, the heat exchanger component is used to *realize* the function of transferring thermal energy from water to product, and this function is used to *achieve* the goal of heating the product. In Fig. 3, both the goal hierarchy and the means-end relations are shown in a graph.

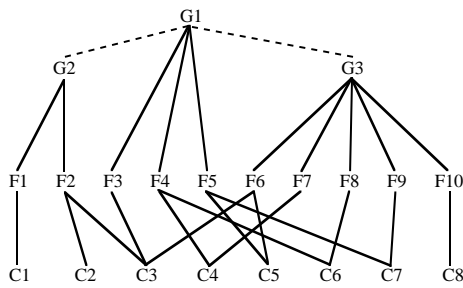


Figure 3. Goals, functions, components, and relations of the heat exchanger system.

As can be seen, the graph of objects and relations is quite complex, even for a small process as the one in the example. In an MFM model, the goals, functions, and relations are represented in a graphical language. A model of the example process is found in Fig. 4.

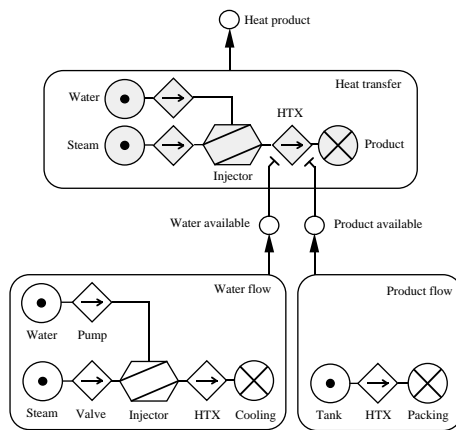


Figure 4. A flow model of the heat exchanger system. The goals and goal hierarchy are shown in the tree structure of the graph, while the functions are connected into flow paths in three networks. The topmost goal, to heat the product, is achieved by the heat transfer flow path, (upper network), while the subgoals, to bring water and product to the heat exchanger, are achieved by the water flow path, (lower left), and the product flow path, (lower right). The components and realize relations are not shown.

7. MEASUREMENT VALIDATION

Most industrial processes are equipped with a large number of sensors, of which several directly or indirectly measure the same variables. Especially when material and energy balance equations are taken into account, the total set of measurements commonly gives rise to redundancy, which can be used to check the consistency of the signals, i.e., to *validate* them.

Flow Semantics

In order to use MFM as a basis for measurement validation, a semantics has been defined that assigns flow values and grouping information to the different flow functions. Four of the flow functions have their attributes in common, and have thus been grouped into one class, called *flow carrier*. Sources, transports, non-forking balances, and sinks are all flow carriers. Storages, barrier functions, and forking balances are given a separate treatment.

Flow carriers have one flow value; a quantitative variable that corresponds to a physical flow of mass or energy. Storages have three flow values. There are input and output flows connected to corresponding measurements. There is also a third attribute, corresponding to the rate of change of the mass or energy contained in the storage. Barriers have no flow value, as they do not transport any matter or energy in working states. Forking balances have no flow value. Instead, the sum of the inflows should equal the sum of the outflows.

Generation of Measurements

The measurement validation method takes a set of flow signals as inputs. These inputs can be obtained in several different ways. They can be direct or filtered signals from sensors. It would be more probable, though, that they were the outputs of some low level data filtration on the direct signals, e.g., outputs from a Kalman filter or from some statistical algorithm. The signals could also come from any other hardware or software that produced flow values; the origin of the flow values does not matter for the method.

Consistent Subgroups

With use of the semantics above it is possible to split an MFM model, (i.e., a set of connected flow functions), into internally consistent subgroups. This is done via use of the following rules:

- If the flows of two connected flow carriers are equal, the flow carriers belong to the same consistent subgroup. If the flow values disagree, they belong to separate subgroups.
- If the input flow of a storage is equal to the flow of the flow carrier connected at the input of the storage, the input part of the storage belongs to the same subgroup

as the flow carrier. The corresponding is true for the output flow of a storage and the flow of the flow carrier connected to it.

- For each storage, the derivative of the volume may be measured. In this case a simple balance equation should hold between input, output, and derivative.
- For each balance, the sum of the inflows should equal the sum of the outflows.
- If the flow values of two flow functions agree, and the flow functions are in the same flow path, but separated by one or more inconsistent subgroups, they still belong to the same subgroup.

Application of the five rules above will enable a splitting of each flow into smaller groups with consistent measurement values. It should be noted that the last rule means that there can be groups with holes in them; they need not be directly consecutive. This is the case in Fig. 5.

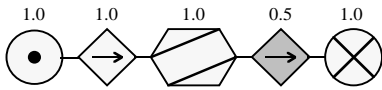


Figure 5. An example of a flow path. Here the flow functions form two consistent subgroups, where one group is surrounding the other. The three possible fault hypotheses are that the four measurements are correct and the one is faulty, that the one is correct and the four are faulty, and that all measurements are faulty.

Flow Propagation

The description so far has used the assumption that all flow functions have measurements. This is quite seldom the case. Many of the flow values needed in the algorithm will usually be unknown.

The MFM flow paths can be used, however, to propagate flow information, i.e., to guess the unknown flow values. The idea is quite simple: if an unknown flow value is connected to a known one, the known value is propagated to the unknown. Values that support each other have precedence over values based on single measurements.

Validation

Each flow value has a corresponding validated flow attribute. This is set according to the following rules:

- If a flow value is the only one in its subgroup, and it is surrounded by a consistent subgroup, its own flow

value is overridden, and the flow of the surrounding group becomes the validated flow of the flow function.

- In all other cases, the validated flow is equal to the corresponding measured flow.

