Model-Based Alarm Analysis Using MFM

Larsson, Jan Eric

1991

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

Citation for published version (APA):
Larsson, J. E. (1991). *Model-Based Alarm Analysis Using MFM*. (Technical Reports TFRT-7470). Department of Automatic Control, Lund Institute of Technology (LTH).

Total number of authors:
1

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

CODEN: LUTFD2/(TFRT–7470)/1–10/(1991)

Model-Based Alarm Analysis Using MFM

Jan Eric Larsson

Department of Automatic Control
Lund Institute of Technology
January 1991

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name Technical report	
Date of issue January 1991	
Document Number CODEN: LUTFD2/(TFRT-7470)/1-10/(1991)	
Author(s) Jan Eric Larsson	*Supervisor*
	Sponsoring organisation

Title and subtitle

Model-Based Alarm Analysis Using MFM

Abstract

This report describes a method for model-based alarm analysis. It uses the multilevel flow modeling technique to separate primary alarms from alarms that may be consequences of these. The method has been implemented in the real-time expert system shell G2, and tested on two simple processes.

Key words
Alarm analysis, Knowledge-based systems, Model-based analysis, Real-time expert systems

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title	*ISBN*

Language English	*Number of pages* 10	*Recipient's notes*
Security classification		

The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

Model-Based Alarm Analysis Using MFM

Jan Eric Larsson

Department of Automatic Control
Lund Institute of Technology
January 1991

Department of Automatic Control
Lund Institute of Technology
Box 118
S–221 00 LUND
Sweden

Published 1991
Printed in Sweden

Model-Based Alarm Analysis Using MFM

Jan Eric Larsson M.Sc. L.Sc. B.A.

Department of Automatic Control, Lund Institute of Technology
Box 118, S–221 00 LUND, Sweden
Phone: +46 46 108795, E-mail: JanEric@Control.LTH.Se

Abstract

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. This paper describes a new method for automatically recognizing the primary failures. It is fairly general and built upon model-based reasoning. The modeling technique used is multilevel flow models (MFM), as suggested by Lind (1990b). First, the basics of MFM are described, and then an example of how such a model can be used in alarm analysis is given.

An Introduction to MFM

An industrial process can be modeled and described in several different ways. An operator often reasons about it in terms of its *goals* and the *functions* available for achieving these goals. The standard way of presenting the process for the operator is, however, with a process diagram, i.e., a formal description of topological and geographical properties, that contains little or no *means-end* information. It is therefore highly desirable to provide the operator with *functional models* of the plant, in addition to the topological ones. *Multilevel flow models* can be used for building functional models for industrial processes. The contributions so far have been made by Morten Lind at the Technical University of Denmark and Jens Rasmussen at the Risø National Laboratory, Rasmussen and Lind (1982), Lind (1990a, b).

In MFM there is a distinction between different views of a process. The functional view represents the *goals* of the process and the *functions* provided. The goals describe the operational objectives of running the process, e.g., achieving production, efficiency, and safety. The goals are achieved by functions or networks of functions, and are connected to these via *means-end* relations; thus, the goals and functions form a hierarchy of such relations.

The physical view describes what components are present in a system and how these connect into subsystems. The relations between objects in this view are *connection* relations and the relations between systems and subsystems are *part-whole* relations. They all describe the topological structure of the physical system. The components are connected to the functions via *realize* relations.

Thus, MFM models provide a description of structural and functional relations between objects, expressed in a graphical language. This representation may be used for several different tasks.

- *Error diagnosis.* The classical use of knowledge-based systems in process control is to aid the process operator in diagnosing errors. In MFM the functional dependencies are explicitly represented, so when a certain control goal fails, i.e., an error occurs, the MFM model will provide information on which functions may be in error, and thus, in which component sub-systems the reasons for the failure can be found.
- *Measurement validation.* If all measurements are propagated into the net of flow functions, inconsistent values of mass or energy flows can easily be found. Through further propagation of consistent information, a subset of singularly inconsistent measurement points may be computed.
- *Alarm analysis.* MFM models describe how different functions of the process depend on each other. By analyzing this structure, it is possible to say which alarms may and may not depend on each other. This will be more closely described in the rest of this paper.
- *Planning.* When the operator is planning different operations, he may use the MFM models to find out which goals depend on the function he plans to change or delete. If these goals may not be violated, something has to be done before the proposed action is performed.

The Basic Flow Functions

The MFM representation contains objects such as goals, abstract flow functions, networks, etc., and relations between these, e.g., connection of flow functions, achieving a goal by a certain function, and conditioning a flow function by a certain goal. A graphical language has been developed for describing this, see figure 1. The symbols are only a selection; more symbols can be seen in the examples below.

