HIERARCHIES OF ALARMS FOR LARGE DISTRIBUTED SYSTEMS

Marco Boccioli, Manuel Gonzalez Berges, Vasileios Martos, CERN, Geneva, Switzerland
Oliver Holme, CERN, Geneva, Switzerland and ETH Zurich, Switzerland

Abstract

The control systems of most of the infrastructure at CERN make use of the SCADA package WinCC Open Architecture by ETM*, including successful projects to control large scale systems such as the Large Hadron Collider (LHC) accelerator and associated experiments.). Each of these systems features up to 150 supervisory computers and several millions of parameters [1]. To handle such large systems, the control topologies are designed in a hierarchical way (i.e. sensor, module, detector, experiment [2]) with the main goal of supervising a complete installation with a single person from a central user interface. One of the key features to achieve this is alarm management (generation, handling, storage, reporting). Although most critical systems include automatic reactions to faults, alarms are fundamental for intervention and diagnostics. Since one installation can have up to 250k alarms defined, a major failure may create an avalanche of alarms that is difficult for an operator to interpret. Missing important alarms may lead to downtime or to danger for the equipment.

This paper presents the features and benefits of hierarchical alarms.

INTRODUCTION

In a SCADA system, alarm handling plays an important part of the final implementation. Fundamentally, alarm handling is based on limit and status checking and performed in the data servers [3]. For instance, if the value of the entity deviates from the good range, an alarm is automatically generated on the supervisory station in order to inform the operator. In some cases, alarm events must be acknowledged by the operator. More complex rules, involving several measurements and statuses, can also be attributed to an alarm.

In a large, distributed SCADA system such as those implemented for the Detector Control Systems (DCS) for the LHC experiments at CERN, the amount of defined alarms can be very high. In case of an avalanche of alarms, the analysis of a normal, flat view of alarms might be problematic for a single operator. Due to the size and complexity of an LHC detector, its control is implemented in a hierarchical way [4]. Therefore, the concept of handling alarms hierarchically follows naturally.

CERN has been collaborating with ETM to implement an advanced technique to show alarms in a hierarchical way, reducing the amount of individual alarms displayed while giving the possibility to get an immediate overview of a problem.

*ETM professional control GmbH - www.etm.at

ALARM HANDLING IN WINCC OA

WinCC OA implements a powerful and sophisticated alarm handling framework. Here follows a brief introduction to its concept.

Architecture

The WinCC OA package has a modular architecture, as schematized in Figure 1. Each type of module is a running process, called a manager, handling specific tasks. In the case of alarm handling, the value of an item coming from any driver (D) is processed by the Event Manager (EV). EV handles the alarm generation (e.g. is the value in good or bad range?). In the case an alarm is generated, EV sends the alarm information to the Data Manager (DB) for archiving, and to all relevant managers for displaying or for processing. Figure 1 shows an example of data flow from a driver D_2 through EV, to DB, CTRL and UI_2.

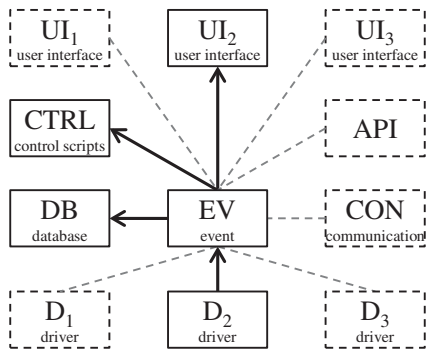

Figure 1: WinCC OA architecture. The continuous lines show an example of alarm handling data flow.

An alarm definition is associated to an entity to be monitored. Depending on the type of data to be checked, the defined alarm can be of different natures (Table 1): continuous (thresholds) or discrete (a specific set of values defined as bad). The definition of alarms consists of one or more good/bad ranges, each range including different properties (value limit, text, hysteresis, priority, colour, acknowledgeable, etc.).

Table 1: Types of Data and Associated Types of Alarms.

