A GENERIC INFERENCE ENGINE FOR ALARM FILTERING IN AUTOMATED PRODUCTION SYSTEMS

A.K.A. Toguyéni, A. Ghariani, E. Craye

Laboratoire d'Automatique et d'Informatique Industrielle de Lille
(L.A.I.L.- CNRS UMR 8021),
Ecole Centrale de Lille, B.P. 48, 59651 Villeneuve d'Ascq Cedex, France)
Mail : {Armand.Toguyeni/Abdallah.Ghariani/Etienne.Craye}@ec-lille.fr
Telephone number: (33) 3-20-33-54-16, Fax number: (33) 3-20-33-54-18

Keywords: alarm filtering, supervision, alarm processing, recovery actions.

Abstract: The goal of this paper is to propose new alarm filtering techniques for Intelligent Alarm Processing Systems (IAPS). A basic idea of our approach of alarm filtering is to use a type of knowledge model (the functional graph) to define synthesis alarms. Using causal inhibition principle, detail alarms are inhibited by synthesis alarms to reduce the number of alarm presented to the operator. Moreover, we propose a hierarchy of concepts to characterise the status of an alarm. This hierarchy enable us to propose integrated filtering techniques that take into account both filtering due to system properties (validation, causal inhibition) and the interaction of the operator with the alarm system to sort presented alarms (acquitement, suppression).

1 Introduction

Today alarm systems are widely used in the industry to pilot and to survey complex industrial plants. In spite of the number of supervision software on the market, very few progresses have been achieved to reduce the complexity of operator tasks in case of fault situations. The goal of this paper is to propose new features to design Intelligent Alarm Processing Systems (IAPS) [7, 9]. Alarm filtering is one of the features of a tool we are developing in the European CHEM[1] project. The main objective of CHEM is to develop a set of new Decision Support Systems (DSS) to allow an intelligent supervision of actual plants.

The paper is organised as follows. The first section will set the problem of fault treatment from piloting operator point of view. The goal of this section is to define what type of information need these operators and what are the problems they meet with actual alarm systems. In the second section, we will introduce different types of alarm that must be distinguished when we build an alarm system. The third section will introduce the Functional Graph and several concepts we apply for alarm filtering. We will notably propose a hierarchy of these concepts to build robust alarm filtering algorithms. The fourth section presents a prolog prototype that implements the alarm filtering techniques proposed in this paper.

2 Fault treatment for piloting operators point of view

Most of the supervision systems of actual complex Automated Production Systems are based on alarm systems. The roll of an alarm system is mainly to report process misbehaviours (faults) to the operators. In this case, the operators can select an appropriate procedure to apply corrective actions to avoid the propagation of the fault consequences. The main problem of actual alarm systems is to provide too much information to the operators. Several reasons explain this fact. The first reason is the way alarms are triggered off.

In general, an alarm is triggered off when a fault event occurs in the process. When only one event characterises a fault or when the event can be generated by a critical fault, an automatic procedure is used to treat the fault. In this case, some authors propose to trigger off informative alarms. Their goal is to report to the operators the fault context (the plant state), the unavailability of some plant resources and the new configuration of the plant. In this case the operators have nothing to do except to take into account these information to perform future piloting or corrective actions. However, in general, the relation between a fault and an event is not bijective. Indeed, from faults viewpoint, several events are generally necessary to characterise a fault. At the opposite, a misbehaviour event can be produced by different faults. This leads some designers to link an alarm to each of the events produced by a fault. The idea is that the operators can reason from these alarms to diagnose the fault that have produced them. This kind of reasoning is difficult in case of multiple faults because all the events produced by an alarm do not necessary occurred at the same time and then can interleave with the events that characterised another fault. Moreover, causal relations in a process imply fault propagation notably if corrective actions are not taken quickly. In consequences, they generate alarms avalanches that increase the difficulty for the operator to diagnose the root fault of process misbehaviour.