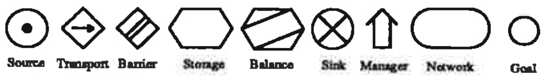

Figure 1. The basic flow function symbols.

The abstract flow functions are source, transport, barrier, storage, balance, and sink. Each of these can be concerned with either mass, energy, or information flows. Thus, there are three different types of flow functions. The manager describes a control function, and the network is a means of grouping several flow functions into a flow system.

Connection of Flow Functions

Flow functions may only be connected according to the following rules. The original formulation of Lind also allows storages and barriers to be connected, but this is disallowed here.

- Sources may only be connected to outgoing transports.
- Transports may be connected to sources, storages, balances, and sinks. They must be outgoing from sources and incoming to sinks, but may have any direction when connected to storages and balances.
- Barriers may only be connected to balances.
- Storages may only be connected to transports.
- Balances may only be connected to transports and barriers.
- Sinks may only be connected to incoming transports.

There are further rules. Flow functions of a certain type may only be connected to functions of the same type, and transports may not be connected so that any node is filled or emptied only.

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 working 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.

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

  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.

  $$|F_1 + F_2 + \cdots + F_n| \leq \varepsilon_1$$

  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.

It is important to observe that an MFM model is *normative* instead of *descriptive*, i.e., it describes how the process is intended to work, and failures in the process are found by noting differences between the intended model and the actual state. This guarantees completeness, i.e., we will catch every failure, if the model captures the important aspects of the process. However, all differences from the intended state are treated as failures, even if there can be other states in which the process is partly or fully operational. Thus, if several flow functions violate their conditions, they are failed by definition, even though the process might still be running in a new state.

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, primary failures in some types of flow functions may cause secondary failures in the connected functions, while failures in others may not. An example is given in figure 2, where a source F1 is connected to a transport function F2. This could correspond to, e.g., a tank connected to a pump.

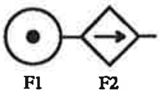

Figure 2. A connected source and transport.

Assume that the source F1 has an output flow F_s controlled by the throughput flow F_t of the transport F2. In the working state we have

$$0 \leq F_t = F_s \leq F_{cap},$$

where F_{cap} is the maximum capacity of the source. Let us further assume that the working interval of the transport,

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

is small enough so that F_{cap} is always outside it.

If the capacity F_{cap} of the source should fall below the desired outflow, the throughput flow F_t of the connected transport will be forced out of and below its working interval. This implies that a *locap* alarm of a source *forces* a *loflow* alarm in the connected transport function.

If, on the other hand, the current throughput flow of the transport should become higher than the upper limit of the working interval, this may or may not lead to the output flow of the source going above the maximum capacity. Thus, a *hiflow* alarm of a transport *may* cause a *locap* alarm in a connected source.

With the use of these two rules for how errors may cause other errors, and thus, how alarms may cause other alarms, any alarm situation concerning a source connected to a transport may be analyzed. If either function is in an alarmed state but not the other, that alarm is a primary one. If both functions are in an alarmed state, however, one of the situations above will apply, and accordingly, it is possible to tell which of the alarms that must be primary, and which that may be a consequence, a secondary alarm.

Possible Secondary Alarms

The example above can be extended to all the allowed connections of flow functions. This will give a list of rules for how an alarm in one flow function may or will cause consequential alarms in the connected functions. The complete list of rules are 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 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*.
- 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*.
- 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*.
- A balance *fill* may cause a connected incoming transport to have a *loflow*.
- 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.

These rules 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.

Unknown Alarm States

When some alarm states are unknown, the rules above can be used to deduce the missing values, with a consequence propagation strategy. 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 the rules. If an alarm will force another, the resulting alarm will certainly be known, but if the rule implies that an alarm may cause another, the resulting value will only be possible but not certain. The new alarm values may immediately be used in the continuing alarm analysis. By thus combining the alarm analysis rules with a consequence propagation strategy, unknown alarm states may also be treated.

An MFM Toolbox in G2

A set of definitions, rules, and procedures has been written in the G2 system, Moore *et al* (1987, 1990), to perform the method described above. The algorithm is implemented with two rule groups. One is concerned with the alarm analysis and consists of 37 rules, the other takes care of the consequence propagation and consists of 26 rules. These rule bases use a database of connected flow functions to yield a incremental and local algorithm, i.e., the alarm states of the flow functions are updated as soon as new values are available, and only local information about neighboring functions is needed.

Together with definitions, data structures and functions for MFM model design, error diagnosis, measurement validation, etc., this code will hopefully develop into a toolbox for functional modeling with MFM.

Measurement Errors

A common source of false or uncorrect alarms are errors in the sensors. In such cases the state of the process will not be correctly shown in the alarm presentation, and situations where this occur can be very dangerous.