Data type	Alarm type
boolean	discrete, multi-instance
real	continuous, discrete, multi-instance
integer	continuous, discrete, multi-instance
double word	discrete, multi-instance

Copyright © 2014 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-139-7

Example: Figure 2 shows an alarm associated to a water tank level. The alarm has the following properties:

- Good range: below 2m
- Warning range: above 2m
- Fatal range: above 3m

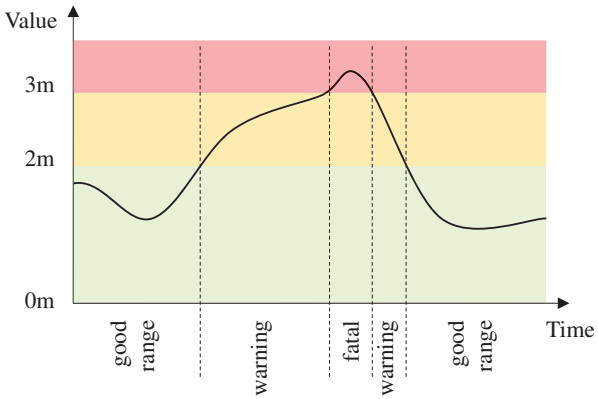

Figure 2: A continuous alarm with two bad ranges.

When the value of the monitored item is out of the good range for the first time (i.e. exceeding 2m), then an instance of the alarm (referred to in this paper as simply an alarm) is triggered, or *came* in WinCC OA terminology. In this example, it is a *warning*. The value of the level then increases to above 3m, reaching the *fatal* range. In such situation, WinCC OA generates a second alarm. At the visualisation level, both alarms can be shown or, optionally, only the most recent or highest priority alarm can be shown.

If defined in the alarm, the operator must send an acknowledgement to prove that the alarm was seen. For this reason, alarms have a life cycle of states such as: no alarm, *came unacknowledged, came acknowledged, went unacknowledged*.

Single Alarms

Continuous and discrete alarms are both called single alarms, since they have a one-to-one connection to the data item to be monitored.

Multi-instance Alarms

An alarm can be connected directly to an external source. In this case, the external source creates the alarm, which is then received by the EV manager. The EV propagates the externally generated message to the UI and to the DB. Multiple instances of alarms can be sent by the device and each one can have different properties.

Summary Alarms

A summary alarm is an aggregation of alarms and it has its own attributes (e.g. colour, class, priority). If one or more alarms in the aggregation list are generated, then the summary alarm generates its own alarm. The class is inherited from the active single alarm with highest priority (only if no class is specified on the summary).

Example: Figure 3 shows a summary alarm with four single alarms. Two single alarms came, so the summary alarm also comes. Given that the red single alarm has the highest priority, its class is inherited by the summary alarm.

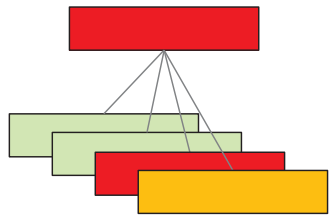

Figure 3: A summary alarm with four single alarms. The boxes represent: green = no alarm, yellow = *warning* alarm, red = *fatal* alarm.

Hierarchical Alarms

Summary alarms tend to multiply the visible alarms as the single and summary alarms are both shown by default. It is possible to show only summaries or only children alarms to the operator, but this is not sufficient for complex cases such as hierarchies of alarms. A hierarchical alarm can monitor a mixture of single or summary alarms, called children alarms. Being based on a summary alarm, it becomes active as soon as one or more children alarms come. In addition, the hierarchical alarm features a customisable visibility threshold. If the amount of *came* children alarms exceeds the threshold, the children alarms are hidden and only the hierarchical alarm is shown. This provides a method of alarm reduction to limit the number of alarms seen by the operator. The concept of *hidden/shown* differs from the concept *came/went*. *Came/went* relates to the generation of alarms based on alarm conditions, whereas *hidden/shown* relates solely to the visualisation of an existing alarm.