In addition to the presentation of validated flow values, the implementation also presents some subgroup information to the user:

- A coloring scheme is used to separate the inconsistent subgroups in the graphical representation of the MFM model. Thus, the symbols of the flow functions in the different subgroups receive a light gray, gray, or dark gray rim, depending on which group they belong to.
- Each flow function that is alone in its group is highlighted in red.

The decision to explicitly mark all single subgroups is only one possible alternative of many. It is derived from the obvious possibility that the measured value in question probably is in error. It is very important to observe, however, that this is only probable, not certain. It is also possible, albeit with a lower probability, that all measurements of a larger, consistent subgroup is in error, while the single value is correct. The third possibility is that all the measurements are wrong. It is very important that these cases be taken into account when the results of the analysis are presented to the operator or higher level algorithm. This is the reason why the implemented system primarily displays information about the different consistent subgroups.

Examples of Measurement Validation

Let us demonstrate the method on a small example. The process consists of a storage tank, a pump, and two cylindrical tanks, see Fig. 6.

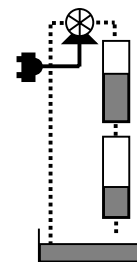


Figure 6. The tanks process. Water is pumped from a storage tank, to a cylindrical tank, from where it flows down into another tank, and then back to the storage again.

The main goal of the process is to keep the water level in the tanks at a specified level. This is achieved by the primary mass flow, i.e., the circulation of water. Here, the storage tank has been modeled both as a source and a sink. One of the transport functions, (the one that corresponds to the pump), depends on the subgoal that the pump motor has power, and this goal in turn is achieved by a secondary flow of electrical energy. The power support system is quite complicated but has been modeled simply as a source, a transport, and a sink. All this can be seen in the MFM model in Fig. 7.

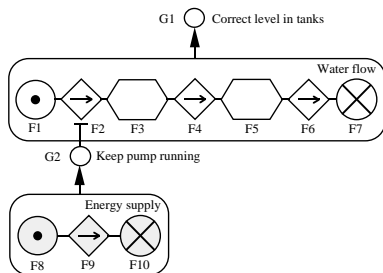


Figure 7. An MFM model of the tanks process. The main goal is to keep the level of the upper tank correct, and it is achieved by a water flow. In order for the transport function F2 to be available, i.e., to keep the pump running, energy must be supplied.

EXAMPLE 1

Now assume that flow measurements are available from the outflow of the storage tank, the throughput flow of the pump, and the inflow, derivative, and outflow of the upper tank, and that these flows have the following values.

flow of F1	$20 \times 10^{-6} \text{ m}^3/\text{s}$	(outflow from storage)
flow of F2	$10 \times 10^{-6} \text{ m}^3/\text{s}$	(flow through pump)
inflow of F3	$20 \times 10^{-6} \text{ m}^3/\text{s}$	(upper tank inflow)
deriv of F3	$0 \times 10^{-6} \text{ m}^3/\text{s}$	(volume change)
outflow of F3	$20 \times 10^{-6} \text{ m}^3/\text{s}$	(upper tank outflow)

Table 1. A set of flow measurement values.

The situation described in Table 1 is also shown in Fig. 8, where only the concerned flow functions are found. The flow values are shown above the flow function symbols; the storage function realized by the upper tank has three values, corresponding the inflow, derivative of volume, and outflow.

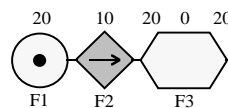


Figure 8. A flow path corresponding to Table 1. The flow of F2 disagrees from the rest, and there are two consistent subgroups, whereof one is single and surrounded.

The flow of F1 and all the flows of F3 agree, and thus they form a consistent subgroup. The flow of F2 disagree, however, forming another subgroup, with only one function in it. The system marks the two subgroups in different colors, thus notifying that there is an inconsistency. In this case it will also mark F2 specially, as it is a single function group, and the validated flow value of F2 will be set to $20 \times 10^{-6} \text{ m}^3/\text{s}$, (the flow of the surrounding group).

The consistent subgroup information has been shown in Fig. 8 with a shading system. In addition to this, the flow marking function F2 should also have a special marking, for forming a single function group. □

flow of F1	$20 \times 10^{-6} \text{ m}^3/\text{s}$	(outflow from storage)
flow of F2	$20 \times 10^{-6} \text{ m}^3/\text{s}$	(flow through pump)
inflow of F3	$10 \times 10^{-6} \text{ m}^3/\text{s}$	(upper tank inflow)
deriv of F3	$5 \times 10^{-6} \text{ m}^3/\text{s}$	(volume change)
outflow of F3	$5 \times 10^{-6} \text{ m}^3/\text{s}$	(upper tank outflow)

Table 2. A second set of flow measurement values.

EXAMPLE 2

Further assume the situation described by Table 2, which is also found in Fig. 9.

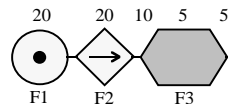


Figure 9. A flow path corresponding to table 2. Here there are two consistent subgroups which both consists of more than one measurement. The algorithm signals that the flows do not agree, but it cannot guess which ones that are correct.

Here we have two consistent subgroups, each with more than one measurement to support it. This situation is difficult to assess, as many sensor values must be wrong. The system will mark the two inconsistent groups, but will take no further action. Marking any particular value as wrong could be misleading and potentially dangerous. □

8. ALARM ANALYSIS

Most industrial processes are equipped with a large number of alarms. In a failure state it is quite usual that many of the alarms will trigger. Some of them will be directly connected to the primary sources of error, but others may be secondary, i.e., not connected to any failed equipment, but due only to consequential effects of the primary failures. In a failure state it is vital for the operator to separate the primary from the secondary alarms.