The method described is not aimed at discovering uncorrect alarms. Instead it is sensitive to false alarms and largely dependent on correct sensor values. However, some cases can be detected as situations where the alarm state contains a sensor error. This is true when one alarm will *force* another alarm. If the latter alarm is not active, something is wrong with the measurements leading to one (or more) of the alarm states.

If, for example, a source and a transport is connected, as shown in figure 2, and the source has a *locap* alarm, the transport will have a *loflow* alarm. Should this not be the case, then either of the flow functions is in an incorrect alarm state, and the algorithm can issue a warning about this. However, it can give no answer as to which alarm that may be incorrect, and a more thorough measurement validation will have to be performed with other methods.

An Example — The Tanks Process

Let us now demonstrate the method on a small example. The process used consists of a storage tank, a pump, and two cylindrical tanks. Water is pumped from the storage tank and into the upper tank. From there it flows through a hole in that tank's bottom to the lower tank, and back to the storage tank, see figure 3.

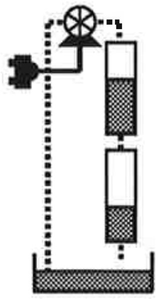

Figure 3. The tanks process.

The main goal of this 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 must have 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 figure 4.

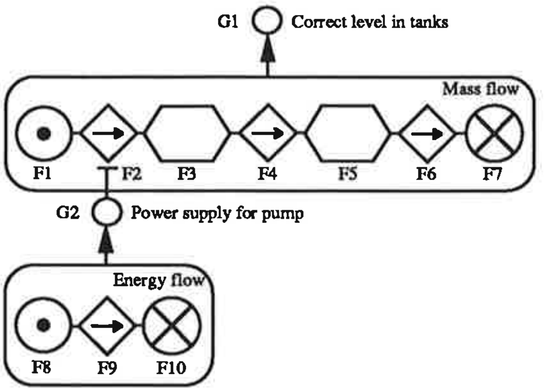

Figure 4. An MFM model of the tanks.

Assume that the functions F1, F2, F3, and F5 are alarmed, and let F1 have a *locap*, F2 a *loflow*, F3 a *lovol*, and F5 a *lovol* alarm. This situation would correspond to the plethora of alarms that could appear in, say, a complicated fault situation in a larger plant, although it is, of course, far simpler.

An application of the rules shown above 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 alarmed. 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. 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.

If the function F9 (transport of electrical energy to the pump motor) was to have an alarm also, the last rule above would imply that F2 (the pump) also had had a primary fault, i.e., there would now be at least two faults: no power supply for the pump and too little water in the storage tank.

If instead F3 had a *hivol* alarm, the algorithm would conclude that there were three primary alarms, the *locap* of F1, the *hivol* of F3, and the *lovol* of F5. This would correspond to a dynamic process state, where the pump flow and the volume of the lower tank were too low, while the volume of the upper tank was too high.

Experimental Experiences

The method described has been implemented and tested on two processes, the small laboratory process in the example above, and the Steritherm. The latter is a small-scale mass and energy flow process for ultra-high temperature treatment of diary products such as milk or cream, and it is the target process used in the Swedish IT4 project "Knowledge-Based Real-Time Control Systems," see Asea Brown Boveri AB (1988). The method could successfully distinguish between primary and secondary alarms in all the test situations. These situations were realistic, but it should be noted that the Steritherm is still a rather small process.

Conclusions

This paper describes an automatic method for distinguishing positively primary alarms from those that may or may not be primary. The method has been implemented in G2, and works under rather general conditions; there should be an MFM model of the process, this model must capture the important aspects of the process, and there should be no faults in the alarm measurements. Under these conditions the method provides a general solution to the problem of alarm analysis.

Acknowledgements

My thanks goes to Morten Lind, the originator of the MFM techniques, and to Professor Karl Johan Åström and Doctor Karl-Erik Årzén for inspiration. This project is an informal part of the Swedish IT4 project "Knowledge-Based Real-Time Control Systems," Asea Brown Boveri (1988), and further information can also be found in Larsson (1990).

References

Asea Brown Boveri AB (1988): *Knowledge-Based Real-Time Control Systems — IT4 Feasibility Study*, Studentlitteratur, Lund.

Larsson, J.E. (1990): "A Multilevel Flow Model of Steritherm," *Proceedings of the Nordic CACE Symposium*, Technical University of Denmark, Lyngby, Denmark.

Lind, M. (1990a): "Abstractions," Technical report, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, Denmark.

Lind, M. (1990b): "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.

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 (1990): "Process Control Using a Real-Time Expert System," *Proceedings of the 11th IFAC World Congress*, Vol 7, Tallinn, Estonia, pp. 234–239.

Rasmussen, J. and M. Lind (1982): "Coping with Complexity," Technical report, Risø National Laboratory, Roskilde, Denmark.