[1] CHEM: "Advanced Decision Support System for chemical and petrochemical processes". Project is funded by the European Community under the Competitive and Sustainable Growth programme of the Fifth RTD Framework Programme (1998-2002) under contract G1RD-CT-2001-00466. See www.cordis.lu or www.chem-dss.org

Another cause of alarm overload is the occurrence of false alarms. For example, in nuclear power plant system it is classical to have a few hundred alarms on operators control consoles in normal working. False alarms are generally due to the use of static thresholds to monitor continuous variables of the plant. These thresholds are fixed with regard a specific working mode. In a lot of plants they are chosen with regard to a steady state of the plant. Consequently, they are not suitable for transitory modes such as starting working mode or closure mode. Another source of false alarms is systematic maintenance operations on the plant resources. Since maintenance mode is not taken into account, when maintenance agents operates on a resource they modify the value of the variables that characterise this resource. False alarms can also be triggered off by a bad use of the process. As an example, let us consider the case of a machine tool stopped by a maintenance operator in order to perform systematic maintenance operations. If the Control-command tries to run the program of the machine and gets no response, an alarm would be triggered off to warn the operators about the faulty state of the machine.

However, the overload of piloting operator consoles is not only cause by alarms. As invoked in above paragraphs, an Automated Production System (APS) must be piloted according to the production objective and to all the events that enable the operator to identify the state of its plant. Moreover, the operators must always know which resources are available and which of them are unavailable. For example, informative alarms in case of automatic reconfiguration or alarms induced by maintenance operations aim to inform of the status of concerned resources. Graphical user interfaces of supervision systems are generally designed to allow the operators to understand quickly which resources are operational.

3 Taxonomy of alarms and alarm filtering

To reduce the number of alarms, they can be composed before alarm presentation step. The goal is to produce fewer alarms with more expressive power. We distinguish four types of alarms depending on the place where they are activated (the command or the alarm system) and the way they defined from fault events. A **non-synthetic alarm** is triggered off by the command on the occurrence of a fault event. At the opposite, **synthetic alarm**s are triggered off by the alarm system from a set of characteristic example. For example, Causal Temporal Signature [3] is a useful formalism to combine fault events to define synthetic alarms. **Synthesis alarms** are defined in the alarm system. They summarise to the operator the information given by several detail alarms. **A detail alarm** can belong to all the alarm type introduced in this section. A basic idea of our approach of alarm filtering is to use a type of knowledge model (the functional graph) to define synthesis alarms. Using causal inhibition principle (see section 4.4), detail alarms are inhibited by synthesis alarms to reduce the number of alarm presented to the operator. In next section we will use the expression "raw alarm" for synthetic or non-synthetic alarms.

4 Alarm filtering based on the Functional Graph and the alarm status

In this section we introduce the different states of an alarm that enables us to filter with regard to fault detection and isolation process and the action of an operator. We use StateCharts formalism [6] to specify these states. This formalism is useful to represent the hierarchy of the concepts and the inhibition property [5] that links these concepts.

4.1 Brief recall on Functional Graph

A Functional Graph (FG hereafter) is a graphical tool that enables us to filter raw alarms of an APS [1, 2]. An FG models the relations between the main functions of a plant and the internal functions delivered by plant items. At each node we associate a function of the system. Directed arrows that model functional dependency relationships interconnect these nodes. The dependency relationships are of different types: composition relationships, flow relationships or adaptation relationships [2]. In the functional hierarchy given by an FG, some functions are distinguished: Principal Functions that corresponds to the system services and Initial Functions (Figure 1). Initial functions are implemented by elementary plant items. Consequently they are the initial causes of every faults of the system.

