Towards Alarm Flood Reduction
Yannick Laumonier, Jean-Marc Faure, Jean-Jacques Lesage, Hervé Sabot

To cite this version:
Yannick Laumonier, Jean-Marc Faure, Jean-Jacques Lesage, Hervé Sabot. Towards Alarm Flood Reduction. 22nd IEEE International Conference on Emerging Technologies and Factory Automation, Sep 2017, Limassol, Cyprus. hal-01590704
Towards alarm flood reduction

Y. Laumonier, J.-M. Faure, member IEEE, J.-J. Lesage, member IEEE, and H. Sabot

Abstract—Alarm systems play critically important role for the safe and efficient operation of modern industrial plants. However, most existing industrial alarm systems suffer from poor performance due to too many alarms needing to be handled by operators in control rooms. This paper proposes a method to reduce the alarm flood by detecting redundant alarms so that they can be filtered later before being transmitted to operators. To do that, an approach based on pattern mining is selected. That method is then applied on an actual dataset coming from a General Electric power plant. The results show that removing redundant alarms allow significantly reducing alarm flood, without loss of efficiency nor safety.

Keywords—Alarm filtering; pattern mining; alarm flooding; industrial alarm systems; correlated alarms

I. INTRODUCTION

An alarm has been defined by ANSI/ISA 18.2 as “An audible and/or visible means of indicating to the operator an equipment malfunction, process deviation or abnormal condition requiring a timely response.” Alarm systems are critically important for safe and efficient operation of modern industrial plants such as manufacturing systems, oil refineries, petrochemical facilities, and power plants. The deviation of process variables from normal/optimal operating zones usually implies negative effects such as off-specification products as well as excessive consumption of raw materials and energy, and more generally leads to all kinds of faulty behaviors; alarms are therefore useful to increase a plant efficiency. However, the spread of numerical tools and of wireless sensors technologies makes much easier the creation of alarms and increases the number of alarms beyond which traditional alarms systems where designed for. As such, many existing industrial alarm systems are associated with poor performance, where the most observable phenomenon is that there are far too many alarms to be handled by industrial plant operators, referred to as alarm flooding.

Table 1 lists statistics of several basic performance metrics of alarm systems collected by [1] and compares them to the Engineering Equipment and Materials Users Association (EEMUA) recommendations. In this table, the “average alarms per day” refers to the total number of alarms occurring during the observation divided by the observation duration in days. Whereas the “peak alarms per 10 minutes” is obtained by determining the 10-minute period that contains the highest number of alarm occurrences during this period. Obviously, the statistics of performance metrics from various industries are much greater than the EEMUA recommendations.

As shown in Table 1, the alarm flood problem is shared by different industries but it is the power industry, the industrial context of this paper, which is the most impacted. Alarm flood is a general phenomenon that encompasses different phenomena such as false alarms, alarm redundancy or even chattering alarms. The huge discrepancies between the recommended values and the actual values found in industry highlights the pressing need to reduce the number of alarms.

A classical approach to reduce the number of alarms is called alarm rationalization. Alarm rationalization is defined by ANSI/ISA 18.2 as “The process of ensuring an alarm meets the requirements set forth in the alarm philosophy, including the tasks of prioritization, classification, settings determination, and documentation.” Alarm rationalization is discussed in [2] and generic guidelines to reduce the operator load and effectively presenting the alarm information to the operator are given. Complementary approaches, such as filtering alarms before they reach the operator, have also been explored. For example, the search of correlation between intrusion alarms is used in [3] to identify alarms which can be filtered. It appears that these techniques of correlation searching can be adapted to address the alarm flood problem due to redundant alarms in industrial systems. To do so, the different research problems inherent to alarm rationalization, such as the possible existence of wrongly configured alarms, must first be identified.

In [4], a survey on industrial alarm systems is proposed; the issues linked to the alarm flood are listed and the underlying research problems are outlined. It appears that one of the main difficulty in alarm management is to know whether the cause of the overflow comes from the alarm configuration step, the alarm design step or the alarm removal step. Consequently, the research problems to be solved are both to find new methods which avoid wrong configuration of alarms and to identify the root causes of the occurring alarms. A study of global causes for the existence of alarm flood, such as an...
abnormality propagating through a system, is also proposed in [4], but no technique to reduce the alarm flood is proposed. In [5], several rules to filter or suppress alarms for a nuclear power plant reactor by analyzing alarm datasets are described. This study shows that a complete analysis of an alarm log is a powerful way for discovering rules, like for example the cause-consequence rule which allows to limit abnormality propagation. However, completely analyzing alarm logs is not an easy task: alarm logs contain a wide alarm history and can possess a lot of hidden information, some of them being only revealed by searching correlations between alarms. Other approaches based on an exploration of alarms logs exist. One of them consists in finding temporal dependencies between alarm events in an alarm time series is presented in [6]. This work does not consider the reasons of those dependencies nevertheless.

