A Dynamic, Real-Time Alarm System for Power Plants

ILSE LEAL–AULENBACHER
Instituto de Investigaciones Eléctricas
Gestión Integral de Procesos
Reforma 113, 62490 Cuernavaca
MEXICO
ilse.leal@iie.org.mx

JOSÉ M. SUÁREZ–JURADO
Instituto de Investigaciones Eléctricas
Gestión Integral de Procesos
Reforma 113, 62490 Cuernavaca
MEXICO
jmsuarez@iie.org.mx

EFREN R. CORONEL–FLORES
Instituto de Investigaciones Eléctricas
Gestión Integral de Procesos
Reforma 113, 62490 Cuernavaca
MEXICO
coronel@iie.org.mx

Abstract: Alarm systems have been widely used to help power plant operators identify abnormal conditions that require their attention. However, there are certain well–known issues with alarm systems that have a negative effect on their ability to provide useful information to operators. Alarms can become difficult to manage when too many of them are presented simultaneously. Power plants often perform maintenance tasks or change configurations periodically. When alarm limits are configured according to ideal operation conditions alone, alarms may become irrelevant when operation conditions change. In this paper, we present a real–time alarm system, which incorporates tactics such as hierarchical organization of alarms and alarm processing, relative to current power plant operation conditions. We show how these strategies can be used to improve how alarms are presented to power plant operators.

Key–Words: alarm system, real–time, data acquisition system

1 Introduction

Although alarm systems have been used for a long time, there are some problems that make managing alarms difficult and can even become a burden to operators. Alarms are defined as "an audible and/or visible means of indicating to the operator an equipment malfunction, process deviation, or abnormal condition requiring a response" [1].

A recurrent issue with alarm systems is the number of alarms presented to operators. Alarms systems should provide operators with relevant and timely information to help them identify abnormal operation conditions. In practice, however, alarms displays can overflow with irrelevant or redundant alarms.

In this paper, we present a dynamic, real–time alarm system that implements several strategies that help organize alarms in relation to structure and division of responsibilities in power plants. Instead of configuring a set of alarm limits based on normal operation conditions, our alarm system supports the definition of different sets of alarm limits, which adapt to different operation conditions. Our system can be configured to detect current operation mode in a power plant automatically and adjust alarm limits accordingly. In this way, we seek to ensure that alarm limits are verified in relation to current operation conditions.

In addition, our system categorizes alarms in alarm subsystems, which can be modeled after the structure and organization of power plants. Therefore, alarms are presented to operators based on their roles and responsibilities. We believe that this mechanism helps overcome the problem of overwhelming operators with too many alarms. Our system was implemented in C++ under Linux and it is in its initial testing phase.

The remainder of this paper is structured as follows. In section 2, we describe issues with alarm systems in general. We present our system design principles and architectural drivers in section 3. Section 4 describes our development practices, which are the result of years of accumulated experience with real–time, mission–critical systems. In section 5, we describe our main alarm system features and describe how they help present alarms more effectively through the use of alarm subsystems and hierarchical organization. Section 6, gives a high–level overview of our system architecture. Finally, we present our conclusions and future work.

2 Issues in Alarm Systems

An alarm system must help operators identify conditions that require their attention. Information must be presented in a clear and timely manner. When too much information is presented, it is difficult to use it to make decisions. Moreover, if alarms being displayed are not relevant or useful, operators might end up ignoring them altogether. We believe it is important to keep researching these issues, because most of the literature that talks about them is not very recent. Although core issues in alarm systems have been identified for a long time, they still affect alarm systems today.

In this section, we discuss these issues to put our design objectives and system features in context.