The alarm panel can show alarms in a flat way (all *came* alarms are listed) or in a hierarchical way (all alarms not flagged as hidden are listed). In the latter case, a hierarchical alarm hiding its children will be the only one appearing in the alarm list on the operator screen.

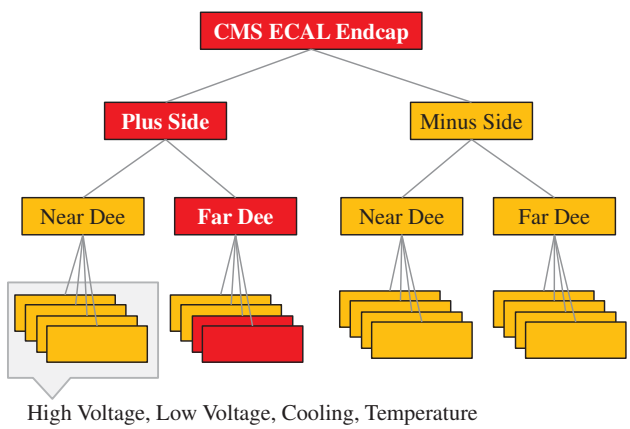

Figure 4: An example of how hierarchical alarms could be implemented in part of an LHC Detector Control System.

Example: Figure 4 shows a hierarchical alarm tree with some alarms. All the children alarms *came*. With the thresholds properly set, only the top alarm would be

ISBN 978-3-95450-139-7

Copyright © 2014 CC-BY-3.0 and by the respective authors

shown to the operator. Without the alarm reduction and hierarchical alarm list, 23 alarms would be displayed.

For diagnostics purposes, the list of active children alarms that triggered the hierarchical alarm can be shown on request.

THE CHALLENGES OF HIERARCHICAL ALARMS

The definition and specification of hierarchical alarms was designed to handle any possible alarm conditions. The result is a robust mechanism that is able to provide consistent alarm reporting, as described in this chapter. For simplicity, alarms with only a single bad range are used in the examples, unless otherwise stated.

Custom Thresholds

Consider the case in Figure 5. The threshold of the hierarchical alarm B_1 is t=2: this means that, if two or more children alarms *came*, B_1 hides them and shows itself. If less than two children alarms *came*, then B_1 hides itself and the children alarms are visible. In the example, the alarm B_1 summarizes four possible children alarms, two of which *came*. Therefore B_1 hides its children and shows itself.

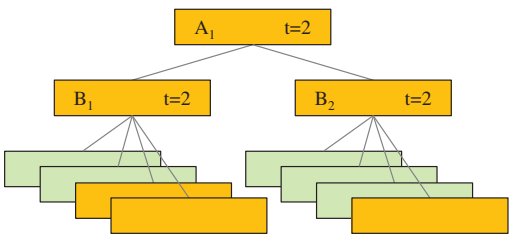

Figure 5: Alarm hierarchy with specific thresholds.

The hierarchical alarm B_2 also has threshold t=2. Since only one child alarm *came*, B_2 is hidden, and the child alarm is shown. A_1 has t=2 and two children (B_1 and B_2). Hierarchical alarms only count children that *came* and are visible. Since B_1 is the only visible child, then A_1 is hidden and B_1 is visible. The result of this configuration is illustrated on Figure 6.

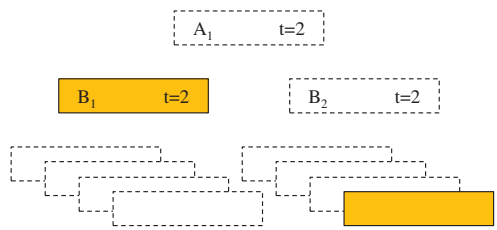

Figure 6: The visible alarms are with continuous borders.

In this way, it appears that the threshold can be used to freely adjust the number of alarms shown in a hierarchy in a specific problem situation, enabling an optimal amount of detail to be shown. However, such a customisation of alarm information detail is not practical as explained in the following section.