Failure Conditions for Flow Functions

Every flow function may or may not be alarmed, i.e., be connected to a corresponding part of the process, in such a way that a measurement tells whether the function is currently available or not. However, the alarm conditions are limited according to the following rules:

- A source is working if the current outflow F is less than the source's maximum capacity F_{cap} :

$$F \leq F_{cap}.$$

If this condition is not fulfilled, the alarm *locap* is true.

- A transport is working if the current flow F lies within an interval, specified in the design:

$$F_{lo} \leq F \leq F_{hi}.$$

If the flow F is below F_{lo} the alarm *loflow* is true; if it is above F_{hi} *hiflow* is true.

- A barrier is working if the current flow F is low enough, (approximately zero):

$$F \leq \varepsilon_1.$$

If this condition is not fulfilled, the alarm *leak* is true.

- A storage is working if the current volume V lies within a specified interval:

$$V_{lo} \leq V \leq V_{hi},$$

and the following inequality is fulfilled:

$$\left| \frac{dV}{dt} - F_i + F_o \right| \leq \varepsilon_1.$$

If the volume V is lower than V_{lo} , the alarm *lovol* is true, if it is higher than V_{hi} , *hivol* is true. If the expression within bars is less than $-\varepsilon_1$ the alarm *leak* is true; if it is larger than ε_1 the alarm *fill* is true.

- A balance is working if the following inequality is fulfilled:

$$|F_1 + F_2 + \dots + F_n| \leq \varepsilon_1.$$

If the expression within bars is less than $-\varepsilon_1$ the alarm *leak* is true; if it is larger than ε_1 the alarm *fill* is true.

- A sink is working if the current inflow F is less than the sink's maximum capacity F_{cap} :

$$F \leq F_{cap}.$$

If the condition is not fulfilled, the alarm *locap* is true.

A Method for Alarm Analysis

Failures can only propagate from flow function to flow function in certain ways. This is a consequence of the failure conditions described above. Thus, some primary failures in some types of flow functions may cause secondary failures in the connected functions, while failures in others will not. An example is given in Fig. 10, where a source F1 is connected to a transport function F2. This could correspond to, e.g., a tank connected to a pump.

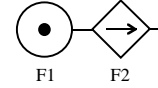


Figure 10. A connected source and transport. The source has a maximum outflow capacity, F_{cap} , and the transport has a working interval, $F_{lo} \leq F_t \leq F_{hi}$. If the wanted outflow of the source goes over its capacity or if the actual flow through the transport leaves the working interval, alarms occur.

The source has an output flow F_s , which must lie beneath the maximum capacity, F_{cap} , of the source. Thus, the following inequality must hold:

$$F_s \leq F_{cap}.$$

If it does not, either because the capacity, F_{cap} has fallen or because the output flow F_s has risen, the source will have a *locap* alarm.

The transport has a throughput flow F_t , which must lie in between a lower and an upper limit, F_{lo} , and F_{hi} , which are set during the design. Thus, the following inequalities must hold:

$$F_{lo} \leq F_t \leq F_{hi}.$$

If they do not, the transport will cause one of two alarms. If $F_t \leq F_{lo}$, the alarm will be *loflow*; if $F_t \geq F_{hi}$, the alarm will be *hiflow*.

Note the *normative* character of these assumptions. Here, the working interval of the transport must be decided during the design and modeling phase.

Assume further that the output flow or the source is controlled by the throughput flow of the transport, so that during normal operation:

$$F_s = F_t,$$

and that the working interval of the transport is small enough so that F_{cap} is always outside it during normal working conditions. Then the following analysis can be performed:

- If the capacity F_{cap} of the source should fall below the desired outflow, the transport will not get enough flow medium, and its throughput flow F_t will be forced out of and below the working interval.
- If, on the other hand, the current throughput flow of the transport should become higher than the upper limit of the working interval, i.e., F_t rises above F_{hi} because of some fault, this may or may not lead to the output flow of the source going above the maximum capacity F_{cap} .
- If the current throughput flow of the transport should fall below the lower limit of the working interval, the flow demanded from the transport will still be below the source's capacity, and the source will not be affected.

This analysis can be expressed very simply and crisply in two rules, where all the quantitative information is suppressed and only the alarm information used:

- A source *locap* alarm will force the connected transport to have a *loflow* alarm.
- A transport *hiflow* alarm may cause a connected source to have a *locap* alarm.

With the use of these two rules for how faults may cause other faults, and thus, how alarms may cause other alarms, any alarm situation concerning a source connected to a transport may be analyzed.

Assumptions of Flow Function Behavior

In the examples above, the different working conditions gave rise to a set of assumptions of how the flow functions involved will react when connected to each other. Using such assumptions for all flow functions, a small G2 program was written to automatically generate rules for all possible alarm causations, see

Larsson (1992 d). In fact, a set of rules was first generated by hand; and when the automatic rule generation program was ready, the previous hand-generated rules were checked and found to be correct.

Possible Secondary Alarms

The examples can be extended to all the allowed connections of flow functions. This will give a set of rules for how an alarm in one flow function may or will cause consequential alarms in the connected functions. A complete set of rules is as follows:

- A source *locap* will force the connected transport to have a *loflow*.
- A transport *loflow* may cause a storage connected at the inlet of the transport to have a *hivol*, and a storage connected at the outlet to have a *lovol*. It may cause another transport connected in the same direction via a balance to have a *loflow*. If the balance has no other connections the same alarm will be forced.
- A transport *hiflow* may cause a connected source or sink to have a *locap*. It may cause a storage connected at the inlet of the transport to have a *lovol*, and a storage connected at the outlet to have a *hivol*. It may cause a transport connected in the same direction via a balance to have a *hiflow*. If the balance has no other connections, the same alarm will be forced. It may cause another transport connected in the opposite direction via a balance to have a *loflow*.
- A barrier *leak* may cause a transport connected via a balance to have a *loflow*, or a *hiflow*.
- A storage *lovol* may cause an outgoing connected transport to have a *loflow*.
- A storage *hivol* may cause an incoming connected transport to have a *loflow*, and it may cause an outgoing connected transport to have a *hiflow*.
- A storage *leak* may cause the same storage to have a *lovol*.
- A storage *fill* may cause the same storage to have a *hivol*.
- A balance *leak* may cause a connected outgoing transport to have a *loflow*, and a connected incoming transport to have a *hiflow*.
- A balance *fill* may cause a connected incoming transport to have a *loflow*, and

a connected outgoing transport to have a *hiflow*.