2.1 Number of alarms presented to a user

One common problem with alarm systems is that too many alarms may be presented to an operator during mode changes or plant transients [2]. Operators need to analyze each alarm to determine what needs to be done. Therefore, operators may be able to properly handle only a limited number of alarms at a given time. If too many alarms are presented at the same time, the process of handling alarms may prove time–consuming. In fact, research suggests that one alarm in ten minutes (about 150 alarms per day) can be considered acceptable. Even up to two alarms in ten minutes can be manageable. However, anything higher than that, can hinder an operator's capacity to properly manage alarms [3] [4]. It has been shown that frequent alarms can become a distraction instead of helping operators, who might even end up disabling alarm systems [5].

Our system aims to overcome this issue by categorizing alarms in subsystems. Since power plants are complex, usually their management is naturally divided in subsystems. Thus, certain operators focus on specific areas such as electrical systems, turbines, reactors, etc. Our alarm system takes advantage of this fact and categorizes alarms in subsystems. With this mechanism, operators can focus on alarms that are related to their area of responsibility.

Although categorizing is a good starting point to reduce the number of alarms presented to operators, it is also necessary to consider that power plant management is hierarchical in nature. For example, control room operators need to be able to monitor the entire power plant, with all of its subsystems. Hence, our system organizes subsystems in a configurable, hierarchical structure. This feature will be described in more detail in section 5.1.

2.2 Alarms not handled on time

Another important factor in alarm systems is related to what happens to alarms when they are not acknowledged for a long time.

Alarms can remain unacknowledged for several reasons: In occasions, certain alarms that were not configured correctly can be systematically ignored over time (they become *nuisance* alarms). In this case, it is important to identify such alarms and either eliminate them or configure them correctly. Other scenario could involve human error. Perhaps an operator got sidetracked while dealing with other alarms, or maybe an alarm was raised and an operator ignored it thinking it was going to be handled by a peer. These situations should be brought to the attention of a supervisor. In this way, these situations can be promptly identified and solved. Our system addresses these issues through a timeout mechanism. This mechanism is directly related to our hierarchical organization of alarm subsystems (see section 5.1). Each alarm subsystem can have a timeout assigned to it. Therefore, if an alarm is not acknowledged after a certain period of time, it is moved to the alarm subsystem immediately above it in the hierarchy. This mechanism is described in more detail in section 5.2.

2.3 Alarms becoming irrelevant

Another issue with alarm systems is that certain alarms may become irrelevant over time. This happens since power plant operation is dynamic in nature. Power plants may add or remove equipment, undergo maintenance, change configuration, etc. These changes happen over time and have a direct impact on how alarm limits should be configured.

In many cases, alarm limits are configured with assumptions about ideal operation conditions. However, operation conditions can change. For example, nuclear power plants refuel every 18–24 months [6]. This constitutes a different operation mode in which the reactor is under maintenance (not in operation). If an alarm system does not take this into account, reactor–related alarms might be triggered. However, in that particular situation (a power plant operating in "refuel mode"), those alarms would not be relevant. Our system addresses this issue by dynamically adjusting to the current power plant operation mode. This is achieved through the use of *Alarm Plans*, which are described in section 5.3.

3 Architectural drivers

To design and implement our alarm system, we started by identifying our main architectural drivers. Software architecture is defined by certain functional, quality and business requirements. These requirements are known as *architectural drivers*[7]. Conflicts among system requirements are common. For example, a system could require a high level of security *and* a high level of modifiability. However, both requirements will most likely impact each other. For instance, a highly configurable system might have more security vulnerabilities. Architectural drivers help software architects and designers manage conflicting requirements from the start by clearly identifying quality attributes that will impact the whole system architecture.

Since our system will be integrated to a critical data acquisition system (known as NDAS), it is important to consider our alarm system as a critical system as well. Critical systems are systems whose failure can have negative economic, physical or safety consequences on organizations or people. Reliability and Performance are among the most important architectural drivers for these types of systems[8].

To put our architectural drivers in context, we will give a brief overview of the NDAS in the next section.

3.1 Data Acquisition System

The NDAS was originally developed in order to gradually replace a data acquisition system that ran on a VAX/VMS platform. It is capable of acquiring thousands of signals that power plant operators and engineers use to monitor the state of a nuclear power plant and its different processes[9].