The aim of this paper is to propose a method to detect correlations between alarms by using Discrete Event Systems (DES) modeling and pattern mining techniques. Once these correlations found, redundant alarms can be removed to reduce alarm flood, without loss of efficiency nor safety. After giving an overview of this contribution in Section 2, detection of frequent alarm patterns and alarms correlation are addressed in Section 3. Section 4 presents experimental results obtained with this method on real data issued from an industrial plant. Finally, concluding remarks and prospects are drawn up in the last section.

II. METHOD OVERVIEW

The main role of an alarm system is to warn, in real time, the operators about abnormal situations in the process. Another significant, while not so crucial, role is to build a database that contains alarm logs so that the monitored process and/or its control be improved during later re-design or revamping phases.

The final objective of this work is to introduce in the alarm system real time filters that remove on line redundant alarms therefore avoid that the operators are overwhelmed by alarm floods (Fig. 1 a). Construction of these filters require obviously to know which alarms are redundant and consequently can be removed.

The method proposed in the paper consists in automatically analyzing the alarm logs, by using pattern mining techniques, to detect frequent alarm patterns (step $\alpha$ in Fig. 1 b) then to examine these patterns to detect different kinds of correlation between alarms so as to state which alarms are redundant and why they are in this way (step $\beta$ in Fig. 1 b).

It is important to underline that construction of the filters is not addressed in this paper. Focus is only put on the offline part which corresponds to the frequent patterns detection and the alarm correlations statement (part b of Fig. 1).

Some assumptions will be made in the sequel of the paper:

- simultaneous alarm occurrences are not possible; the alarms occurrence set is totally ordered.
- the physical time is not considered.
- there is no knowledge of the system when detecting frequent alarm patterns.

III. DETECTION AND CHARACTERIZATION OF FREQUENT ALARM PATTERNS

A. Definitions

Each alarm is considered as an event whose occurrences are asynchronous, i.e. they are not synchronized by a clock. This approach allows to consider the history of alarm occurrences while satisfying the paradigms of Discrete Event Systems (DES).

Definition 1: Alphabet of alarms

The alphabet $\Sigma_A$ is the set of all alarms $a_i$ that are included in the alarm log.

$$\Sigma_A = \{a_1, \ldots, a_i, \ldots, a_{|\Sigma_A|}\} = \bigcup_{i=1}^{n} \{a_i\} \quad (1)$$
If the alarm log is represented by the example presented Fig. 2, the alphabet $\Sigma_A$ is $\{a_1, a_2, a_4, a_7, a_9, a_{11}, a_{13}\}$.

**Definition 2:** Alarm sequences

An alarm sequence is an ordered succession of alarms occurrences. $\Sigma_A$ is the set of all possible alarm sequences defined on $\Sigma_A$. An alarm log is a particular case of alarm sequence.

The length of an alarm sequence $S$ will be noted $|S|$. The length of the sequence $S = a_1a_2a_5a_{11}$ where 3 alarms are involved is equal to 4 ($|S| = 4$), for instance.

**Definition 3:** Patterns

In common language, pattern is a generic term that indicates a particular way in which something is done, is organized, or happens. In the DES field, this term refers to a sequence of event occurrences on which focus is put. Hence, in this paper, a pattern $P_k$ is a considered sequence of alarm occurrences contained in the alarm log. The set of alarms that can be found in the pattern $P_k$ will be noted $\Sigma_{P_k}$ (alphabet of the pattern) and its length, number of alarms occurrences that compose the pattern, $|P_k|$; the minimum length of a pattern is equal to two because a single event occurrence is not considered as a pattern.