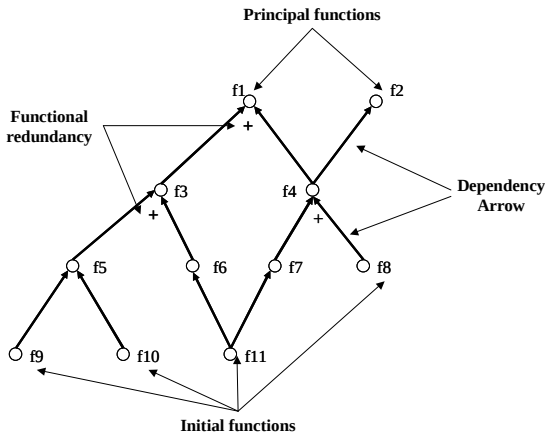

Figure 1: An example of Functional Graph

Graphically, the model is also completed by the addition of "or" operators (noted by a "+") that enable the designer to model the cases of functional redundancy. A functional redundancy is the fact that higher-level function (for example f3 in Figure 1) can be implemented differently by several lower level functions (for example f5 and f6 in Figure 1). It expresses the active or passive flexibility of the system. This flexibility characterises the different working modes and will be used when a failure will occur. In the other cases, the functional dependence relationships are constrained by an "and" operator. This operator is implicit in our models. On this example, since alarms at f10 or f11 nodes imply an alarm at f3 level, we can say that an alarm at f3 is a synthesis alarm. The two others are detail alarms.

4.2 Activation concept

Basically an alarm can be in two states: activated or not activated. The default state of an alarm is not activated because most of the time a plant works normally without faults. An alarm is activated when the plant behaviour is abnormal and verifies some conditions. The activation is a necessary condition to present the alarm to the operators. According to the fault detection (C1) and isolation specification of the plant, different cases are distinguished to allow an alarm to come back to the inactivation stated. Generally, it is sufficient that the starting condition becomes false. This is notably used when the starting condition depends on the monitoring of continuous variables. For example, when the variable value exceeds a threshold the alarm can be activated. As soon as the value goes down again under de threshold, the alarm can be deactivated even if the operator has not yet treated it. This scheme is more difficult to apply in discrete event system where many faults are characterised by rising edge of instantaneous events. In this case, the alarm can be deactivated only if the fault has been diagnosed (C2).

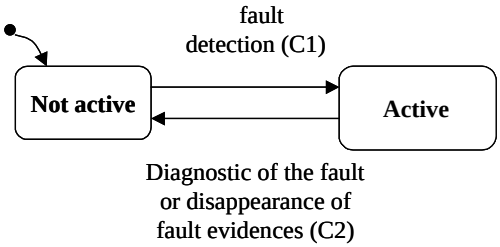

Figure 2: Activation states and their transition conditions.

4.3 Validation concept

When an alarm is activated, it must not be necessary presented to the operators. Its presentation depends on its validation status. By default, an alarm is not validated. It state switches automatically to validate if the actual working mode of the plant requires the use of the associate function to assume the plant production (transition condition C3 in Figure 3).

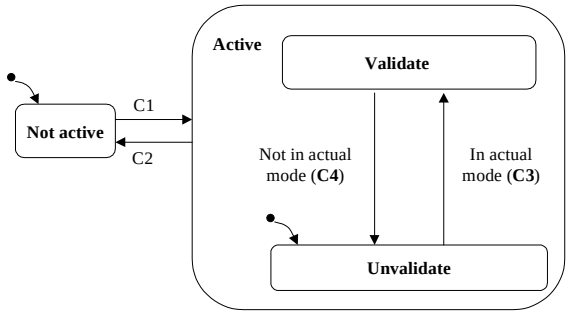

Figure 3: Validation states and their transition conditions.

Let us assume that the choice of a mode allows us to retain in exploitation a specific configuration of an FG. It means that the selected mode enables us to determine how a higher function is implemented by its redundant sub-functions. For example, let us consider the case of the process modelled in Figure 1. The actual mode selects function f3 to implement f1, f6 to implement f3, and f7 to implement f4. In Figure 4, the parts of the graph in grey colour represent the parts of the system that are not exploited by the current mode. If a raw alarm is triggered off concerning function f9, it must not be validated because this function is not in production.

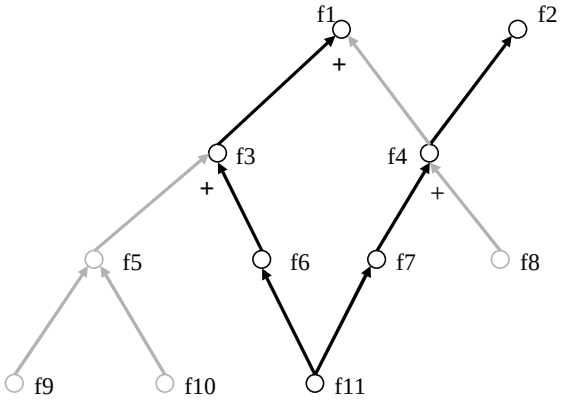

Figure 4: Projection of a Functional Graph by using a specific working mode.

