ALARM ANALYSIS ON LARGE SYSTEMS USING MULTILEVEL FLOW MODELS

Bengt Öhman

Department of Information Technology, Lund University
P.O. Box 118, SE-221 00 Lund, Sweden
bengt@it.lth.se

Abstract: Industrial processes are becoming more and more complex. When the number of alarms in a system increases beyond a certain level, there is a risk of overloading the operator with alarms. In order to improve the situation for the operator, some type of alarm reduction is needed. Until now, this has mainly been done via static priority levels. In this project, we are using Multilevel Flow Models as a tool to separate a large number of alarms into primary and secondary alarms, to show the operator the real causes of the alarms.

Keywords: Complex systems, computer-aided diagnosis, decision support systems, diagnosis, knowledge engineering, nuclear plants, real-time AI

1. INTRODUCTION

Modern industrial systems are equipped with a large number of sensors and alarms, partly because the computer-based control systems make it easy to add "standard alarms" for every single signal in the system. As the number of components in a system grow, the number of alarms also grow correspondingly. The large number of alarms is beneficial when only small disturbances occur, since it might give a more detailed view of the situation than a system with fewer alarms might.

However, if a larger disturbance occurs in the system, there may be a flood of alarms, and then it is not easy for the operator to spot the important ones. There are many cases in the industry where several hundred alarms have been fired in a few seconds. Since this is a known problem, various methods have been applied to simplify the situation for the operators. The most common approach is to assign a static priority to each alarm and enhance the visibility of the alarms with higher priority, and maybe suppress the alarms with lower priorities. Our method, alarm analysis based on MFM (see below), does not use priorities. Instead, the algorithm groups alarms into *primary* and *secondary* alarms, where the primary alarms are associated with the root cause for the disturbance, and the secondary, or *consequential* alarms, are associated with consequential faults in the system. Other methods, such as an expert system approach, can also be used to sort the alarms into primary and secondary, and to assign priority levels.

Expert systems have a few problems, though. For example, the construction of the knowledge base is a non-trivial task. You will also have to check the consistency of the rule base, so that no contradictions occur in the rules. A rule base for a large industrial system will also be very large, and the analysis of the

rules for any given situation can take a very long time to perform. There are no guarantee that the analysis will be performed within a reasonable time. An expert-system solution is therefore rarely used in practice.

Our method, however, is to use a model-based method based on *Multilevel Flow Models* (MFM). MFM is a graphical modeling language, which was invented by Morten Lind (1990). MFM shows the capabilities (called *functions*), and the causal connections in a system, and it also explicitly shows the intentional *goals* in the system. The goals in the model are the answer to the question "why," and the functions are the answer to the question "how."

Multilevel Flow Models is an ideal base for some diagnostic algorithms, which was shown by Jan Eric Larsson, who invented several algorithms operating on MFM models (Larsson, 1992). These algorithms include *Fault Diagnosis*, which tries to find the root cause for a fault, *Measurement Validation*, which tries to find suspicious sensor values using redundancy between sensors, and *Alarm Analysis*, which separates a large number of alarms into primary and consequential alarms. The algorithms operate similarly to a standard expert system, but the inherent structure of an MFM model gives these algorithms several nice properties. They are, for example, very fast (the response time for even a large model is typically less than a millisecond on a modern PC), and since the graphs are fixed, it is possible to calculate a worst-case execution time, which give the algorithms real-time properties. Diagnosis using MFM models has been used in some projects with good results. An example is described in (Larsson, *et al.*, 1997), where MFM was used to diagnose complications in a human body in an intensive care unit.

The algorithms have been improved over the years. The alarm analysis algorithm has been studied and improved by Fredrik Dahlstrand (1998), who extended it with fuzzy reasoning capabilities, and a new algorithm for failure mode analysis which is suited for on-line operation has been developed by Bengt Öhman (1999). This algorithm is used to predict future failures in a system.

In order to show the usability of Multilevel Flow Models for alarm analysis in large industrial processes, we are currently working on a project to build a model of a nuclear power plant. We will try to model a substantial part of the plant within the time frame of this project. Currently we have a demonstrator system that can take care of about 100 different alarms, and sort these into primary and consequential alarms, depending on the given situation. This demonstrator is only a first attempt at creating such a system, and we are working on implementing the same system by including our models and algorithms in commercial control systems, that enables them to be used in an existing industrial environment.