**Definition 4:** Frequent patterns

A pattern $P_k$ is featured not only by its alphabet and length but also by another attribute: its support. The support value of a pattern $Supp_{P_k}$ is the number of times this pattern appears within the alarm log. A pattern $P_k$ is defined in [7] as a frequent pattern if $Supp_{P_k}$ is superior to a minimum threshold. In this work, the value of this threshold is equal to two as there is no prior knowledge of the system. Hence, a frequent pattern is formally defined by the following equations:

**Example:**

For the alarm log of Fig. 2, the pattern $P_1 = (a_7, a_{13}, a_1)$ with:
- $\Sigma_{P_k} = \{a_1, a_7, a_{13}\} \subseteq \Sigma_A = \{a_1, a_2, a_4, a_7, a_9, a_{11}, a_{13}\}$ ;
- $|P_k| = 3$ ;
- $Supp_{P_k} = 2$ ;

is a frequent pattern.

The other patterns $P_2$, $P_3$ and $P_4$ are defined on the same alphabet $\{a_1, a_2, a_9\} \subseteq \Sigma_A$, but the length of $P_2$ is different from the length of $P_3$ and $P_4$, and the order in which the alarms occur in $P_3$ and $P_4$ is different. Hence, none of these patterns is a frequent pattern.

Only frequent patterns will be considered in this work. Therefore, to alleviate the sequel of the paper, frequent patterns will simply be called patterns.

**B. Problem statement**

As mentioned at section II, the overall aim of this work is to find the redundant alarms that could be filtered. To meet this objective, two analyses will be sequentially performed:

- Automatic detection of the frequent patterns of the alarm log
- Expert analysis of these patterns in order to highlight the possible correlations between the alarms composing these patterns.

These two steps are described in what follows.

**C. Automatic detection of the frequent patterns**

Numerous data and pattern mining techniques are available to extract the frequent alarm patterns from an alarm log. A good survey of these techniques can be found in [8]. The well-known AprioriAll algorithm presented in [9] has been selected for this study because it scales well and therefore can be applied to industrial-sized alarm logs.

This algorithm requires first to define a sliding observation window, continuous subsequence of the alarm log. This window can be defined in different manners, as described in [10]. The three main solutions (Fig. 3) are:

- explicit segmentation, where the decomposition is based on the partial or global knowledge of the monitored system, for instance about its operation modes;
- time-based windowing, where the duration of the window is fixed;
- alarm-based windowing, where the length of the window, in term of number of alarm event occurrences, is fixed.

As, in this work, it is assumed that no knowledge of the system is available and the physical time is not considered, alarm-based windowing must be selected. The length of the window is determined by running the program with different length values then choosing the value that minimizes the computation time and maximizes the maximal support value (bi-criteria optimization).
Once the length of the alarm-based window selected, the second value to define is the shift of the sliding window. Two cases are presented at Fig. 4: shift equal to one alarm occurrence (Fig. 4 a) and shift equal to the window length (Fig. 4 b). The second solution has been selected in this study because it reduces the number of observation windows, and consequently the computation time, without missing any alarm occurrence.

Once the sliding window completely defined, automatic detection of the frequent patterns is performed by the Pattern detection algorithm. The Pattern detection algorithm is straightforward: the first line indicates the input of this algorithm while line 2 to 5 initialize the variables. Then, for the lines 6 to 13, the alarm log is run through to create the observation windows (lines 7 to 9) and to stock them as input for the AprioriAll algorithm (line 10). Once this step is achieved, the AprioriAll algorithm is run with the list of observation windows as input (line 14). Finally, line 15 is here to indicate that the Patterns detection algorithm returns a list of detected patterns as output.

**Example:**

When applying this algorithm to the example of the alarm log of Fig. 2, with a window length equal to 6, 5 observation windows $Obs_i$, with $i \in [1,5]$, are extracted from the alarm log, as represented in Fig. 5:

- $Obs_1 = a_1a_2a_3a_4a_5a_6a_7$;
- $Obs_2 = a_5a_6a_7a_1a_2a_3$;
- $Obs_3 = a_4a_1a_2a_3a_4a_5$;
- $Obs_4 = a_7a_4a_1a_2a_3a_4$;
- $Obs_5 = a_1a_2a_3a_4a_5a_6a_7$.

Processing of this set of sequences by the AprioriAll algorithm from [9] provides the following frequent patterns: $a_5a_1$, $a_1a_2a_3$, $a_1a_2a_3a_4a_5a_6a_7$, and $a_7a_1a_2a_3$. The support of each one of these patterns is equal to 2.