- A sink *locap* will force the connected transport to have a *loflow*.
- An alarm in a network will force a function depending on this network to fail.

An Alarm Analysis Algorithm

The rules above can be used for automated alarm analysis. Given a set of alarms, it is possible to decide which of the alarms that must be primary ones, and which ones that *may* be secondary. It is important to observe, however, that one cannot be certain that a fault is indeed secondary; there might be multiple faults. Thus, the method will differentiate between positively primary alarms, and alarms that may be either primary or secondary.

As soon as a new alarm value is discovered, the corresponding alarm of the concerned flow function is set to an alarm value, e.g., a transport *loflow*, a storage *hivol*, or a balance *fill*. Then all rules that can be applied to the new alarm are tried, in order to see if they match the new situation. If so, the failure state of one or several flow functions may change, from *normal* to *primary failed* or *secondary failed*. It should be noted that the failure state *secondary* really means *primary or secondary*.

Unknown Alarm States

When some alarm states are unknown, the flow networks can be used to guess the missing values. This method will be called *consequence propagation*. The idea is simple. Given a set of known (primary and secondary) alarms and a set of unknown alarm states, the unknown values are filled in with secondary alarms according to a set of rules.

The rules used for this exactly correspond to the alarm analysis rules. Every such rule can be converted to a guessing rule, to be used in case the flow function in question is not given an alarm state from measurements.

Examples of Alarm Analysis

Let us now demonstrate the method on an example. Once again the tanks process will be used.

EXAMPLE 3

Assume that the functions F1, F2, F3, and F5 have measurements connected to their alarm states, while F4 and F6 have not.

Further assume that F1 has a *locap*, F2 a *loflow*, F3 a *lovol*, and F5 a *lovol* alarm. This alarm situation is shown in Fig. 11. The shading of the flow function symbols is used to indicate the failure state, (a dark shade means a primary alarm, a lighter shade a primary or secondary, and white a normal or unalarmed state).

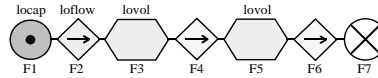


Figure 11. An alarm analysis situation. The functions F1, F2, F3, and F5 have *locap*, *loflow*, and *lovol* alarms. The alarms of F4 and F6 have been guessed. In this situation, the *locap* of F1 is the only primary alarm.

This would correspond to the plethora of alarms that could appear in, say, a complicated fault situation in a larger plant, although this situation is, of course, far simpler.

An application of the presented methods will result in that the *locap* of F1 must be a primary alarm, while the *loflow* of F2 and the *lovol* of F3 may be secondary. The consequence propagation implies that F4 and F6 might have had *loflow* alarms, had they been measured. Thus, assuming that F4 has a *loflow* alarm, the alarm analysis can also conclude that the *lovol* of F5 may be a secondary alarm. The result is that the *locap* of F1 is the only primary alarm, while all the others may be consequences of it. This has been shown with the shading in Fig. 11. F1 is the source function of the storage tank, and the sole cause of the fault situation could thus be that there is too little water in that tank. □

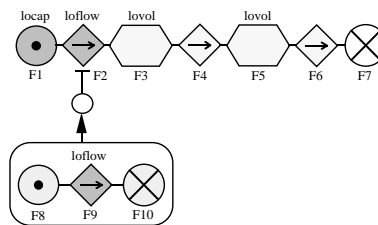


Figure 12. Another alarm analysis situation. Here the *loflow* of F2 must be primary, as there is an alarm in the achieving network, (the *loflow* of F9).

EXAMPLE 4

If the function F9 (transport of electrical energy to the pump motor) was to have an alarm also, the last rule in the rule set, (the rule concerning causation via the *achieve* and *condition* relations), would imply that F2 (the

pump) also had had a primary fault, i.e., there would now be at least two primary faults: no power supply for the pump and too little water in the storage tank, see Fig. 12. □

9. FAULT DIAGNOSIS

The classical use of knowledge-based systems in process control is to aid the process operator in diagnosing faults. This is usually done by a rule-based expert system, running the rules in backward chaining. The techniques are well-known, a good example being MYCIN, see Shortliffe (1976).

Problems with Rule-Based Expert Systems

The techniques of building and using standard rule-based expert systems are now more or less mature. They have, however, several shortcomings, some of which MFM can be used to remedy:

- Each rule base is specific for a certain process and task. Thus, a new system must be designed, built, and validated each time fault diagnosis is needed for a new process.
- A rule base may contain inconsistencies, and a large rule base most probably will do so. It is very difficult to guarantee consistency between the rules in an automatic fashion.
- A large rule base is difficult to overview, both in building and updating.
- A rule-based system can only diagnose the faults anticipated in the design of the rule base.

These shortcomings can be solved to a large degree by using MFM. The whole process of design, construction, and updating of the knowledge database becomes much more efficient, as the MFM models are easy to present graphically, and thus quicker to build and change. The other definite advantage of MFM is that consistency within the model is guaranteed. The graphical syntax makes inconsistencies in the database impossible. In addition, MFM can find any deviation from the working state.

