AN ARCHITECTURE FOR AUTOMATIC ON-LINE SUPERVISION

Fredrik Dahlstrand

Department of Information Technology
Lund Institute of Technology
Lund University, SE-221 00 Lund, Sweden
fredrikd@it.lth.se

Abstract: Most diagnostic AI-algorithms are in need of high-quality observations to produce reliable results. The same is true for any human being, who is performing diagnostic reasoning. It is impossible to state an accurate and reliable diagnosis using false or faulty observations. In this paper an architecture to support the diagnostic algorithms based on multilevel flow models is presented. However, any diagnostic algorithm would benefit from using this architecture. The diagnostic algorithms based on multilevel low models already exist. This paper outlines the requirement for a system that shields the diagnostic algorithms from the uncertainties in the process.

Keywords: Artificial intelligence, alarm systems, expert systems, fault diagnosis, models, and supervision.

1. INTRODUCTION

Large industrial processes are in general equipped with many sensors, which are providing information about the state of the process. This sensor information is used by the process supervision system, which is responsible for monitoring the process and report process anomalies (faults) to the process operator. When a fault occurs in the process, this may, due to the many measurements, cause a cascade of alarms. This causes a heavy cognitive load on the process operator, who may make more or less serious mistakes, since it is difficult to get an overview of the situation.

For example, the Three-Mile Island incident (see for example Perrow, 1999) is a good example of a situation where the plant operators were overloaded by the amount of information that was provided from the supervision system. An operator in a similar situation may be helped by a system that aids him in focusing on the important alarms, find the root cause (causes) of the alarm situation, and if possible advice the operator how to handle the situation.

In this paper, an architecture for an on-line supervision system is outlined. The fault diagnosis, measurement validation, and alarm analysis algorithms used here are based on *Multilevel Flow Models* (MFM). MFM has successfully been used for modeling the Steritherm process, which is a process for sterilization of liquid food products, for example milk (Larsson, 1992). In the Guardian project MFM was used for monitoring patients in an intensive care unit (Larsson and Hayes-Roth, 1998). In an on-going project, MFM is used for supervision of the Barsebäck nuclear power plant (Larsson and Öhman, 1998). Multilevel flow models and the algorithms for measurement validation, alarm analysis, and fault diagnosis will be described later.

In this paper a *failure* means any deviation from the expected (designed) behavior of the process, and a *fault* is the failure of a single physical process component. An *alarm* is a fault that has been detected by the

supervision system and reported to the process operator. The *components* are the physical process components, for example, pipes, valves, and storage tanks.

1.1. Requirements for an On-Line Supervision System

The first task of a supervision system is *fault detection*. This may not always be an easy task, for example, some of the important process variables may not be possible to measure but have to be calculated, or estimated from other process measurements. Some of the process measurements may be unimportant, or even unnecessary in different modes of plant operation.

When a fault is detected the supervision system has to find which component or components that have failed. The task of determining the location of the fault is often referred to as *fault isolation*. The fault isolation may show that more than one component has failed, which may not be the true explanation. For example, if a pump that is providing water to a storage tank should fail, the fault isolation shows that both the pump and the storage tank have failed. The fault isolation will report two failures when there is only one *root cause*: that is, the failure of the storage tank is a *consequential fault*, due to the failure of the pump, which is the root cause. In a large process the situation is even worse, a single failure can cause a long chain of consequential faults. To sort out the root causes, from the consequential faults is a task for a *diagnostic* algorithm. It is important to notice that a consequential fault in a process may be more important than the root cause. For example, if the feed water pump in a nuclear power plant fails, then the reactor core does not get enough cooling, and even though it is a consequential failure, it is more important to prevent the potential nuclear meltdown than to get the feed water pump running again.

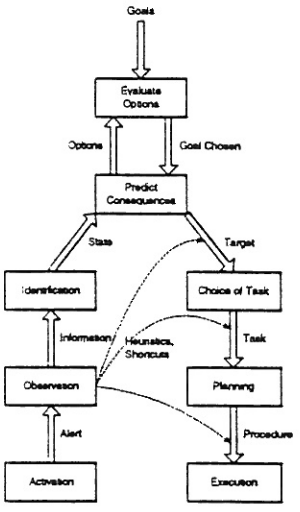

Fig. 1. The decision ladder shows the tasks that must be performed in order to detect, diagnose, and correct a failure situation. From (Rasmussen *et al.*, 1994).