Data can be acquired on real time or in historical mode. Real–time acquisition involves a 1Hz sampling rate. In contrast, Historical mode acquisition involves sampling rates that can range from 1 to 250Hz[10] [11].

This system is critical because it is used 24/7 by engineers and power plant operators to assess the state of the plant and to aid in decision–making.

Hence, our alarm system must be able to monitor and process thousands of data points in real time. In addition, we consider that it is important for this system to be flexible enough to support changes in plant configuration or structure. Therefore, our main architectural drivers are reliability, performance and modifiability.

3.2 Reliability

Reliability is our most important architectural driver. As we mentioned before, our alarm system will be integrated to a DAS system that operates 24/7 and which is a mission–critical system. For this reason, our system was designed in such a way that it does not interfere with the NDAS in any way. Hence, our alarm system must function as a separate module with respect to the NDAS.

To achieve this, our alarm system is self–contained to minimize any dependence with the NDAS. In this way, our alarm system obtains its inputs (thousands of data points in real–time) from the NSAD in a controlled manner, by using specific routines that access acquired data in read–only mode. Therefore, although our alarm system will be an integral part of the NDAS, we took care to design our system to minimize coupling. Our alarm system code is independent from the NDAS core processes. This also aids to achieve modifiability, as discussed in section 3.4.

3.3 Performance

Performance is another very important architectural driver in our alarm system. Our alarm system must be capable of analyzing thousands of data points in real–time. Performance considerations in our system are closely related to reliability concerns. In order to issue alarms correctly, our system must be capable of accessing the information it needs in the most efficient way possible. As we mentioned before, our system is implemented in C++ and Linux. This allowed us to take advantage of inter–process communication (IPC) mechanisms and other strategies that we have developed over the years in our experience designing and implementing real–time data acquisition systems. We describe such strategies below.

3.3.1 Use of shared memory areas

Shared memory is the fastest form of inter-process communication (IPC) available [12]. Our system databases and data are stored in shared memory areas to maximize performance. This is a tactic that we have been using since we were developing the NDAS [10]. Therefore, our system reads data points in real–time from a read–only shared memory area in the NDAS.

3.3.2 Database access mechanism

Databases and configuration files in both our alarm system and the NDAS are simple text files that are copied to shared memory areas. We decided not to use a database management system (DBMS) for our databases for several reasons: to avoid dependence on commercial products (this issue will be covered in more detail in section 3.4), to avoid unnecessary overhead and to be able to easily copy databases to shared memory areas.

Although our databases have thousands of records, our real–time shared–memory mechanism provides excellent access times (in the order of milliseconds).

3.4 Modifiability

Modifiability is a key architectural driver in our design. Although our alarm system was developed with the NDAS in mind (see section 3.1), we want to be able to deploy our alarm system in other power plants.

In addition, one of the most important features in our system is that it adapts to current operation conditions. To achieve that, our system must be highly configurable. For that reason, our alarm system can be configured to adapt to different operation scenarios (see section 5.3).

Our configuration files are plain text files that are easy to edit and manage. End users can configure several aspects of the system, such as alarm categories (subsystems) and their hierarchical structure.

Another important consideration to achieve modifiability was to avoid dependence upon commercial software. In this way, we maintain control over our system code, making it easy to adapt or modify as necessary. This is very important because our systems must be guaranteed and supported for a minimum of ten years.

4 Development practices

Since our software will be part of a mission–critical system, we need to take special considerations with regards to our development practices. We need our code to execute as fast as possible for very long periods of time. To achieve this, we propose the following guidelines:

- *Avoid instantiating complex objects in iterative code.* We create and initialize objects only once. We seek to reuse objects instead of creating them every time.

- *Avoid using dynamic memory.* We prefer to use static memory in order to improve reliability in our systems. Although nowadays garbage collectors and memory managers perform quite well in managing dynamic memory, it is important for us to maintain total control over how and when memory is used.