The MFM Data Structure for Diagnosis

The working conditions for flow functions used in the fault diagnosis algorithm are the same as used in alarm analysis. Thus, each flow function might be in a normal or working

state, or have a fault, more or less directly corresponding to a *locap*, *loflow*, *hiflow*, *lovol*, *hivol*, *leak*, or *fill*.

The fault diagnosis algorithm must have a way of finding out the failure states of the physical components corresponding to the different flow functions. Thus, each flow function may have a question to be asked, or a check or test to be performed, in order to investigate the failure state of the function. Each flow function can also have a remedy of the fault, in the form of a text string to be output by the algorithm.

As an example, we will once again use the flow model of the tanks process. In order to enable a fault diagnosis, the different flow functions should be assigned questions. The source F1 could for example have the question “*Is there water in the storage tank?*” associated to it, together with the remedy “*Fill water in the storage tank.*”

The Search Strategy

An MFM model consists of information about the goals of a process, how these goals are achieved by networks of functions, how the functions depend on subgoals, and how they are realized by physical components. In a standard rule-based expert system, this information structure is implemented in rules, but in MFM it is explicitly described. Thus, a fault diagnosis can be easily implemented, as a search in the model graph. The strategy used for this search is as follows:

- The user chooses a goal for diagnosis. If this is a top-level goal, the whole model, (and thus the whole process), will be investigated. However, the goal chosen can also be a subgoal, in which case only part of the process will be diagnosed.
- The search propagates downwards from the goal, via achieve relations, into the connected network of flow functions, each of which is now investigated.
- Each flow function may have a diagnostic question, which is asked in order to find out whether the corresponding physical component is currently realizing the function, i.e., whether the function is available or not. Alternatively, there can be a rule or relation to a physical component, whereby information about the working order of the function may be found.
- If a flow function conditioned by a subgoal is found to be at fault, or has no means

of being checked, the connected subgoal is recursively investigated. If, however, a function is working, that part of the subtree is skipped.

An Example of Fault Diagnosis

Let us now demonstrate the method on a small example. Once again, the tanks process will be used.

EXAMPLE 5

Assume that the level of the upper tank is not correct, i.e., the goal $G1$ is violated, and that the user asks for a diagnosis of that goal. The algorithm starts at the goal $G1$, i.e., the topmost goal, and moves down into the network describing the mass (water) flow and checks the flow functions in turn.

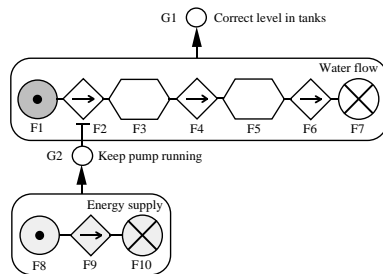


Figure 13. The diagnostic search started from the goal $G1$, followed the achieve relation down into the water flow network, and has reached the source $F1$.

The source $F1$ describes the source function of the storage tank, and is the first flow function to be reached by the search algorithm. The current position is marked in the graphic representation of the flow model, see Fig. 13. The source $F1$ has the following question associated to it:

Q: Is there water in the storage tank?

The user checks this and discovers that there is almost no water left in the storage tank. Thus he gives the answer 'no' and $F1$ is marked with a *locap*. The alarm analysis is activated but can draw no further conclusions, see Fig. 14.

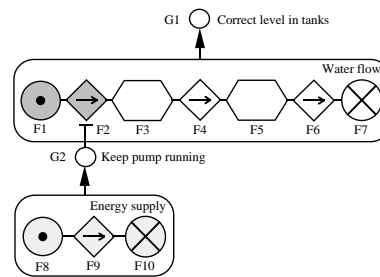


Figure 14. The diagnostic search has concluded that the source $F1$ is faulty. Then it has moved on to reach the transport $F2$.

The algorithm now moves on to $F2$ and asks the following question:

Q: Is the pump running?

Once again, the answer is 'no' and $F2$ is marked with a *loflow* alarm. The alarm analysis is activated and deduces that the *locap* of $F1$ must be a primary fault, while the *loflow* of $F2$ may be caused by the fault of $F1$, see Fig. 15.

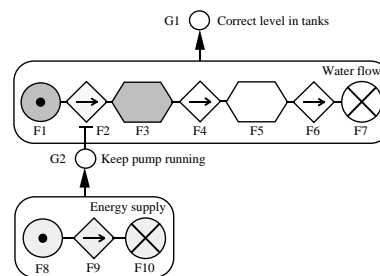


Figure 15. The diagnostic search has concluded that the transport $F2$ was at fault, and the alarm analysis signals that the fault of $F1$ is primary, while the fault of $F2$ may be secondary. Then the search has reached the storage $F3$.

The storage function $F3$ corresponds to the upper tank. It could have a question associated to it, that asked whether the volume of that tank was within the correct limits. Assume, however, that it is connected to a level alarm sensor. It will automatically be assigned a *lovol* alarm. Before the diagnosis has started, this alarm was considered primary, but once the *loflow* of $F2$ has been established, the alarm analysis algorithm decides that it may be a consequential fault.

The transport function $F4$ corresponds to the gravitationally caused outflow from the upper tank. It has no alarm and no question, so here the consequence propagation will be used to guess the alarm state, which will be a *loflow*. The rest of the flow functions in the flow path get their values in a similar manner.

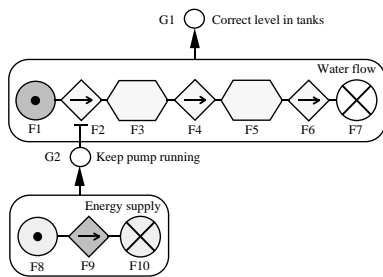


Figure 16. The diagnostic search has finished its investigation of the water flow network, and the alarm analysis concludes that there is one primary fault, three secondary, and two guessed faults, at F4 and F6. As the transport F2 was at fault and there is a condition relation, the search has continued down to the energy supply network and has reached the transport F9.