Figure 1 shows Rasmussen's decision ladder (Rasmussen, *et al.*, 1994), where the first steps (activation, observation, and identification) are the fault detection and isolation. When the operator is aware of the failure state of the plant he has to find an action to restore the plant back to a non-failure state. This is done by evaluating his options, predict the consequences of these options, plan a series of actions, and finally execute the plan. This is the sequence of actions that must be taken both by humans and automatic supervision systems. However, through experience (both real and programmed) it may be possible to skip some of the steps in the decision ladder. For example, a simple PID-controller would, based on observations, execute a task without any consideration to effects in other parts of the plant. This is referred to as a *heuristic shortcut*.

1.2. Measurement Disturbances

Almost all process measurements are affected, to some degree, by (Rosén, 1998):

- *Noise*, that is, electromagnetic and other disturbances that cannot be explained by the process characteristics.
- *Missing measurements*, that is, for some reason the measurements are unavailable or incorrect.
- *Outliers*, that is, single measurements that are obviously not correct.
- *Drift*, that is, a cumulative error, over time, of the sensor.

Any useful plant supervision system must be able to handle these uncertainties in the process. If this is not done, the diagnostic algorithms cannot provide the operator with reliable information.

Furthermore, any on-line supervision system should not alert the operators when there is no alarm as for example, noise or badly tuned alarm limits may cause the alarm system to trigger false alarms. This is a potentially dangerous situation since the operator may learn over time that these particular alarms are always false, and ignore them when there is a real failure. The opposite is also a bad scenario. If the alarm limits are set too wide the alarm system may never trigger, that is, never lets the operator be aware of the failure.

2. MULTILEVEL FLOW MODELS

To describe the multilevel flow models (MFM), a very small process, the *two tank process*, will be used. This process, see Figure 2, contains a pump, which pumps water from an external water supply to a tank. The water is then transported to a second tank through a pipe. Finally, the water is let out from the system to

the environment. Furthermore the pump requires electrical energy to run.

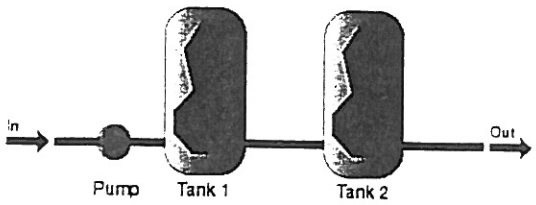

Fig. 2. The two tank process.

MFM is a graphical language for describing the goals and intended functionality of an (industrial) process, and it was invented by Professor Morten Lind (Lind, 1990). The major difference between MFM and other modeling approaches is that it models the *means-end* dimension rather than the *part-whole* dimension. MFM uses *goals* and *functions* to model the processes. The goals describe the purpose of the process — the answer to the question "why?" — and the functions describe the functionality needed to fulfill these goals — the answer to the question "how?". The functions describe the capabilities of the process in terms of flows of mass and energy.

Fig. 3. Some of the MFM symbols.

There are ten different MFM functions, but only four are needed to describe the process in Figure 2. The four MFM functions in this example are (see Figure 3):

- A *source* represents a component's capability of providing mass or energy (a battery).
- A *transport* represents a component's capability of transporting mass or energy (a pump).
- A *storage* represents a component's capability of storing mass or energy, (a tank).
- A *sink* represents a component's capability of receiving mass or energy (a lamp in an electrical circuit)

An MFM model of the process in Figure 2 is shown in Figure 4. The main goal of the process is to keep correct levels in both tanks. There is also a subgoal (provide electrical energy to the pump) that needs to be fulfilled in order for the pump's capability of transporting water to be available. The MFM functions are connected together to describe flows of mass or energy. Such a connection of MFM functions is called a *network*. A network is connected to a goal by an *achieve relation* (the thick arrows). In order for a goal to be achieved (fulfilled), all functions in the connected network must be available. The thin line between goal G1 and the leftmost transport in the mass network, in Figure 4, is a *condition relation*, which means that in order for the transport to be available the goal must be fulfilled.

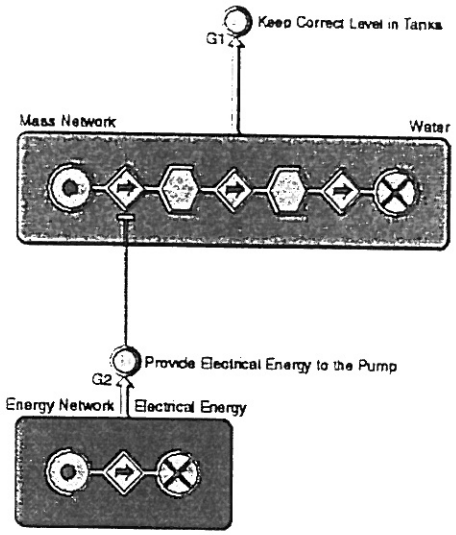

Fig. 4. MFM model of the tank process in Figure 2.