The use of the validation concept is very useful to suppress false alarms such as the alarms produced by maintenance operations on the process. In addition, this concept is very powerful for filtering because it defines filtering rules in a positive way. Indeed, in some IAPS (Intelligent Alarm Processing System), the authors try to define forbidden states for an alarm or a group of alarms. Since the forbidden states are generally due to failures and since it is difficult to identify all the states that can be reached because of the combination of multiple failures, it is easier to know the normal behaviour of a system and to define alarm presentation with regard to this state.

4.4 Causal inhibition concept

To reduce the number of alarms that are presented to the operators we also propose to exploit causal inhibition concept. This concept exploits the causal relations between the functions of the different levels of an FG. These causal relations are illustrated by the existence of paths that link the initial functions to the principal functions of an FG. On a causal path, if we observe the behaviour of the system at different levels, in case of fault occurrence, raw alarms can be triggered off by all of these observable functions. The presentation of all these alarms is not necessary because they all inform about a common and unique fault (the failure of the initial function that initiates the causal path) and about the same consequences. Causal inhibition consists in masking all the alarms belonging to the same causal path, excepted one of them. The remaining alarm depends on the status of the operator. Piloting operators need to know which flexibility remains in the process to continue the production. For them, the reconfiguration process consists in finding the lowest alternative (“OR” node) from the initial function where it is possible to commute. In this case, causal inhibition consists in

masking a detail alarm (lower-level alarm) by a synthesis alarm (higher-level alarm). For maintenance operator, it is the opposite because these operators need to diagnose the root cause of plant misbehaviour in order to identify which plant item must be repaired or replaced.

Consequently a validated alarm can be in two states (Figure 5): non-inhibited or inhibited. Non-inhibited is the default state because initially, the alarm must be treated, as it was alone. If an inhibiting alarm exists or occurs after (transition condition C5), immediately its state becomes inhibited and the alarm is no more presented. The transition from the inhibited state to the non-inhibited state depends on the absence of all of its inhibiting alarms (transition condition C5). An alarm can be inhibited only by validated alarms.

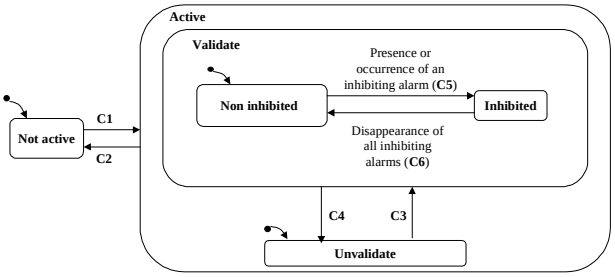

Figure 5: Inhibition states and their transition conditions.

4.5 Suppression concept

When an alarm is presented to an operator, sometimes he wants to suppress it (C7 condition in Figure 6) in order to distinguish the most important alarms that are displayed on his console. Such suppression does not necessary means that the alarm has been treated. Since the operator can have forgotten this suppression, it must not be definitive. It is the reason why two states are necessary to refine the status of a presented alarm: presented or not presented. When an alarm is not presented this means that after a delay, it must be displayed again on the console of the operator (C8) (Figure 6).

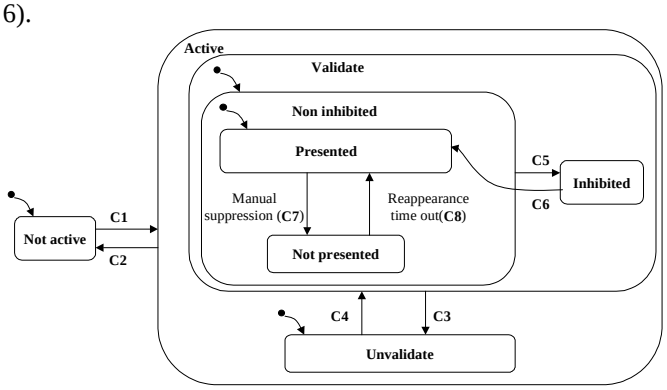

Figure 6: Presentation states and their transition conditions.