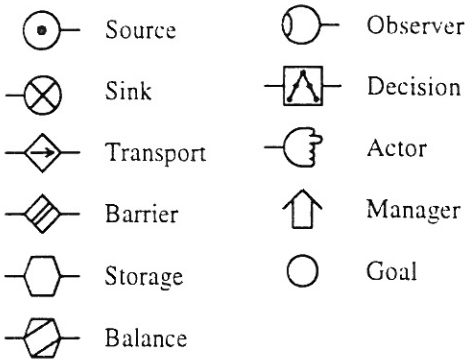

Figure 1. The MFM flow functions

2. ABOUT MFM

In order to show how our system works, it is necessary to first give a brief explanation of Multilevel Flow Models (MFM). MFM is a modeling method where the *goals* and *capabilities* of a system are shown in graphical models. The goals describe what the subsystems are supposed to do, for example, "keep the temperature at an appropriate level," or "provide electrical energy to the motor." The capabilities describe how the goals are achieved, and are modeled using *flow functions*. A function is an object which describes a certain capability of a component, such as "store fluid," "transport electrical energy," "provide cooling water," etc. The functions are connected into *flows*, and the flows are contained within *networks*, with one flow per network. The goals are connected to the networks, meaning that when all the flow functions in a network operate correctly, the connected goal is *achieved*. If one or several functions in a network do not operate correctly, the goal is *failed*. The different MFM functions are shown in Figure 1.

The purpose of the most important functions is as follows: A *source* can provide mass or energy, a *sink* can receive mass or energy, a *transport* can transport mass or energy, and a *storage* can store mass or energy. These functions do not depend on what type of physical component that accomplishes these tasks. A transport, for example, can be realized by a wire (transport of electrical energy), or a pump (transport of mass). A source can be realized by, for example, a battery or a power strip (source for electrical energy), a heating element (source for heat energy), or even the environment (as a source for, for example, oxygen).

These functions are connected via *networks* and *relations*. A network contains one group of connected flow functions. The network is connected to a goal via

an *achieve relation*. This means that, if all the functions in the network perform as designed, the goal is *achieved*. If, however, one or more of the functions is not working correctly, the goal is *failed*.

The *condition relation* is used to model dependencies between networks. In order for the connected function to be working, the connected goal must be achieved. The symbols for the networks and the different relations are shown in Figure 2.

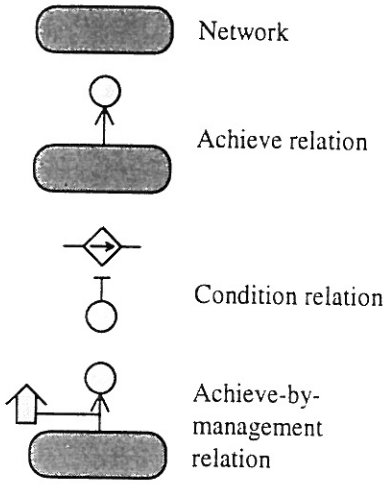

Figure 2. The MFM relations

In order to show how MFM is used, an example is presented here. Figure 3 shows a small sample process, which consists of a tray tank, a pump, and two cylindrical tanks. The pump pumps water from the tray tank to the upper tank, and from there the water runs down to the lower tank via a hole in the bottom of the tank, and from the lower tank the water runs back to the tray tank. The pump needs electricity to run, and is operated by a simple switch. The objective of this process is to keep the water level in the upper tank at a constant level.

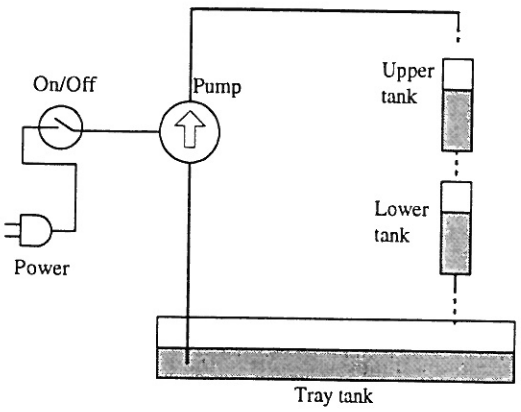

Figure 3. A sample process

In this process, the obvious top-level goal is to keep the water level in the upper tank at the correct level. There is also a subgoal, to provide electrical power to the pump. There are also some easily identifiable functions, for example, to transport water, to store water, to provide electrical energy, etc. It is clearly visible that the pumps ability to transport water is dependent on the goal "to provide electricity," so therefore there must be a condition relation between these objects. An MFM model of this system may then look like the one in Figure 4.