The mass network describes the flow of water in the process, and the MFM functions represent, from left to right, the external water supply (a source), the pump (a transport), the first tank (a storage), the pipe between the two tanks (a transport), the second tank, (a storage), the outflow from the second tank (a transport), and finally the environment that receives the water (a sink). The energy network describes the flow of electrical energy. These MFM functions represent from left to right, the whole power grid (a source), the cord between the outlet and the pump (a transport), and the pump (a sink). Note that the pump has two capabilities, both to act as a transport of water and as a consumer of electrical energy.

3. DIAGNOSTIC MFM ALGORITHMS

Each of the algorithms below are highly dependent on the quality of the process measurements to perform well. This is not only the case for diagnostic algorithms based on MFM; all diagnostic algorithms have this drawback. For a more detailed description of the fault diagnosis and measurement validation algorithm see (Larsson, 1996). The alarm analysis algorithm is described in (Larsson, 1996; Dahlstrand, 1998), and the fault mode analysis algorithm is described in (Öhman, 1999).

3.1. Fault Diagnosis

The MFM models can be used to perform a fault diagnosis. Associated with each goal is a boolean value that shows if the goal is fulfilled or not. If the goal is not fulfilled, a downward search in the MFM model is performed to find the cause. By asking a series of

questions, the fault diagnosis algorithm finds the cause of the goal no being fulfilled. These measurements may be either questions to the operator or process measurements. The search starts from the top goal and asks a question for each function in the network connected to that goal. If any function is failed, or its state is unknown, and it is connected to a condition relation then the fault diagnosis algorithm continues its search downward by diagnosing the goal connected to that condition relation. For example, if the two tanks are empty, that is, the top goal is not fulfilled then the questions and answers would be:

1. *"Is the environment providing enough water?"— yes.*
2. *"Is the pump pumping water?" — no.*
3. *"Is electrical power available from the grid?" — yes.*
4. *"Is the switch on the pump turned on?" — no.*

The result of this fault diagnosis would be that the levels in the tanks are not correct since the pump is not pumping water, which in turn is caused by the pump being turned off.

3.2. Alarm Analysis

Another diagnosis algorithm using MFM is the *alarm analysis* algorithm. Alarm analysis is the task of sorting a cascade of alarms into *primary alarms*, that is, the root causes of the situation, and *secondary alarms*, which are the effects of the root causes.

To accomplish this, each MFM function is associated with different *alarm states* that describe how a function has failed. Some of these alarm states are shown in Table 1. For example, a source may either be in a state of *low capacity*, which means that it is not being able to provide as much mass or energy as required, or it may be in a state of *normal capacity*, that is, working as expected.

Table 1 The alarms states for some of the MFM functions

MFM Function	Alarm States		
Source	Low Capacity	Normal Capacity	—
Transport	Low Flow	Normal Flow	High Flow
Storage	Low Volume	Normal Volume	High Volume
Sink	Low Capacity	Normal Capacity	—

These alarm states are determined by comparing the measurements from the process monitoring system against thresholds, which may be either static or dynamic, see Figure 5. Static thresholds are constant over time, but they may have different values, depending of the operating state of the plant. Dynamic thresholds are based on, for example, prediction using model-based methods (Åström, 1980; Frank, 1990), neural networks or statistical methods, which reflect the expected behavior of the measurements. The alarm analysis algorithm is not dependent on the order in which the alarms arrive to the supervision system.

The measurements are connected to the MFM functions. For example, in the two tank process one could measure the levels in the two tanks. These measurements will then be connected to the two storages in the MFM model in Figure 4. Let us consider a situation, when the levels in both tanks are too low. Then the alarm analysis algorithm will start from the knowledge gained from the measurements, that is, both storages are in the *low volume* alarm state. Then the states of the functions that are not measured are *guessed*, based on the known states of the MFM functions. For example, if we know that the level in a tank is too low then the logical guess, if no other information is available, would be that the outflow from the tank is also too low. In the case when the alarm state for the two storages is *low volume*, the alarm analysis algorithm would say that the root cause is that the level in the first tank is too low, and that the alarm from the second tank is just a consequence of that alarm. Furthermore the states for the other functions in the network are guessed, thus giving a prediction of state of the water flow in the process.

