ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA118

IMPROVING ALARM HANDLING FOR THE TI OPERATORS BY INTEGRATING DIFFERENT SOURCES IN ONE ALARM MANAGEMENT AND INFORMATION SYSTEM

M. Bräger*, M. Bouzas Reguera, U. Epting, E. Mandilara, E. Matli, I. Prieto Barreiro, M. P. Rafalski,
CERN, Geneva, Switzerland

Abstract

CERN uses a central alarm system to monitor its complex technical infrastructure. The Technical Infrastructure (TI) operators must handle a large number of alarms coming from several thousand equipments spread around CERN. In order to focus on the most important events and improve the time required to solve the problem, it is necessary to provide extensive helpful information such as alarm states of linked systems, a geographical overview on a detailed map and clear instructions to the operators. In addition, it is useful to temporarily inhibit alarms coming from equipment during planned maintenance or interventions. The tool presents all necessary information in one place and adds simple and intuitive functionality to ease the operation with an enhanced interface.

INTRODUCTION

Alarm systems are an essential component in all environments where technical equipment is supervised by control systems. At CERN, different alarm system solutions have been designed over time and were mostly dedicated to cover precise expert systems. With the grouping of several control rooms in the CERN Control Center (CCC) and the increasing infrastructure for LHC operation, more equipment needed to be supervised by the Technical Infrastructure (TI) operators. The existing tools and their technology were developed in the last millennium and it was difficult to maintain these with young developers and modern tools.

Today's alarm system integrates state of the art technologies and provides the operators with all functionality for efficient alarm handling. Each alarm has information about precise location, fault cause and consequences and the actions to be done. The information is concentrated in a tool that easily allows to identify alarm avalanches and provides detailed information for single alarms. The official CERN map is used to give a geographical overview with animated live alarm information. Extensive filtering possibilities have been introduced, which makes the tool usable not only for the TI operators, but also for the equipment specialists and other services like the CERN fire brigade.

The paper provides an overview of the chosen architecture and design decisions for ALIS to deliver an improved alarm handling experience to the end-users.

* Matthias.Braeger@cern.ch

MOTIVATION AND FUNCTIONALITY

Today's core requirements for a sophisticated alarming tool have not changed much since the early 90's [1]. Only data variety and velocity have tremendously increased. In first place stakeholders want a service that is agnostic to all kind of errors and helps to quickly get an overview in critical situations. Due to the regular feedback of the TI operators CERN's in-house alarm systems became over the years more and more sophisticated and today we are technically able to provide a state-of-the-art ALarm Information System (ALIS) that integrates many long-requested features and consolidates the current diversity of tools. To fulfill modern user needs ALIS must be accessible from all kind of devices including mobile devices. This naturally lead to the choice of creating a web application instead of a traditional industrial SCADA desktop program.

The application comes with an intuitive search interface to browse through more than 170'000 TI alarms. The biggest alarm sources are electrical devices, security related equipments such as access doors or smoke- and fire detectors, and pre-calculated alarms from other SCADA devices. As speed was one of the main requirements to the new system, search results are non-blocking and provided in between 300ms (search for all active alarms) to maximum few seconds. The filtered alarms can then be displayed either as a list or on an interactive map including live updates. Furthermore, the user can consult all alarm details from the so-called Single Alarm Page that is opened when selecting a particular alarm.

To prevent unauthorised access to the sensible content the tool makes use of the CERN OAuth2 Single-Sign-On service.

ARCHITECTURAL OVERVIEW

The ALIS system is based on an existing open-source infrastructure called CERN Control and Monitoring Platform (C2MON) [2] [3] [4] (see also Fig. 1). C2MON is used at CERN as back-end for the Technical Infrastructure Monitoring (TIM) [5] service which is acquiring and storing sensor data for a multitude of SCADA applications and user groups. TIM also provides live updates for the previously mentioned 170'000 TI alarms, which are evaluated out of 230'000 input tags coming from various hardware and software sources. The service has a separate web-based and domain specific workflow-driven data entry system called MoDESTI [6]. It allows to enter and review all relevant data for a proper alarm treatment. MoDESTI takes care of validating the entered information before storing into the TIM