4.6 Acquitement concept

In order to focus the attention of the operators, new alarms flash on the console's screen. However, this feature can disturb them when the screen is overloaded by many flashing alarms. To prevent this situation, the operator can acquit some alarms without treating immediately their causes (transition condition C9 in Figure 7). This action stops their flashing and eventually ranges these alarms in a list of acquitted alarms. But these alarms remain visible contrary to suppressed alarms. This status reduces the risk for an operator to forget to treat an alarm. Two states implement acquitement concept: flashing (the default state) and acquitted. When the operator select the acquit action, the alarm status becomes acquitted. An acquitted alarm comes back in flashing status after a temporisation time out or when the alarm is represented to the operator because a new detection of the same fault (transition condition C10 in Figure 7).

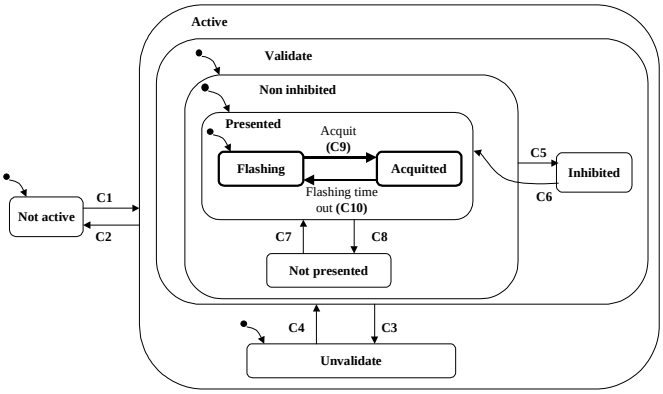

Figure 7: Acquitement states and their transition condition.

4.7 Conclusion

Acquitement or suppression actions of an alarm depend on the operator choice. This choice must be based on the gravity and the delay to treat an alarm. These information must be supplied to the operator. This problem is out of the scope of this paper. An operator can also decide to suppress an alarm after its acquitment. In this case the presented/acquitted status of the alarm is inhibited, and the alarm becomes not presented (see Figure 7).

5 A prolog prototype for alarm filtering

In this section we will describe an inference engine for alarm filtering based on the Functional Graph and the concepts evoked in the sections 3 and 4. The inference engine is developed in prolog language. Its purpose is to filter alarms for a piloting operator by taking into account the model of alarm states (Figure 7) and the notion of synthesis alarms. Some features of the inference engine such as the search for recovery actions are explained in [1] and are not developed in this paper.

5.1 The prototype functioning

Figure 8 shows the architecture of the developed prototype. In fact, it is based on two principal parts: a base of predicates and a base of rules.

At first, some predicates describe the structure, the properties and currently used configuration of the FG. The last point is necessary to decide whether an alarm is validated or not. "**arrow(Fi, Fj)**" means that there is a dependency relation from Fj to Fi.

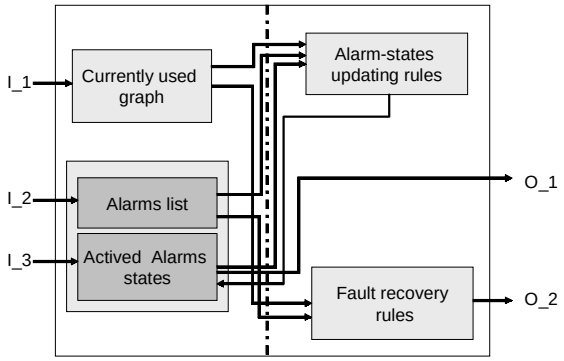

Figure 8: The prolog prototype architecture

"**node_OR (Fi)**" predicate qualifies "OR" node. "observable(Fi)" means that the behaviour of the system (faulty or not) can be established at this node level. Consequently, raw alarms are only triggered off at observable nodes. "**arrow_c(Fi,Fj)**" enables a designer to distinguish the part of an FG that is not validated by the actual working mode. This predicates means that there is a potential dependency relation from Fj to Fi.