Floating Alarms

Consider now the case in Figure 7. Hierarchical alarm B_1 has threshold t=2. Since only one child *came*, B_1 is hidden and the child is visible. On the other hand, B_2 and B_3, having threshold t=2, show themselves and hide their two children that *came*.

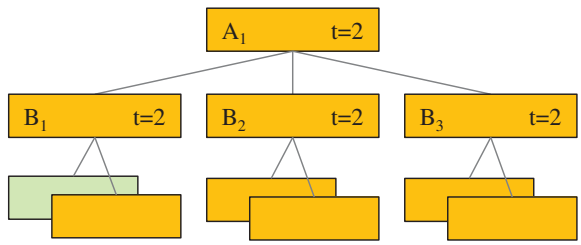

Figure 7: Specific thresholds may cause floating alarms.

The alarm A_1 (t=2) sees B_1, B_2 and B_3 as *came*, but sees only B_2 and B_3 as visible. This will show A_1, hide B_2 and hide B_3. The problem is visible looking at Figure 8.

If a hierarchical alarm is visible, any other alarm lower in the hierarchy is supposed to be hidden. This is not the case in the example, and it is an unwanted situation, since the operator may be confused (is that single alarm belonging to the same system as the alarm A_1?).

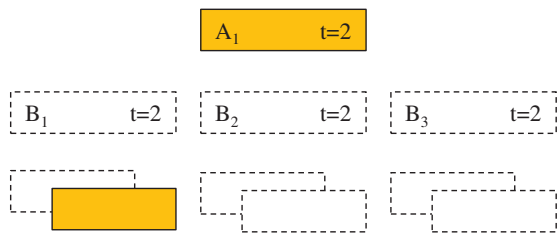

Figure 8: A floating alarm.

The floating alarm that is shown in the example might even be of lower priority than the alarms that are hidden, which again could mislead the operator (why is this lower priority alarm given such importance?).

The solution to this misleading situation is the introduction of automatic thresholds. This option sets, for each hierarchical alarm, the threshold value equal to the total amount of children alarm definitions.

By automatically setting the threshold, the previous example of Figure 7 would become as in Figure 9.

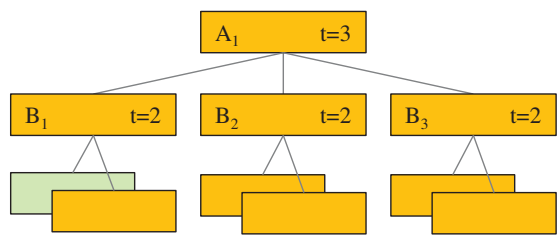

Figure 9: Automatic thresholds.

In this case, each hierarchical alarm has threshold equal to the amount of children alarm definitions (the threshold does not depend on whether the alarms came or not). Due to the structure of the example hierarchy, B_1, B_2 and B_3

Copyright © 2014 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-139-7

remain with t=2. Since A_1 has t=3, it hides itself and show its children. The result is visible in Figure 10.

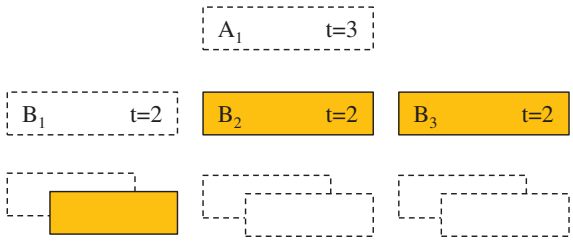

Figure 10: Automatic thresholds avoid floating alarms.

In this case, the single alarm below B_1 is visible, and no superior alarm is visible. B_2 and B_3 are visible, and all their children alarms are hidden. This is a consistent situation.

With automatic thresholds, the threshold value is updated whenever the hierarchy is modified, which ensures the alarm reduction mechanism continues to provide valid results.