As there was a fault in F2, the algorithm now goes down in the subtree below it, and starts diagnosing the goal G2. It moves further down, finds the transport F9, see Fig. 16, which corresponds to the power switch of the pump, it asks the following question:

Q: Is the power switch on?

The user discovers that the power switch is in the 'off' position and answers 'no' to this question. The transport function F9 is marked with a *lofflow* alarm. The alarm analysis now deduces that the fault of F2 was indeed primary, as there is a fault in its support system. The total fault situation is thus that there are two independent causes of the level being to low; there is not enough water in the storage tank, and the pump power switch is not on, see Fig. 17. □

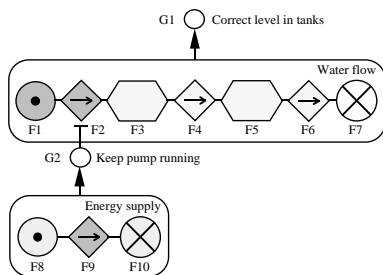


Figure 17. The state after the fault diagnosis. As the transport F9 is at fault, the fault in F2 is also a known primary fault. The user may now ask for explanations and remedies, and the algorithm will search through the graph and output the appropriate text strings from the failed flow functions.

EXAMPLE 6

The implemented system also allows the flow functions to have explanations and remedies associated with them, and these can now be asked for. In the example, the algorithm would go through the different *primary faults*. The remedy for F1 could be "Fill water in the storage tank," and the remedy for F9 "Switch on the power." □

Instead of writing out a remedy in text form, the system could also activate rules or procedures to perform any actions needed. These rules and procedures should be written in the general G2 rule format. □

10. IMPLEMENTATION

The algorithms described above have all been implemented in G2, and tested simulations of the tanks process and Steritherm. The latter is a widely used, moderately sized process for ultra-high temperature treatment of dairy products, see Figure 18. The MFM model of Steritherm consists of somewhat more than 100 objects.

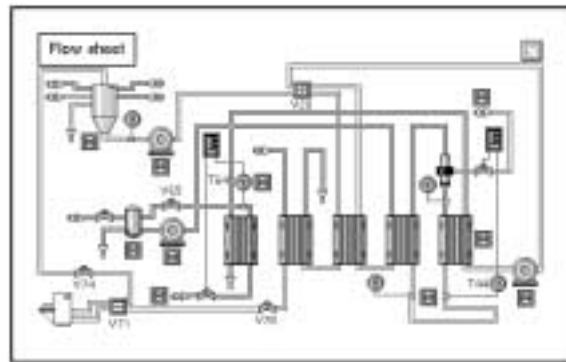


Figure 18. The Steritherm flow sheet, as it appears in the MFM toolbox implementation.

The toolbox implementation is reasonably small, see Table 3, and provides a toolbox for building and using MFM models in G2. The methods were successful in all test cases, but it should be noted that Steritherm is still a rather small-scale process. G2 was developed by Gensym Corporation, see Moore *et al* (1987, 1991).

Measurement validation	66 rules
Alarm analysis	93 rules
Fault diagnosis	19 rules

Table 3. The three methods have been implemented as G2 knowledge databases.

The algorithms are local and incremental. They work in real-time, and propagate information along static links only. This makes them very efficient, and the effort as a function of model complexity increases at worst linearly with the size of the MFM models. The local nature also has the benefit that feedback and recirculation loops pose no problems for the algorithms.

The efficiency of the fault diagnosis method is shown by a simple implementation in C, executing on-line using only tests and no questions. Searching through the whole Steritherm model, (more than 100 MFM objects), takes only 110 microseconds on a SPARC station 2, while finding a fault in the thermal energy network takes 25 microseconds. The worst case of executing time can easily be found out; for the Steritherm it is 145 microseconds. Clearly, the presented method enables knowledge-based fault diagnosis in an on-line control algorithm. All three methods may be equally efficiently implemented.

Together with definitions of MFM data structures and graphics, the three algorithms constitute an MFM toolbox, which enables a user to build MFM models of any suitable process. Once the MFM model has been constructed, all the three methods are immediately available, without any need for building a specific rulebase. Thus, the toolbox greatly simplifies the construction of knowledge-based systems for measurement validation, alarm analysis, and fault diagnosis.

11. CONCLUSIONS

The paper has presented three newly invented and implemented diagnostic methods for use with multilevel flow models, MFM. The methods use MFM as a database and performs measurement validation, alarm analysis, and fault diagnosis. They have been implemented in G2 and successfully tested on two processes. The search algorithms are all very efficient and work in real-time, and their sensitivity to large scaling of models is at worst linear. Together with an implemented toolbox and demonstrations, the project shows a good example of the usefulness and power of means-end models.

12. ACKNOWLEDGEMENTS

I would like to thank my supervisor, professor Karl Johan Åström, and the originator of MFM, professor Morten Lind, and Doctor Karl-Erik Årzén for inspiration and support. This project has been influenced by the Swedish IT4 project "Knowledge-Based Real-Time Control Systems," and I also wish to thank the members of the project group. The project has been supported by the IT4 project no. 3403 and the TFR project no. 92-956.