- *Avoid using third–party software or libraries.* Integrating third–party software can help speed up development time and is in general a good practice. There are third–party libraries that are open–source and known to be reliable. However, in our case, it is more important to maintain control over our software and abide by our current practices (such as not using dynamic memory, which most libraries do).

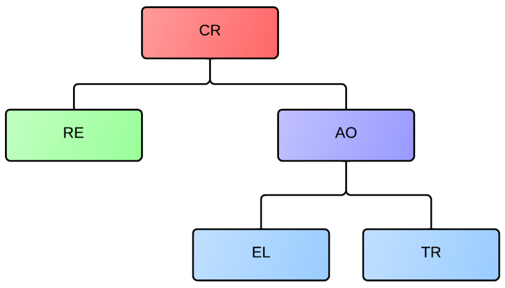

Figure 1: An example alarm subsystem hierarchy

5 Main Alarm System Features

In this section, we present the features we implemented in our alarm system. These features tackle the issues described in section 2.

5.1 Alarm Subsystems

We base our alarm subsystems strategy on the premise that alarms are useful when presented to operators that actually find them relevant. Our approach is based on the idea of dividing alarms into *alarm subsystems*. An alarm subsystem is basically a category of alarms.

Power plants or any big system for that matter, can usually be divided into smaller parts or *subsystems*. For example, in a power plant we can have a turbine subsystem, a reactor subsystem, an electric subsystem, etc. Each subsystem is usually under the responsibility of an operator. Because subsystems are not isolated, their relationships can be represented through a hierarchical organization. This organization should be carefully modeled after the power plant structure.

Fig. 1 shows an hypothetical and very simplified example of how a hierarchical subsystem arrangement could look like in a nuclear power plant. In this example, on the lower level there are two subsystems: electric (*EL*) and turbine (*TR*). These subsystems, are overseen by an auxiliary operation (*AO*) subsystem. At the same time, the auxiliary operation subsystem is supervised by the main control room subsystem (*CR*), which also oversees the reactor (*RE*) subsystem. We will use this example in the next sections to illustrate how our alarm subsystems work.

In our implementation, when alarms are configured, they are assigned to a specific alarm subsystem. Ideally, each alarm subsystem should be assigned to the operator responsible for it in the power plant.

When operators log into our system, they can manage alarms that are assigned to them. When we refer to managing alarms, we mean the action an operator takes to acknowledge or reset an alarm. For instance, if the user responsible for the auxiliary operation (*AO*) subsystem logs in, he can view and manage alarms that belong to the auxiliary operation subsystem itself. This is a good first step, but because alarm subsystems are arranged hierarchically, there are certain important considerations. If the system is too restrictive and limits users to their own subsystems, dividing alarms may become impractical. For that reason, we consider that hierarchy should play an important role on how alarms are managed and presented.

Looking again at our example in Fig. 1, if an *OA* operator logs in, shouldn't he be able to supervise *EL* or *TR* alarms? At the same time, should an *EL* operator be able to see *OA* alarms? To solve this issue, we decided to take a flexible approach and thus, based our alarm management system on the following principles:

- *Operators can view and manage alarms that belong to their own subsystem.* For example, an *EL* operator would be able to manage *EL* alarms.

- *Operators can view alarms from a subsystem in a lower hierarchy level.* Operators should be able to supervise and take actions on alarms directly under them in the subsystem hierarchy. For instance, if an *OA* operator logs in, he can view and manage both *EL* and *TR* alarms.

- *Operators can view alarms from a subsystem at a higher hierarchy level.* It is easy to see that operators should not be able to manage alarms from a higher hierarchy level, since such alarms are not under their responsibility. However, operators may occasionally need to see alarms in a higher hierarchy level. With this in mind, our system allows operators to view alarms from the subsystem directly above them in the hierarchy; yet, it does not allow them to manage such alarms.