**Algorithm Patterns detection**

1. **Require**: Alarm Log L
2. windowLength=??
3. ListWindows=[]
4. start=0
5. Window=[]
6. while start+ windowLength < |L| do
7. for Alarm in range(start, start+ windowLength) do
8. Add Alarm to Window
9. end for
10. Add Window to ListWindows
11. Window=[]
12. start=start+ windowLength
13. end while
14. Patterns=AprioriAll(ListWindows)
15. return Patterns, List of detected patterns

**D. Alarm correlations**

The notion of correlation between alarms is defined in [11] as the description of a situation in which two or more entities have a reciprocal relation; while this reference focuses on alarms in telecommunication networks, this definition can be used for other industrial systems, however. An example of alarm correlation in power industry can be found in [12]; this work focuses on alarm correlation within couples of alarms, while our objective is to highlight correlations between sets of alarms that may include more than two alarms.

Several models (graphs, trees, causal maps, etc.) can be selected to find correlations; in our work, alarms are pinpointed by expert analysis of the set of the most frequent patterns (Fig. 1b). The role of the expert is not only to highlight the correlations but in addition to give an interpretation of every correlation.

A correlation between alarms may be indeed interpreted in different fashions. Two common interpretations are: cause-consequence relation (“the alarm $a_1$ is the cause of $a_2$”), therefore “$a_2$ is the consequence of $a_1$”) and equivalence relation (a set of alarms ($a_1, a_2, a_3$) has a common cause, hence is equivalent to a single alarm). In the case of this study, two other interpretations that focus on the origin of the relation
between the correlated alarms have been defined: physical and logical correlation. This distinction was made to highlight the significance of the numerical transition in the problem of alarm redundancy.

A physical correlation comes from the structure of the monitored plant. For example, in an industrial facility composed of connected tanks and pipes where a liquid or gas is flowing, an abnormal condition at a given point of the circuit is very likely to be propagated to the downstream devices. Another example of physical correlation is the correlation of alarms that are issued from a set of sensors whose detection zones are totally or partially common; an over- or under-threshold process value in this zone will give rise to physically correlated alarms.

A logical correlation comes from the setup of the alarms in the alarm management software. The number of alarms of this category has been strongly increased with the introduction of digital monitoring and alarm management systems, because it becomes much easier to create alarms with these systems, e.g. by assigning multiple alarms to the same system variables.

Examples of both kinds of correlated alarms that have been obtained from a real alarm log will be given in the next section.

IV. Experimental results

A. Alarm log

The results which are presented in this section have been obtained from an alarm log coming from an actual power plant and which has been provided by General Electric. This log contained initially chattering alarms that have been removed before analysis. The cleaned log contains around 1,300,000 alarm occurrences and represents 18 months of operation of the power plant. Therefore, the average number of alarm occurrences per day is equal to approximately 2,400 and the peak alarm occurrences/10 min is 460, which is far beyond the EEMUA recommendations.

B. Frequent patterns detection

This alarm log was processed by the pattern detection algorithm to find the frequent patterns that it contains. To reduce the computation time, this algorithm was applied on sequences of 100,000 successive alarm occurrences at a time.

Different values of the length of the observation window have been tested. These experiments led us to fix the value of this length to 50 because it provides the best compromise between the computation time and the maximal support value. With this value of the length of the observation window, the mean processing time of a sequence of 100,000 occurrences is equal to 3 hours, which shows that the AprioriAll algorithm scales correctly.

2,000 frequent patterns were extracted from a 100,000-occurrence sequence. The length of these patterns varies from 2 to 4 and the value of the support of the most frequent one is equal to 704. Moreover, it must be noted that 80% of the detected patterns have a support value inferior to 10 and 60% have a support value inferior to 5; these patterns are not very frequent, given the size of the alarm log.

C. Finding alarm correlations

The set of detected frequent patterns is too huge to be analyzed by an expert in a reasonable time. Moreover, most of the patterns with a small support value (less than 10) cannot really contribute to alarm rationalization because the alarms of these patterns are not always correlated; they have been associated to form a pattern several times, but not every time they occur. Hence, it has been decided to rank the frequent patterns to be analyzed by the expert according to the value of their supports; the patterns with the longest supports will be analyzed first.