13. REFERENCES

- ALLEN, D. J. and M. S. M. RAO (1980): "New Algorithms for the Synthesis and Analysis of Fault Trees," *Ind. Eng. Chem. Fundam.*, **19**, 1, 79-85.
- CHUNG, D. T. and M. MODARRES (1989): "GOTRES: An Expert System for Fault Detection and Analysis," *Reliability Engineering and System Safety*, **24**, 113-137.
- CRESPO, A., J. L. NAVARRO, R. VIVÓ, A. GARCÍA, and A. ESPINOSA (1992): "RIGAS: an Expert Server Task in Real-Time Environments," *Proceedings of the 1992 IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in Real-Time Control*, Delft, Nederland, pp. 631-636.
- CRESPO, A., J. L. NAVARRO, R. VIVÓ, A. ESPINOSA, and A. GARCÍA (1991): "A Real-Time Expert System for Process Control," *Proceedings of the 3rd IFAC International Workshop on Artificial Intelligence in Real-Time Control*, Rhonert Park, Sonoma, California.
- CREUTZFELDT, J. (1990): *Sensorvalidering, alarmbehandling, og fejldiagnose i større processanlæg. (Sensor Validation, Alarm Analysis, and Fault Diagnosis in Large Processes)*, Ph. D. thesis, preliminary status report, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, Denmark, in Danish.
- DUNCAN, K. D. and N. PRÆTORIUS (1989): "Flow Displays Representing Complex Plant for Diagnosis and Process Control," *Proceedings of the 2nd European Meeting on Cognitive Science Approaches to Process Control*, Siena, Italy.
- DVORAK, D. L. (1992): *Monitoring and Diagnosis of Continuous Dynamic Systems Using Semiquantitative Simulation*, Doctor's Dissertation, AI 92-170, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, Texas.
- DVORAK, D. L. and B. KUIPERS (1991): "Process Monitoring and Diagnosis," *IEEE Expert*, June 1991, 67-74.
- FINCH, F. E. (1989): *Automated Fault Diagnosis of Chemical Process Plants Using Model-Based Reasoning*, Doctor's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

- 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.
- FRANK, P. M. (1991): "Enhancement of Robustness in Observer-Based Fault Detection," *Proceedings of the IFAC/IMACS Symposium SAFEPROCESS '91*, Baden-Baden, pp. 275–287.
- FRANK, P. M. (1992): "Robust Model-Based Fault Detection in Dynamic Systems," *Proceedings of the IFAC Symposium on On-Line Fault Detection and Supervision in the Chemical Process Industries*, University of Delaware, Newark, Delaware.
- ISERMANN, R. (1984): "Process Fault Detection Based on Modeling and Estimation Methods—A Survey," *Automatica*, **20**, 4, 387–404.
- JAGER, R. (1990): "Direct Real-Time Control using Knowledge-Based Techniques," *Proceedings of the ESS '90 Intelligent Process Control Design*, Ghent, Belgium.
- KRIJGSMAN, A. J. and R. JAGER (1992): "DICE: a Real-Time Toolbox," *Preprints of the 1992 IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in Real-Time Control*, Delft University of Technology, Delft, the Netherlands, pp. 637–641.
- KRIJGSMAN, A. J., R. JAGER, H. B. VERBRUGGEN, and P. M. BRUIJN (1991): "DICE: a Framework for Real-Time Intelligent Control," *Proceedings of the 3rd IFAC International Workshop on Artificial Intelligence in Real-Time Control*, Rhonert Park, Sonoma, California.
- KRIJGSMAN, A. J., H. B. VERBRUGGEN, P. M. BRUIJN, and E. G. M. HOLWEG (1990): "DICE: a Real-Time Intelligent Control Environment," *Proceedings of the ESS '90 Intelligent Process Control Design*, Ghent, Belgium.
- KUIPPERS, B. J. (1984): "Commonsense Reasoning About Causality: Deriving Behavior From Structure," *Artificial Intelligence*, **24**, 169–204.
- KUIPPERS, B. J. (1986): "Qualitative Simulation," *Artificial Intelligence*, **29**, 289–338.
- KUIPPERS, B. J. (1989): "Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge," *Automatica*, **25**, 4, 571–585.
- LARSSON, J. E. (1991): "Model-Based Alarm Analysis Using MFM," *Proceedings of the 3rd IFAC International Workshop on Artificial Intelligence in Real-Time Control*, Rhonert Park, Sonoma, California.
- LARSSON, J. E. (1992 a): "Model-Based Measurement Validation Using MFM," *Proceedings of the IFAC Symposium on On-Line Fault Detection and Supervision in the Chemical Process Industries*, University of Delaware, Newark, Delaware.
- LARSSON, J. E. (1992 b): "Model-Based Fault Diagnosis Using MFM," *Proceedings of the IFAC Symposium on On-Line Fault Detection and Supervision in the Chemical Process Industries*, University of Delaware, Newark, Delaware.
- LARSSON, J. E. (1992 c): *Knowledge-Based Methods for Control Systems*, Doctor's thesis, TFRT-1040, Department of Automatic Control, Lund Institute of Technology, Lund.
- LARSSON, J. E. (1992 d): "An MFM Toolbox," Technical report, TFRT-7493, Department of Automatic Control, Lund Institute of Technology, Lund.
- LEES, F. P. (1983): "Process Computer Alarm and Disturbance Analysis: Review of the State of the Art," *Computers and Chemical Engineering*, **7**, 6.
- LIND, M. (1989): "Human-Machine Interface for Diagnosis Based on Multilevel Flow Modeling," *Proceedings of the 2nd European Meeting on Cognitive Science Approaches to Process Control*, Siena, Italy.
- LIND, M. (1990 a): "Representing Goals and Functions of Complex Systems—An Introduction to Multilevel Flow Modeling," Technical report, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, Denmark.
- LIND, M. (1990 b): "Abstractions Version 1.0—Descriptions of Classes and Their Use," Technical report, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, Denmark.
- LIND, M. (1990 c): "An Architecture for Real-Time MFM Diagnosis," Technical report, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, Denmark.
- MAH, R. S. H. (1990): *Chemical Process Structures and Information Flows*, Butterworths, Boston, Massachusetts.
- MARIÑO, O., F. RECHENMANN, and P. UVIETTA (1990): "Multiple Perspectives and Classification Mechanism in Object-Oriented Representation," *Proceedings of the 9th European Conference on Artificial Intelligence*, Stockholm, pp. 425–430.
- MODARRES, M. and T. CADMAN (1986): "A Method of Alarm System Analysis for Process Plants," *Computers and Chemical Engineering*, **10**, 6, 557–565.
- MOORE, R. L., L. B. HAWKINSON, M. LEVIN, A. G. HOFFMANN, B. L. MATTHEWS, and M. H. DAVID (1987): "Expert System Methodology for Real-Time Process Control," *Proceedings of the 10th IFAC World Congress*, Vol 6, München, pp. 274–281.
- MOORE, R. L., H. ROSENOF, and G. STANLEY (1991): "Process Control Using a Real-Time Expert System," *Proceedings of the 11th Triennial IFAC World Congress 1990*, Tallinn, Estonia, pp. 241–245.
- NG, H. T. (1991): "Model-Based, Multiple-Fault Diagnosis of Dynamic, Continuous Physical Devices," *IEEE Expert*, December 1991, 38–43.
- NORBY LARSEN, M. (1990): "Strips as a Planning Method Within Abstractions and MFM Modeling," Technical report, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, Denmark.
- OSMAN, A. (1990): "The Interface Substrate," Technical report, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, Denmark.
- OYELEYE, O. O. (1989): *Qualitative Modeling of Continuous Chemical Processes and Applications*