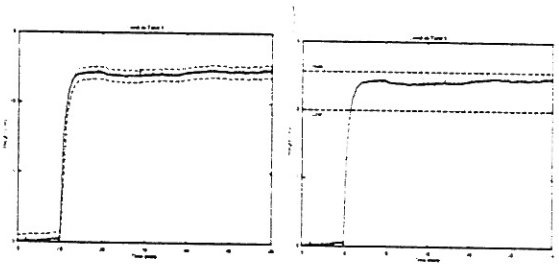

Fig. 5. Example of dynamic thresholds (left) and static thresholds (right).

3.3. Measurement Validation

The *measurement validation* algorithm works by grouping together MFM functions with consistent measurements. If there is more than one group in a network then at least one of the groups contains a sensor that is not working, or there is a leak in the flow. The task of measurement validation is important because if the sensor readings are not reliable then the other diagnosis algorithms, or a human plant operator, cannot do a reliable evaluation of the situation.

3.4. Failure Mode Analysis

The MFM *failure mode analysis* algorithm is used to predict the alarm situation, both on-line and off-line. For example, if the level in the first tank is too low and the level in the second tank is normal, one could guess that, unless the level in the first tank gets back to normal, the level in the second tank will sooner or

later also be too low. This can be used as an early warning system, letting the plant operators know the consequences of the current failures, and get an rough estimate of time before those failures will affect critical parts of the system.

4. ARCHITECTURE

To be able to aid the operator in performing an accurate evaluation of an on-going failure situation, the diagnostic algorithms must retrieve as correct measurements as possible from the process. It is not realistic to expect that any fault diagnosis algorithm will perform accurately if it does not know what is happening in the process, just as the plant operator must receive high quality information from the process to know the state of the process. There is on-going research in fault-tolerant control systems (Blanke, 1999), which are somewhat similar in architecture but does only involve diagnostic algorithms on a lower level, that is, the diagnostic algorithms is more tightly connected to the control system.

Figure 6 shows an outline for an architecture for performing automatic on-line supervision of a process. It consists of two major parts. The lower part contains algorithms that work on single process measurements, that is, measurements that concern small parts of the process. These methods are used for removing as much as possible of the noise and other measurement disturbances. The upper part contains the diagnosis algorithms, which have more knowledge of the process. The lower part can be divided into three layers.

1. Low-level data treatment.
2. Feature extraction.
3. High-level data treatment.

The upper part contains the algorithms described in Section 2. Furthermore, there is a layer, the conversion layer, responsible for converting real world quantitative measurements into the qualitative measurements that some of the diagnostic algorithms use, for example, the alarm analysis algorithm.

The *low level data treatment* layer is responsible for performing simple cleaning of the data by using standard signal processing techniques.

In the *feature extraction* layer characteristics of the measurements are identified, for example, detection of trends and broken sensors. Example of techniques used could be statistical methods, or neural networks. This layer is also responsible for making quality controls of the measurements. Example of validation techniques could be found in (Höjstrup, 1993; Horn, *et al.*; 1997; Mah, 1990).

Based on the information from the feature extraction layer the *high-level data treatment* layer is responsible for repairing the faulty data, if possible. For example, if a broken sensor is detected, the algorithm here could, for example, make predictions or estimations by using other process variables. Whether it is possible or not to repair the measurements, the diagnostic algorithms must be notified of that something is wrong with the measurements. If the data cannot be repaired it should be considered unreliable and should not be used for diagnostic purposes.

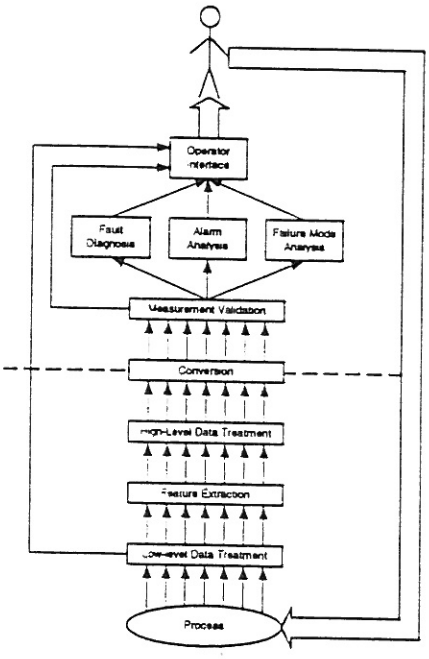

Fig. 6. An architecture for performing automatic on-line plant supervision. This is an extension of the architecture presented by Larsson (1992, 1996).