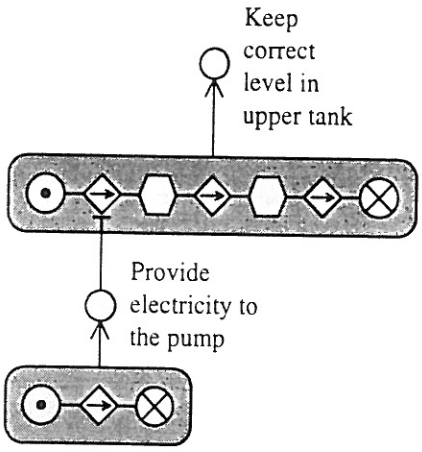

Figure 4. An MFM model of the sample process

In this MFM model, the upper network models the water flow through the process, and the lower network models the flow of electrical energy from the power supply to the pump. In the upper network, the functions from left to right are:

- A source, describing the tray tank's ability to provide water
- A transport, describing the pump's ability to transport water
- A storage, describing the upper tank's ability to store water
- A transport, describing the water flow from the upper tank to the lower tank
- A storage, describing the lower tank's ability to store water
- A transport, describing the water flow from the lower tank
- A sink, describing the tray tank's ability to receive water

In the lower network, the functions represent, from left to right, the power grid, the on/off-switch, and the pump motor, respectively. The condition relation in the model is connected to the lower goal, to provide electricity, and the function that represents the pump's ability to transport water. This means that in order for the pump to be able to transport water, the supply of electricity must be working. In this model, there are

three places where the electricity supply can fail: The power grid, the on/off-button, and the pump motor itself. Failures in these places can correspond to, for example, a power outage, the switch set to the "off" position, and a broken pump motor, respectively.

As can be seen from this model, a physical component may be represented by several MFM functions. The tray tank is for example modeled both as a source of water and a sink for water. The pump also occurs in two places, both as a transport of water, and a drain of electrical energy. The inverse, that an MFM function represent several components, may also be true. This means that it is possible to perform a top-down design, where the first model is a very simplified view of the process, and when needed, it is extended downwards via condition relations to more detailed networks describing the conditioned function.

3. ALARM ANALYSIS WITH MFM

The alarm analysis algorithm for MFM was first developed by Jan Eric Larsson (1992). The algorithm has then been refined by Fredrik Dahlstrand (1998), who also developed a version of the algorithm which operates using fuzzy logic, as opposed to the original algorithm which used only discrete logic. The alarm analysis algorithm is capable of separating alarms into *primary* and *consequential* (or *secondary*) alarms, when an alarm situation is input to an MFM model of the system. A primary alarm is an alarm that can not be explained by other alarms, while a consequential alarm can be explained as a consequence of some other alarm. The alarm analysis algorithm is designed for complex alarm situations, where many alarms go off at about the same time, and it also handles situations where there are multiple faults in the system. When such a situation occurs, the operator may have a hard time finding the failures that originally caused the whole chain of alarms.

This is a well-known problem, and there are some different solutions to it in the industry today. The most common method is to assign priority levels to each alarm, and to show the high-priority alarms to the operator in a different color, or on a separate alarm list, for example. This method has some disadvantages however; the priorities are set statically when the control system is designed. This means that the priorities are not adapted to each unique alarm situation, and may fail to give results that the operator needs in an extraordinary situation.

The alarm analysis with MFM, however, does not classify alarms into priority levels, but rather takes any given situation as input, and outputs the primary and secondary alarms. The primary alarms are then the originating causes for the whole situation, and the secondary alarms are consequential failures, which can be explained by the primary alarms. This is *not* the same as priority levels, so there is certainly a need to keep the priorities — some faults, even if they are consequential faults, may need to be addressed before dealing with the primary alarm.