- to Fault Diagnosis*, Doctor's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.
- PADALKAR, S., G. KARSAI, C. BIEGL, J. SZTIPANOVITS, K. OKUDA, and N. MIYASAKA (1991): "Real-Time Fault Diagnosis," *IEEE Expert*, June 1991, 75–85.
- PETTI, T. F., J. KLEIN, and P. S. DHURJATI (1990): "Diagnostic Model Processor: Using Deep Knowledge for Process Fault Diagnosis," *AIChE Journal*, **36**, 4, 565–575.
- PETTI, T. F. and P. S. DHURJATI (1991): "Object-Based Automated Fault Diagnosis," *Chemical Engineering Communications*, **102**, 107–126.
- PETTI, T. F. (1992): *Using Mathematical Models in Knowledge-Based Control Systems*, Ph. D. Dissertation, University of Delaware, Newark, Delaware.
- SASSEN J. M. A., A. OLLONGREN, and R. B. M. JASPERS (1992): "Predicting and Improving Response-Times of PERFECT Models," *Preprints of the 1992 IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in Real-Time Control*, Delft University of Technology, Delft, the Netherlands, pp. 709–714.
- SASSEN J. M. A. AND R. B. M. JASPERS (1992): "Designing Real-Time Knowledge-Based Systems with PERFECT," *Preprints of the 1992 IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in Real-Time Control*, Delft University of Technology, Delft, the Netherlands, pp. 625–630.
- SASSEN J. M. A., P. C. RIEDIJK, AND R. B. M. JASPERS (1991): "Using Multilevel Flow Models for Fault Diagnosis of Industrial Processes," *Proceedings of the 3rd European Conference on Cognitive Science Approaches to Process Control*, Cardiff, United Kingdom, pp. 207–216.
- SHORTLIFFE, E. H. (1976): *Computer Based Medical Consultations: MYCIN*, Elsevier Science Publishers B. V. North-Holland, New York.
- STRUSS, P. (1987): "Multiple Representation of Structure and Function," in Gero, J. (Ed.): *Expert Systems in Computer-Aided Design*, Elsevier Science Publishers B. V. North-Holland, New York.
- STRUSS, P. (1991): "What's in SD? Towards a Theory of Modeling for Diagnosis," "Working notes of the Second International Workshop on Principles of Diagnosis," CISE-Tecnologie Innovative, Milano.
- STRUSS, P. (1992): "What's in SD? Towards a Theory of Modeling for Diagnosis," in Hamscher, W., L. Console, and J. de Kleer, (Eds.): *Readings in Model-Based Diagnosis*, Morgan-Kaufmann Publishers, Inc., San Mateo, California.
- VINA, A. and B. HAYES-ROTH (1991): "Knowledge-Based Real-Time Control: The Use of Abstraction to Satisfy Deadlines," *Proceedings of the 3rd IFAC International Workshop on Artificial Intelligence in Real-Time Control*, Sonoma, California.
- WALSETH, J. Å., B. A. FOSS, M. LIND, and O. ÖGAARD (1992): "Models for Diagnosis—Application to a Fertilizer Plant," *Proceedings of the IFAC Symposium on On-Line Fault Detection and Supervision in the Chemical Process Industries*, University of Delaware, Newark, Delaware.
- ÅRZÉN, K. E. (1989): "Knowledge-Based Control Systems: Aspects on the Unification of Conventional Control Systems and Knowledge-Based Systems," *Proceedings of the 1989 American Control Conference*, Pittsburgh, Pennsylvania.
- ÅRZÉN, K. E. (1990): "Knowledge-Based Control Systems," *Proceedings of the 1990 American Control Conference*, San Diego, California.
- ÅRZÉN, K. E. (1992): "A Model-Based Control System Concept," Technical report, TFRT-3213, Department of Automatic Control, Lund Institute of Technology, Lund.
- ÅRZÉN, K. E., C. RYTOFT, and C. GERDING (1990): "A Knowledge-Based Control System Concept," *Proceedings of the ESS '90 Intelligent Process Control Design*, Ghent, Belgium.