D. Examples of alarm correlations

The first example that will be detailed is a case of logical correlation which has been highlighted from a frequent pattern whose length and support are respectively equal to 3 and 52. The alphabet of this pattern contains three alarms $a_1$, $a_2$ and $a_3$. Expert analysis shows that these three alarms are defined for the three inputs of a voter. The consequence of this inappropriate setup is that when two or three of these alarms occur only one alarm occurrence is necessary (a majority of the voter inputs is over the defined threshold), whereas when only one of these alarms occurs no alarm occurrence should happen, because a majority of the voter inputs is below the threshold.

Hence, alarm rationalization can be easily achieved by removing the three existing alarms and defining a new alarm at the output of the voter. This modification saves $2 \times 52 = 104$ alarm event occurrences.

The second example is a case of combined physical and logical correlations which has been highlighted from a frequent pattern whose length and support are respectively equal to 4 and 657. The alphabet of this pattern contains four alarms $b_1$, $b_2$, $b_3$ and $b_4$. Expert analysis shows that all these alarms are related to the water level in a tank. $b_1$ and $b_2$ are related to measured values of this level whereas $b_3$ and $b_4$ depend on the percentages of the capacity of the tank which are computed from the measured values; hence, $b_3$ and $b_4$ are logically correlated to $b_1$ and $b_2$ and purely redundant. Moreover, the physical placement of the sensors which measure the level values from which $b_1$ and $b_2$ are defined is such that, with faultless sensors, $b_1$ (very high level reached) cannot occur if $b_2$ (high level reached) did not occur previously. Therefore, $b_1$ is physically connected to $b_2$, but the opposite is not true. This explanation is confirmed by the fact that occurrences of $b_2$ which are not followed by occurrences of $b_1$ can be found in the alarm log, while every occurrence of $b_1$ is preceded by an occurrence of $b_2$.

Alarm rationalization will be achieved in that case by first, removing the occurrences of $b_1$ and $b_4$, then conditioning the occurrence of $b_2$, alarm to the non-occurrence of $b_1$, i.e. if $b_2$ occurs, the alarm filter will store this event and transmit an alarm to the operator only if $b_1$ does not occur after this occurrence of $b_2$. These modifications save $3 \times 657 = 1,971$ alarm event occurrences.

E. Discussion

Table II presents the results of the overall analysis; only the most frequent patterns are considered, for space reasons. Each row of this table corresponds to a pattern which has been
The contribution of this analysis to alarm rationalization can be easily quantified as follows. For every row, multiplying the values of the first and fourth columns provides the number of alarm occurrences of the considered pattern that can be removed from the alarm log. The first pattern, for instance, has a support value of 704 and contains 1 removable alarm; hence 704 alarm occurrences could be removed in the log. This reasoning can be applied to all the frequent patterns.

Globally, filtering the redundant alarms leads to, for the case study, an average number of alarms per day equal to approximately 2,200 and a peak alarms/10 min to 300. If the contribution to the first feature is weak (8% of the alarm occurrences are filtered), the second feature, which determines significantly the overload of the operator, is more impacted (reduction of 35%).

Current investigation is aiming at improving these systems by developing more efficient pattern detection algorithms. The current version is indeed looking for a fully constrained form of pattern, where the number and order of alarm occurrences are fixed; allowing more flexible definitions of pattern, where the order of occurrences could be changed, some occurrences be absent or new alarm occurrences introduced, is a promising way to improve alarm rationalization.

V. CONCLUSION

Alarm management systems are playing an important role for safety and efficiency of critical plants. Unfortunately, these systems are suffering from alarm floods which overwhelm the operators then lessen the overall performances. To tackle out this issue, redundant alarms must be filtered before they reach the operators; construction of redundant alarms filters requires these alarms have been previously identified, however.

This paper has shown that redundant alarms can be detected and interpreted by using a pattern mining technique that extracts frequent patterns from an alarm log and expert analysis of these frequent patterns. The benefit of this contribution to reduce peak alarms/10 min has been shown. Implementation of these results in the GE Industrial Internet of Things Predix® environment is an-going work.

To improve peak alarms/10 min reduction and better lessen the average number of alarms per day, alternative pattern detection techniques are under investigation. Defining more flexible patterns as well as taking into account physical time are promising ways to meet these objectives.

ACKNOWLEDGMENT

The authors would like to thank General Electric Digital Foundry Europe for their cooperation and for having given the material and temporal resources necessary to conduct this study.

REFERENCES