With this approach, not all alarms are presented to every operator. Rather, alarms are categorized so that certain alarms are presented to operators who will find them relevant. By dividing alarms into subsystems, we can help address the issue of presenting too many alarms to operators.

5.2 Timeout Mechanism

Another important key principle in our design was to promote best practices regarding division of responsibility. One way to do so is by implementing mechanisms that ensure that alarms are actually managed. For instance, there may be a case where operators notice an alarm and postpone handling it because they are busy monitoring other events or are not sure what to do about it at that time. After a while, operators may end up forgetting such alarm. In such cases, it may be useful to bring that alarm to the attention of the subsystem directly above it in the hierarchy (a supervisor).

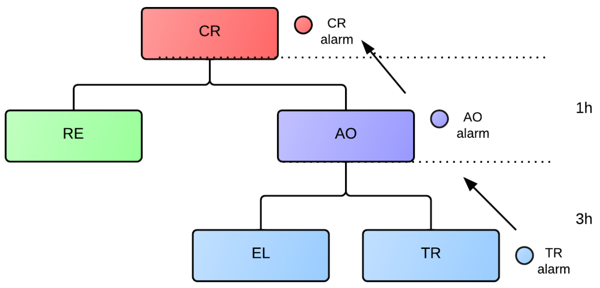

Figure 2: Example alarm timeout

To achieve this, we designed a mechanism where each subsystem has a *timeout*. This means that if an alarm is not acknowledged within a certain period of time, it will be assigned to the subsystem above it in the hierarchy.

Returning to the example hierarchy shown in Fig. 1, let us suppose that an alarm is raised in the *TR* subsystem. Let us also suppose that this subsystem has a timeout of three hours. If an *TR* alarm is not acknowledged within those three hours, our system would move the alarm to the *AO* subsystem. Therefore, if for some reason an *TR* operator was unable to manage the alarm, an operator one level higher in the hierarchy (in this case, *AO*) would be able to see it and manage it. Similarly, let us suppose the *AO* subsystem has a timeout of one hour. If the alarm is not acknowledged within one hour, it would be moved to the *CR* subsystem. This example is illustrated in Fig. 2.

Timeouts are configurable and assigned in a per–subsystem basis. This provides greater flexibility, because the system can adapt to the needs and policies of each power plant.

5.3 Alarm Plans

For an alarm system to be effective, alarm limits must be configured properly. Most data acquisition systems are designed to operate for extended periods of time. Hence, it is important to recognize that power plant operation conditions are likely to change at certain times. Therefore, we consider that it is important for alarm systems to take this into consideration.

In order to address this issue, we propose the concept of *dynamic alarm plans*. An alarm plan is a configuration file that specifies a group of data points with their corresponding alarm limits, among other parameters. In our system design, several alarm plans can be defined, one for each set of operation conditions. For example, we can define an alarm plan for normal operation conditions and another one for "under maintenance" conditions.

Our system supports any number of different alarm plans. Some plants might find it easier to define one alarm plan at first. Other plants might require to define several alarm plans in order to have better alarm management when certain systems are under maintenance.

These alarm plans can enter or go out of processing dynamically, either manually or automatically.

- *Manually.* An operator can manually specify which alarm plan will be used at a given time. Usually, such operation is limited to the operator assigned to the subsystem highest in the alarm subsystem hierarchy.

- *Automatically.* Alarm plans can be associated with an arbitrary data point, which we call "operation–mode data point". Such data point is assumed to have several discrete states, which should indicate the operation conditions of the plant. For instance, a data point value of zero could be associated to normal operation and a value of one to operation under maintenance. Our design assumes that the process of assigning the correct value to the operation–mode data point is a black box. This is to ensure that our system can be easily adapted to different data acquisition systems and power plants.

6 Architecture and Implementation

A high–level view of our system architecture is shown in Fig. 3. Our system is divided in six modules: Alarm Configuration, Alarm Plans, Operation Mode, Alarm Detection and Alarm Operations. As we can observe, our modules interact with each other through three shared memory areas: Alarm Configuration, Alarm Plans and Alarm List.