The automatic threshold reduces the possibilities for customising the quantity and detail of alarms to be shown in case of alarm avalanches. It also implies design restrictions on the structure of alarm hierarchy that is implemented. For instance, a shallow, wide hierarchy where each hierarchical alarm has many children can still generate many visible alarms before the reduction mechanism is activated. A better approach is to create a deep, narrow hierarchy with fewer children per hierarchical alarm. The automatic threshold is optional but strongly recommended as it is the only way to ensure a clear and consistent alarm display to the control system operator.

Dealing with Multi-range, Single Alarms

Since WinCC OA considers each range crossing as a new alarm instance, a hierarchical alarm counts the same single alarm multiple times.

Example: having the tank level in Figure 2, assume a level increase from 1.8m to 2.1m: the *warning* range is reached and *warning* alarm comes. The alarm is a child of the summary B_1 from the previous example, as shown in Figure 11a.

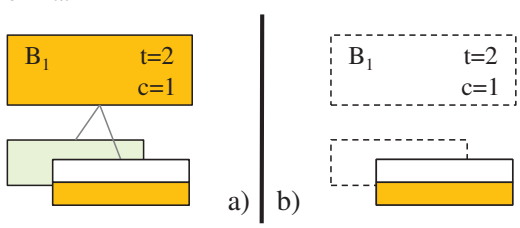

Figure 11: Two-range alarm, first range reached.

In this case, the hierarchical alarm B_1 counts one active child alarm (c=1). As a result, the automatic threshold t=2 is not reached, therefore B_1 is hidden, and the single alarm is shown (Figure 11b). Now, from Figure 2, the level reaches 3.1m, the *fatal* range is crossed and a *fatal* alarm *came*. This is a second instance of alarm. The alarm B_1 then counts c=2 (Figure 12a). Since the auto threshold is set to t=2, the alarm B_1 hides the child alarm and will show itself (Figure 12b).

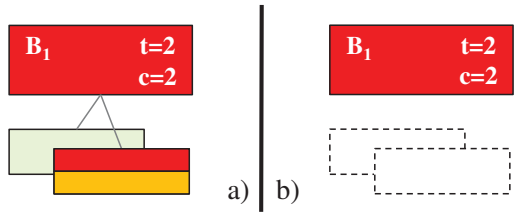

Figure 12: Two-range alarm, second range reached.

With the automatic threshold and the alarm counting scheme of WinCC OA, multiple range alarms cause the alarm reduction mechanism to activate earlier than expected. Generally, the minimum number of children that need to have alarms before the threshold is reached is equal to the number of children divided by the number of alarm ranges on each child. In situations where each child has a different number of alarm ranges, this behaviour is specific to the exact combination of children alarms.

In order to avoid possible misleading situations, it is advised not to mix single and summary alarms in the same hierarchical level.

APPLICATIONS AND CONCLUSIONS

The collaboration between CERN and ETM for the specification and validation of hierarchical alarms has been fruitful enabling avalanches of alarms to be analysed in an easier way.

Hierarchical alarms have raised great interest among the experiment collaborations at CERN. Since the fine-tuning of the hierarchical alarms features has been completed recently, their adoption has just started. Possible applications include integration into hierarchical controls like the Finite States Machines [5]. This can provide an alarm topology consistent with the controls hierarchy, with the two hierarchies offering complementary information to give optimal support to the decision making process of the operator.

REFERENCES

[1] A. Augustinus et al., "The ALICE Control System", ICFA 2008, Menio Park, USA, p.97.
[2] A. Augustinus et al., "The ALICE Control System", ICFA 2008, Menio Park, USA, p.95.
[3] A.Daneels, W.Salter, "What is Scada?", ICALEPCS 1999, Trieste, Italy, p.341.
[4] Yi Ling Hwong et al., "An Analysis of the Control Hierarchy Modeling of the CMS Detector Control System", CHEP 2010, Taipei, Taiwan.
[5] G. de Cataldo et al., "Finite State Machines for integration and control", ICALEPCS 2007, Knoxville, Tennessee, USA.

Copyright © 2014 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-139-7