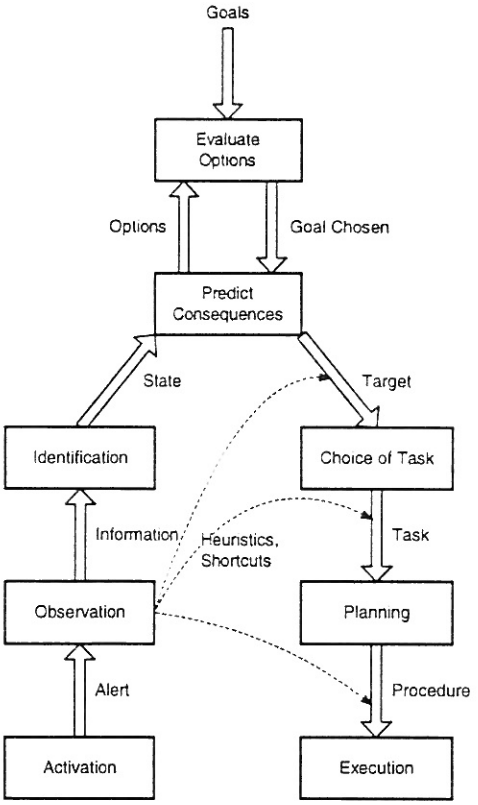

Figure 5. The decision ladder (Rasmussen, *et al.*, 1994)

The alarm analysis algorithm, as all other algorithms for MFM, operates on a high abstraction level. To illustrate this, the decision ladder described in (Rasmussen, *et al.*, 1994) can be used, see Figure 5. This figure describes the process of making a decision. First, there must be an event, the activation. Then, there must make an observation of this event, and it must be identified. After this, the situation can be examined, and, depending on the goals, which strategy to use can be decided. When the strategy is clear, it needs to be performed by breaking it down into steps, which are then executed. In this process, it is possible to take shortcuts — if a well-known event occurs, it is possible to take immediate action without going through the high-level reasoning stages at the top of the figure. If this reasoning model is applied to a control system, the observation and identification can be seen as the work done by the sensors and the fault detection and isolation algorithms. The execution is done by the different actuators in the process (servos etc.), and most of the planning is done by the control algorithms. At the uppermost levels of this

process, there is a human operator. The distinction between tasks that are performed by the operator, and tasks that can be automatically performed by the control system is not entirely clear — it all depends on the level of automation in the process.

The MFM algorithms come in near the top of the decision ladder, at the "identification," and "predict consequences" steps. This means that the whole chain before this has been performed. This includes many operations, for example, fault detection and isolation — see, for example, (Frank, 1996) — measurement validation, and removal of redundant sensor values, etc. The MFM algorithms assume that these steps have already been performed.

The input data to the alarm analysis algorithm is therefore *not* the different signals and levels from the process, but, rather, discrete signals saying that a function is *working* or *failed*, and that the state is *normal*, *too high* or *too low*.

4. ALARM ANALYSIS ON A LARGE SYSTEM

The project desccribed in this article aims to provide a system for alarm analysis for a large process. As this is a project with limited time and resources, there is small hope of putting anything into actual operation, but the result will be a demonstrator system showing that it is possible to do what is claimed to be possible. The target system is one of the Swedish nuclear reactors. This is a suitable target system for several reasons:

- The system is very large and complex, containing many sensors and alarms
- The system mainly consists of different flows of mass and energy, which makes it suitable for MFM

It may sound ambitious to model a whole power plant, but since the MFM models and algorithms operate on a very high abstraction level, it is fully possible to do. The models can also be constructed in a top-down approach, so that a coarse model of the main parts of the system is easy to build. More detailed models can then be constructed by using the top-down approach. The level of details necessary to get good results from the analysis depends mainly on the kind of input that is received — a subsystem modeled in exquisite detail will not provide better results if there are no sensor values or alarms that can provide input values to the MFM functions. On the other hand, if the model is made on a too high level, there might be many alarms which affect the same MFM function. This might or might not be desirable, depending on the situation. This kind of problems have to be analyzed and resolved when modeling a target system.

It is also possible to break down the MFM model into several models, each modeling one part of the system, to make the modeling easier. Each of these models can then be used to supervise its own subsystem. From these submodels, a top-level MFM model can collect summary alarms, which may be presented in the main model. In order to view the details, it is always possible to look at the more detailed model for the affected subsystem.