Other predicates such as "**alarm(Fi)**" enable us to define the different states of an alarm. The last predicate means that an active alarm has occurred at the node Fi. The absence of this predicate to qualify a node means that the associate alarm is inactive. The other concepts introduced in section 4 are implemented by a similar scheme that defines each alarm state (validate, invalidate, inhibited, not_inhibited, presented, not_presented, flashing, acquitted) by a predicate in the form "**state_alarm(Fi)"**. Thus, if the predicate "**validated_alarm(Fi**)" exists this means that there is a validated alarm at the node Fi. When an alarm is activated, and the associated node belongs to the currently used graph, this adds the following predicates to the database: alarm(Fi), validated_alarm(Fi), not_inhibited_alarm(Fi), presented_alarm, flashing_alarm(Fi). These predicates enable the operator to have different sub-lists to consult the alarms of the plant (output O_1 in Figure 8). For example, the operator can know all the alarms that are currently inhibited or invalidated, by simply entering questions such as "inhibited_alarm(X)" or "not_validated_alarm (X)" (Figure 9). In the previous questions, "X" is a variable that is successively assigned to each node that verifies the predicate.

The base of rules is structured in three packages of rules. The first package enables the operator to interact with the alarm system. It contains three categories of rules (see Figure 8). The category I_1 (exploit(Fi,Fj) and not_exploit(Fi,Fj)) enables him to select the configuration of currently used functional graph. In the second category (I_2), "**appears_alarm(Fi)**" is a rule that enables the operator to activate an alarm at an observable node level. To deactivate an alarm the operator has "**disappears_alarm(Fi)**" rule. The rule "**alarm_list([ ])**" gives all the alarm that are actives at a moment. " The third category (inputs I_3) are used to control the presentation of active alarms: "**clear(Fi)**" rule and "**acquitt(Fi)**" rules apply respectively the suppression concept and the acquitement concept.

```
?- appears_alarm(f11).
Yes
?- alarms_list(Liste ).
Liste = [alarm(f11)] ;
No
?- alarm(X ).
X = f11 ;
No
?- validated_alarm(X ).
X = f11 ;
No
?- not_inhibited_alarm(X ).
X = f11 ;
No
?- not_validated_alarm(X ).
No
?- not_inhibited_alarm(X ).
X = f11 ;
No
?- inhibited_alarm(X ).
No
?- supressed_alarm(X ).
No
?-
```

Figure 9: Definition of sub-lists of alarms.

The second package of rules implements alarm filtering techniques according to validation and causal inhibition concept. They are called each time the structure of the graph changes, or when a new alarm is activated. As an example, let us consider the activation of a new alarm at f11 node according to the active FG of Figure 4. This alarm is flashing because it is validated and not inhibited and then displayed on the alarm system screen (Figure 10-a). Let us now suppose the occurrence of another alarm at f3 node. The applying of causal inhibition concept implies the inhibition of the previous alarm, which is no more displayed (Figure 10-b). When a new alarm occurs at f10, it is not presented because it status is invalidated by the fact this node does not belong to the actual configuration of the plant (Figure 10-c).

```
?- appears_alarm(f11).
not inhibited alarm in f11

Yes
?- flashing_alarm(X).

X=f11;

No
?-
```

```
?- appears_alarm(f3).
not inhibited alarm in f3

Yes
?- flashing_alarm(X).

X=f3;
No
?- inhibited_alarm(X).

X=f11;

No
?-
```

```
?- appears_alarm(f10).

Yes
?- flashing_alarm(X).

X=f3;

No
?- not_validated_alarm(X).

X=f10;

No
?-
```

-a **-b** **-c**

Figure 10: Illustration of causal inhibition and validation concepts.

The third package of rule assists the operator to determine the reconfiguration actions (output O_2) to perform in to handle the actual faults of the plant. In this case, reconfiguration decisions are posting on another screen of the alarm system (Figure 11).