In the following sections, we describe how our shared memory areas and modules interact.

6.1 Shared memory areas

Our processes need real–time access to alarm plans and configuration files. Shared memory is the fastest form of inter-process communication (IPC) available [12]. Therefore, to achieve the level of performance we require, our processes communicate through shared memory areas.

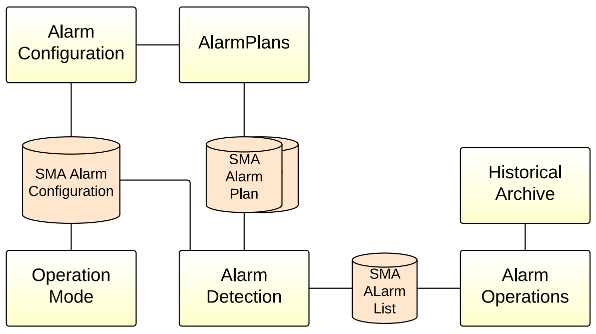

Figure 3: High–level system architecture view

Our databases are stored as text files on disk and then copied to shared memory areas. Since information in our system databases needs to be read by several processes without compromising performance, the use of shared memory areas is an optimal solution. In the following sections, we describe the content of our shared memory areas.

6.1.1 Alarm Plan Shared Memory Areas

During the initialization phase of our alarm system, alarm plans are loaded to memory. A shared memory segment is created for each alarm plan database. One of these alarm plans is accessed every second by our Alarm Detection Process. The alarm plan that is read each cycle, is determined by the current power plant operation mode, which is stored in the alarm list shared memory area (see figure 4).

Alarm plans contain a list of data points to be monitored in real–time to determine if an alarm must be triggered. The number of alarm plans that must be configured depends on how many operation modes were defined in the operation mode database.

6.1.2 Alarm Configuration Shared Memory Area

This shared memory area contains both the Operation Mode Database and the Subsystems Database. It is accessed by the Alarm Detection Process to obtain information about alarm subsystems and their configuration parameters.

The **Subsystems Database** contains the definition of a power plant's subsystems. The number of subsystems defined in this file depends on the organization and structure of a power plant. This database is crucial because it defines subsystems hierarchy, which impacts how alarms are processed by our system. It also contains configuration parameters for subsystems, such as timeouts.

The **Operation Mode Database** is where power plant operation modes can be configured. As we mentioned earlier, the number of operation modes that need to be configured can vary depending on different power plant requirements. Each operation mode is associated with an alarm plan.

6.1.3 Alarm List Shared Memory Area.

During the execution of our system, Alarm Configuration and Alarm Plans shared memory areas are read–only. However, the Alarm List shared memory area is updated every second by the Alarm Detection Process, which detects and triggers alarms (see Fig. 4).

When an alarm condition is detected, an alarm record is created. Each alarm record contains a timestamp, the name of the alarmed data point, and general information about the alarm. When an alarm operation is performed, the corresponding alarm record is updated to indicate whether an alarm has been acknowledged, reset or if its timeout has expired.

6.2 Main Modules

Our system is divided in well–defined modules, which are designed to minimize coupling. Each module is self–contained, making our system easy to test and maintain. The block diagram in Fig. 3 shows our main system modules. As we can observe, our modules interact with each other mainly through reading and writing data in shared memory areas. In the next sections, we describe our modules in more detail.

6.2.1 Alarm Configuration

This module validates alarm plans and databases and loads them into their corresponding shared memory areas. This module is also in charge of performing initializations and indexing databases for faster access.

One of the most important validations performed by this module, is verifying that the alarm subsystems hierarchical organization is valid. For example, we verify that alarm subsystems appear only once in the hierarchy or that they do not have more than one parent node.