In the *conversion* layer redundant sensors are merged together. Process variables that cannot be measured directly are calculated, or estimated, from existing measurements. This layer is also responsible for making simple temporal synchronization. Furthermore, the quantitative process measurements are converted into qualitative measurements for those diagnostic algorithms that require it, for example, the alarm states used by the alarm analysis algorithm. The alarm states may be either crisp logic alarm states, or fuzzy alarm states.

When the raw process measurements are cleaned, validated, and converted, they are provided to the diagnostic algorithms, which were described in Section 3.

The maybe most important layer of all is the *operator interface*. It is important that the user interface of the supervision system is designed in such way that the operator has all relevant information easily available without causing a cognitive overload. It is also important that the extra information provided to the operator from the supervision system does not burden the operator unnecessarily, and non-critical process information should only be presented when the operator asks for it.

5. CONCLUSIONS

The architecture for on-line supervision in this article is intended for use with the MFM algorithms explained in Section 2. However, any fault diagnosis algorithm benefits from this architecture, since it is impossible to do an good fault diagnosis with faulty process observations. The choice of algorithms in the lower part of the architecture is more process dependent. As a rule of thumb one could say that, the design of lower part algorithms and models requires more component knowledge, whereas the design of higher part algorithms and models require more process knowledge.

The most important, and maybe the most difficult, part of the supervision system is the man-machine interface. It does not matter how well the filtering, data-repair, and diagnosis algorithms work if the results cannot be presented to the operator in a way that is easily understood. It is also important that the automatic supervision system suppresses process information that is not important in an on-going failure situation.

The automatic supervision system has it strongest points in that it never gets tired, nervous, or stressed, and never makes mistakes because of this. Furthermore, although the benefits of an automatic supervision system may be limited in the day to day operation, it is an important aid for the operators when a rare or once-in-a-lifetime failure situation occurs.

One interesting extension of this architecture would be to add a plan generation system. The first simple approach would be to link the supervision system to a database of premade plans, if such plans exist, and let the supervision system automatically retrieve plans based on the results from the fault diagnosis algorithm. A more challenging task would be to let the supervision generate new plans based on the results from the fault diagnosis algorithm.

6. ACKNOWLEDGMENTS

I would like to thank my supervisor Jan Eric Larsson and my colleague Bengt Öhman at the Department of Information Technology for their excellent support and inspirational discussions. I would also like to thank Anu Uus for her help with proof-reading.

REFERENCES

Åström, K.J. (1980). "Maximum Likelihood and Prediction Error Methods," *Automatica*, vol. 16, pp. 551–574

Blanke, M. (1999). "Fault-tolerant Control Systems," in Frank, P. M. (Ed.), *Advances in Control — Highlights of ECC'99*, Springer Verlag, London, pp. 171–196.

Dahlstrand, F. (1998). "Alarm Analysis with Fuzzy Logic and Multilevel Flow Models", Proceedings of the 18th Annual International Conference of the British Computer Society Special Group on Expert Systems, ES98, Cambridge, England, pp.173-188.

Frank P. M. (1990). "Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy—A Survey and Some New Results," *Automatica*, vol. 26, no. 3, pp.459–474.

Höjstrup, J. (1993). "A Statistical Data Screening Procedure," *Measurement Science and Technology*, vol. 4, no. 2, pp. 153–157.

Horn, W., S. Miksch, G. Egghart, C. Popow, and F. Paky (1997). "Effective data validation of high-frequency data: Time-Point-, Time-Interval-, and Trend-Based Methods," *Computers in Biology and Medicine, Special Issue: Time-Oriented Systems in Medicine*, vol. 27, no. 5, pp. 389–409.

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

Larsson, J. E. (1996). "Diagnosis based on explicit means-end models," *Artificial Intelligence*, vol. 80, no. 1, pp. 29–93.

Larsson, J. E. and B. Öhman (1998). "Model-Based Alarm Analysis for Large Plants," invited paper, Proceedings of the International Conference on Systems, Signals, Control, Computers, Durban, South Africa.

Larsson, J. E. and B. Hayes-Roth (1998). "Guardian: An Intelligent Autonomous Agent for Medical Monitoring and Diagnosis," *IEEE Intelligent Systems*, vol. 13, no. 1, pp. 58–64.

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

Mah, R. S. H. (1990). *Chemical Process Structures and Information Flows*, Butterworths, Boston, Massachusetts.

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

Perrow, C. (1999). *Normal Accidents — Living with Hihg-Risk Technologies*. Princton University Press, New Jersey.

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

Rosén, C. (1998). *Monitoring Wastewater Treatment Systems*, Licentiate Thesis, Department of Industrial Electrical Engineering and Automation, Lund Institute of Technology, Lund.