The input to the system consists of the different alarms that can be emitted from the control system. The alarms generally contain all the information that is needed for the MFM algorithms, that is, the affected component, and the state of the component. A typical alarm may be "there is a too high temperature at component x in subsystem y." From this information, it is possible to associate the alarm with the correct function (or functions) in the MFM model, and set these to a *high* state, and perform an alarm analysis. Since the analysis is so fast, the analysis can easily be performed for each received alarm, which makes the software simpler than if another approach, such as queueing all alarms and performing an analysis at regular intervals, was used. Tests show that for a model with around 500 functions, it is possible to perform several thousand analyzes per second on a modern PC, which would suffice for most, if not all, alarm situations.

Whenever a new analysis is performed, the list of primary and secondary alarms is updated. Since the alarm analysis is not sensitive to the order of alarms, it is not necessary to receive the alarms in any particular order. This means that if all the consequential alarms come first, the alarm list will briefly show an erroneous result, but, since all information is not yet available, this is the best diagnosis that can be performed with the information available at that time. When the primary alarm comes in, however, a new analysis will be performed, and the correct conclusions will be displayed instead.

Modeling of a complex system is always a challenging task, but the nature of the MFM models makes it possible to create models, even of very large systems, with a moderate effort. Even though it is possible to create an expert system that can perform the same tasks as the MFM algorithms, such a system is very hard to create if the system is too complex.

5. CURRENT STATUS

A demonstrator system exists, consisting of an MFM model, the MFM algorithms and a user interface built by using process pictures. With this system, it is possible to analyze different (simulated) alarm situations and observe the results of the alarm analysis in the

process pictures. In these pictures, each component affected by any active alarm will be marked with a color indicating whether it is affected by an alarm determined to be primary or secondary. It is also possible to show the alarm list in text format, where the primary alarms are colored red and the secondary alarms are colored yellow.

This demonstrator system has mainly been used to test different methods of connecting the alarms from a typical alarm list to the functions in the MFM model. Based on the results of this work, the same functionality, but with a larger MFM model, is now being implemented in the framework of a commercial supervisory control and data acquisition (SCADA) package, in order to show that it is possible to implement these methods in an industry standard environment. The current project is planned to be finished at the end of the year 2000, and by that time, there will be a working demonstrator product that can handle a variety of alarm situations.

6. ACKNOWLEDGMENTS

This project is funded by the Swedish National Board for Industrial and Technical Development (NUTEK), with project number P10504–1, and is part of the research program for complex technical systems. I would like to thank Jan Eric Larsson and Fredrik Dahlstrand for their support, and I would also like to thank Anu Uus for proofreading this paper.

References

Dahlstrand, F., "Alarm Analysis with Fuzzy Logic and Multilevel Flow Models," *Proceedings of the Eight Annual International Conference of the British Computer Society Specialist Group on Expert Systems*, ES98, Cambridge, England, pp. 173–188, 1998.

Frank, P. M., "Analytical and Qualitative Model-Based Fault Diagnosis — A Survey and Some New Results," *European Journal of Control*, vol. 2, pp. 6–28, 1996.

Larsson, J. E., *Knowledge-Based Methods for Control Systems*, Doctor's thesis, TFRT-1040, Department of Automatic Control, Lund Institute of Technology, Lund, 1992.

Larsson, J. E., "Hyperfast Algorithms for Model-Based Diagnosis," *Proceedings of the IEEE/IFAC Joint Symposium on Computer-Aided Control Systems Design*, Tucson, Arizona, 1994.

Larsson, J. E., "Diagnosis Based on Explicit Means-End Models," *Artificial Intelligence*, vol. 80, no. 1, pp. 29–93, 1996.

Larsson, J. E., B. Hayes-Roth, D. M. Gaba, "Goals and Functions of the Human Body: An MFM Model for Fault Diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 27, no. 6, pp. 758–765, 1997.

Lind, M., "Representing Goals and Functions of Complex Systems — An Introduction to Multilevel Flow Modeling," Technical report, 90-D-38, Institute of Automatic Control Systems, Technical University of Denmark, Lyngby, 1990.

Öhman, B., "Failure Mode Analysis Using Multilevel Flow Models," *Proceedings of the Fifth European Control Conference*, ECC '99, Karlsruhe, Germany, 1999.

Rasmussen, J., A. M. Pejtersen, and L. P. Goodstein, *Cognitive Systems Engineering*, John Wiley & Sons, New York, 1994.