6.2.2 Operation Mode

Determines the power plant operational mode. If a data point is associated with the operational mode (*operation–mode data point*, which was described in section 5.3), it constantly monitors its value. It also handles requests for manual operation mode changes. This module is very important because its functionality enables dynamic alarm processing. Depending on the current operation mode, this module switches

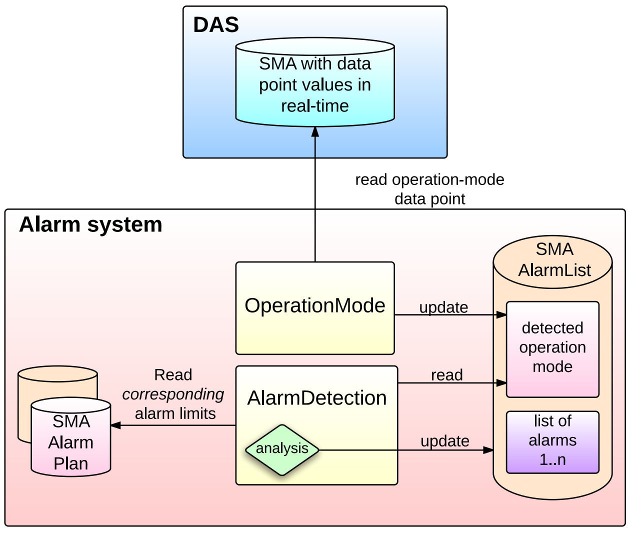

Figure 4: Alarm detection process

to the correct alarm plan, which becomes the *active alarm plan*.

6.2.3 Alarm Detection

This module implements our Main Alarm Detection Process. It is in charge of analyzing data points to determine whether to trigger alarms. It works in conjunction with the Alarm List shared memory, where it updates alarm information in real–time. Alarm Detection constantly reads the Alarm Plan shared memory, which contains alarm limits. As mentioned in section 6.2.2, Alarm Detection accesses the active alarm plan, based on the power plant operation mode detected by Operation Mode. Besides checking alarm limits, Alarm Detection analyzes other aspects such as:

Timeouts. For every alarm, it checks if its timeout has expired. Timeouts are determined by the time limit in seconds configured for each alarm subsystem (see section 5.2).

Deadband analysis. Deadbands are parameters that adjust alarm limits in analogic data points. They are used to prevent data points from entering or exiting a state of alarm because of small variations. When data points reach a value that exceeds an alarm limit, the surpassed limit is adjusted in opposite direction. For example, let us suppose that an analogic data point has a deadband of 0.5 and an alarm range of 5 to 10. If the data point has a value larger than 6, it would enter a state of alarm. Since the lower bound limit was exceeded, it would be adjusted to 5.0-0.5=4.5. In this way, for this data point to exit the alarm state, it would need to have a value lower than 4.5. Once an alarm has been reset, the affected limit will return to its original value. Deadbands are configured in a per–alarm basis.

Time filtering. Time filters are used to avoid triggering too many alarms in a short period. A time filter can be specified for each alarm and indicates the number of seconds that must elapse, before an alarm is triggered.

Grouping. Alarms are grouped in real–time as they are processed. Currently, we group alarms based on time periods. For example, we can choose to group alarms triggered in the last 10 minutes. This can be useful for root–cause analysis.

Chronological ordering. In memory, alarms are indexed in relation to the active alarm plan. Therefore, when a process needs to obtain information about a certain alarm, it can access it directly, without having to search. However, most graphic displays require showing alarms in chronological order. Therefore, when a new alarm is raised, it is inserted into a double linked list, which maintains alarms organized in chronological order.

6.2.4 Historical Archive

When alarms are reset, they are deleted from the alarm list since the condition that triggered them in the first place, no longer applies. However, in many cases, it is necessary to review historical data to identify certain patterns. For example, it can be useful to identify alarms that are triggered too often or at a particular time of the day.

Our system maintains a historical archive, which records the following events: triggered alarms, reset and acknowledged alarms, changes in power plant operation mode and alarms that exceed their timeout. This archive is circular, which means that once it reaches its maximum allowed size, it is overwritten from the beginning. This file is generated in CSV format, so that it can be easily analyzed in any spreadsheet program.