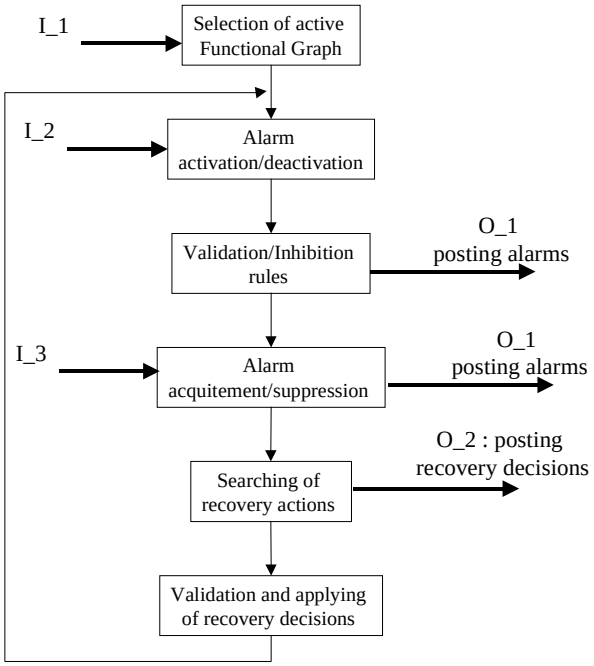

Figure 11: General functioning of the prototype.

6 Conclusion

In this paper we have presented a prolog prototype that implements different techniques for alarm filtering. These techniques consist notably in reducing the number of presented alarms by combining the techniques of synthesis alarms and causal inhibition. Moreover, by using validation technique, we are able to filter all the alarms that are not relevant with the actual working mode of the plant. All these techniques are applied through the functional graph of the plant. We have improved the filtering mechanisms by defining different alarm states that take into account the interaction of the operator with the alarm system. This inference engine is a generic one since the alarm filtering techniques we propose are independent of how raw alarms are triggered off. It can either be used with raw alarms based on residual techniques [4] or artificial intelligence techniques such as Causal Temporal Signatures [3].

The prototype presented is entirely off-line and obliges the operator to enter manually the alarms or their disappearance. But, we are actually developing an on-line prototype.

In future, we want to take into account in our alarm system a more accurate modelling of the plant-working mode. The idea consists to associate with each node of the functional graph a behavioural model [2, 8]. The goal is to refine reconfiguration procedures that are proposed to the operator in case of fault.

7 References

[1] A. Ghariani, A.K.A. Toguyéni and E. Craye. “A Functional Graph Approach for Alarm Filtering and Fault Recovery for Automated Production Systems”. In: the proceeding of WODES’2002, Manuel Silva (Ed.), Saragosse (Spain).

[2] A.K.A. Toguyéni and E. Craye (2000). Alarm Processing and Faults Recovery Based On Functional Modeling. In: the cdrom of WAC'2000, Seventh International Symposium on Manufacturing with Applications (ISOMA), Maui (Hawaii).

[3] A.K.A. Toguyéni, E. Craye and J.C. Gentina. “Time and reasoning for on-line diagnosis of failures in Flexible Manufacturing Systems”. In the proceedings of 15th IMACS World Congress on "Scientific Computation, Modelling and Applied Mathematics", Berlin (Germany), Vol.6, pp709-714, August 1997.

[4] B. Ould Bouamama, A.L. Gehin and M. Staroswiecki. “Alarm Filtering by Plant item Modelling and Bond Graph approach”, 4th IFAC Workshop on On-line Fault Detection and Supervision in the Chemical Process Industries (CHEMFAS-4), Seoul, Korea, June 8-9, 2001.

[5] C. André and J.P. Rigault. “Variations on the semantics of graphical models for reactive systems”. In the cdrom proceeding of IEEE System Man and Cybernetics conference, Hammamet, Tunisia, October 2002.

[6] C. Harel. “StateCharts : A visual formalism for complex systems”. In Science of Computer Programming, vol. 8, pp. 231-274, 1987.

[7] D.S. Kirschen and B.F. Wollenberg. “Intelligent Alarm Processing in Power Systems”. In the proceedings of the IEEE, Vol. 80, N°5, pp. 663-672, 1992.

[8] N. Dangoumau, S. El Khattabi and E. Craye. “Modes Management for Automated Production Systems”. In the cdrom of WAC'2000, Seventh International Symposium on Manufacturing with Applications (ISOMA), Maui (Hawaii), 2000.

[9] N.N. Naito and S. Ohtsuka. “Intelligent Alarm-Processing system for Nuclear Power Plants”. Nuclear Technology, Vol. 109, pp. 255-264,1995.