6.2.5 Alarm Operations

This module manages operations performed on alarms. For instance, when users acknowledge an alarm, this module validates the operation and accesses the Alarm List shared memory area to update the corresponding alarm status. This module receives and handles alarm operations by communicating with our system's graphic user interface (GUI).

7 Conclusion

In this paper, we described our implementation of a real–time alarm system that incorporates different strategies to present alarms in a more efficient way. Strategies such as dividing alarms in categories and establishing an alarm subsystem hierarchy can help tackle problems such as presenting operators with too much information.

One of the main advantages of our alarm system is its flexibility. Our system allows alarms to be configured and organized according to the structure of any power plant. However, configuring an alarm subsystem hierarchy can be complex. In the future, we intend to research ways to facilitate the process of alarm configuration.

Our architecture was designed so that our alarm system can be adapted to different power plant configurations.

We have measured the performance of our alarm system and we have obtained good results. We have tested our process CPU usage under different scenarios, by simulating events that trigger a large number of alarms. Even under a high workload (approximately 8000 alarms), our system uses less than 1% of CPU.

Regarding reliability, our alarm system has been integrated to a data acquisition system with optimal results. Our alarm subsystem has been running alongside a DAS for extended periods of time (more than one month) and both systems have remained stable.

It is important to mention that our system is in its initial phase of testing. We intend to perform further analysis once our system is operating with live data in a power plant.

8 Future work

We would like to evaluate how our alarm processing strategies impact how operators manage alarms by obtaining further feedback from power plant operators.

Acknowledgements: The authors would like to acknowledge (in alphabetical order) Brito–Fernández L. Michel, Del Toro–Cortés A. Laura and Télis–Pérez Carlos, who participated in the development of the alarm system presented in this paper.

References:

[1] "ANSI/ISA-18.2-2009 Management of Alarm Systems for the Process Industries," 2009.

[2] "Advanced control room alarm system: Requirements and implementation guidance," EPRI, Technical Report, 2005.

[3] B. H. Wayne Crawford, "Better management of plant alarms," *Energy-Tech Magazine*, May 2010.

[4] A. Hand, "How to build a better operator," *Control Design*, April 2012.

[5] M. Cvach, "Monitor alarm fatigue: an integrative review," *Biomedical Instrumentation & Technology*, vol. 46, no. 4, pp. 268–277, 2012.

[6] "Costs: Fuel, operation, waste disposal & life cycle," Nuclear Energy Institute, Technical Report, 2014. [Online]. Available: http://www.nei.org/Knowledge--Center/Nuclear--Statistics/Costs--Fuel,--Operation,--Waste--Disposal--Life--Cycle

[7] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1998.

[8] I. Sommerville and G. Kotonya, *Requirements Engineering: Processes and Techniques*. New York, NY, USA: John Wiley & Sons, Inc., 1998.

[9] R. Montellano-Garcia, I. Leal-Aulenbacher, and H. M. Bernal, "A gradual data acquisition replacement strategy for the laguna verde nuclear power plant," in *Proceedings of the 9th WSEAS International Conference on Data Networks, Communications, Computers*, ser. DNCOCO'10. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2010, pp. 41–46.

[10] I. L. Aulenbacher, J. M. S. Jurado, and E. R. C. Flores, "A real-time data acquisition system for the Laguna Verde nuclear power plant," *W. Trans. on Comp.*, vol. 9, no. 7, pp. 778–787, July 2010.

[11] I. L. Aulenbacher and J. M. S. Jurado, "A data acquisition system for the laguna verde nuclear power plant," in *Proceedings of the 8th WSEAS international conference on Data networks, communications, computers*, ser. DNCOCO'09. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 142–146.

[12] W. Stevens, *UNIX Network Programming: Interprocess Communications*, 2nd ed. Prentice Hall, 1999, vol. 2.