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA118

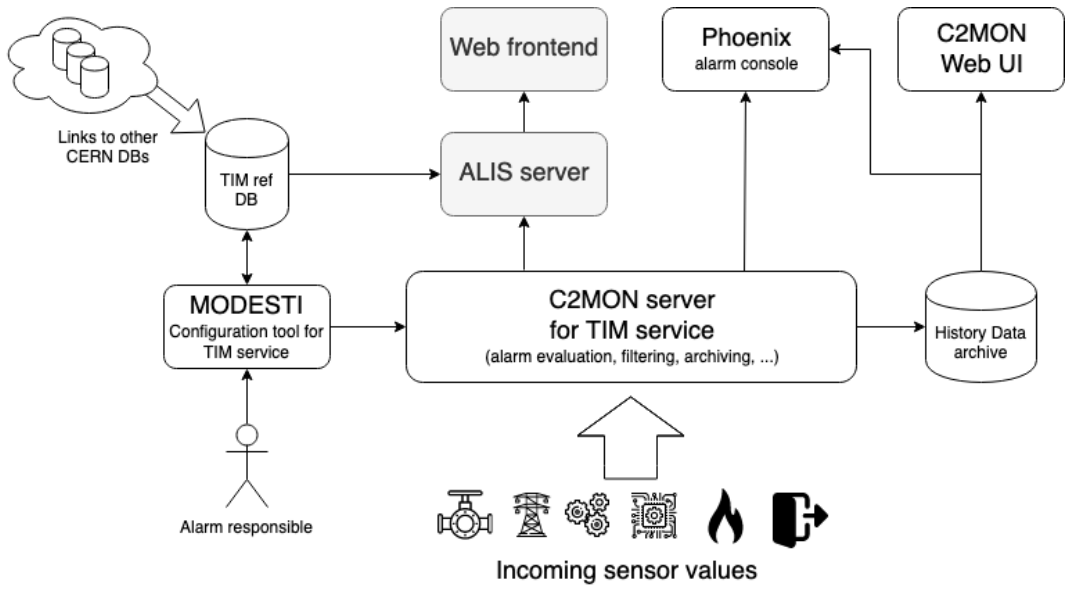

Figure 1: Architectural overview.

reference database and handling the online re-configuration of the TIM-C2MON server.

C2MON has the concept of attaching metadata information to tags and alarms, which is used in the TIM context to configure alarms together with the core information. Metadata are kept in-memory in C2MON, which has next to speed the advantage that ALIS can still run in a degraded mode and provide basic information for incoming alarms to the operators.

Alarms belong to at least one context related category, for instance to the category *Electrical Alarms*. This allows users to create individual channel configurations by selecting categories of their interest. Together, with all other metadata entered via MoDESTI the user profits from powerful filtering combinations in ALIS which can be extremely helpful when trying to understand complex alarm scenarios.

BACKEND

The server back-end of ALIS is written in Java and Spring Boot [7]. Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications with an embedded Apache Tomcat [8] server for running the web front-end.

During the startup phase the back-end is initialising local caches for the most re-used data, that is:

- active alarms,
- geo-coordinates of the CERN buildings,
- geo-coordinates for equipment that have been integrated into the CERN ArcGIS [9] map.

This is providing the high responsiveness of the application and avoids to constantly request data from the TIM-C2MON instance or the reference database.

The list of all active alarms is retrieved with the C2MON client API and then continuously synchronised from a live update listener thread. This thread is also responsible of propagating the alarm value updates to the subscribed clients via web-socket, which is realized with the Atmosphere framework [10].

Query Optimisation Approach

The main use case of ALIS is searching and filtering for alarms, in particular for active ones. Therefore, special care was put on optimizing the query speed for this case.

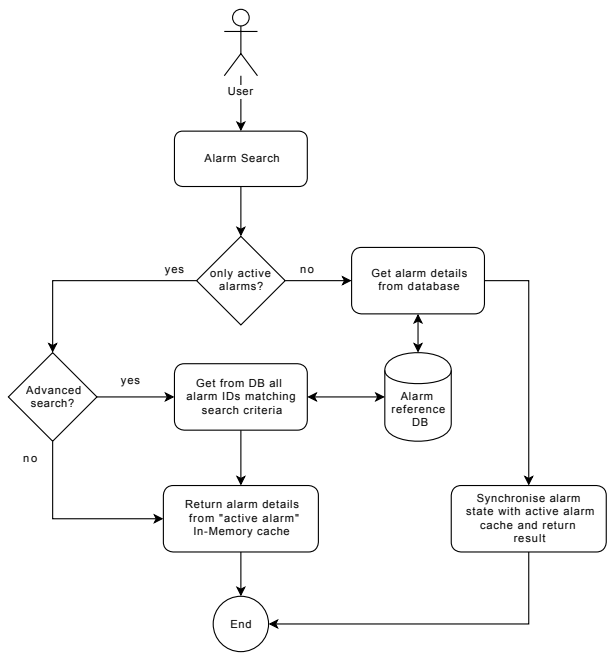

Figure 2: Workflow for optimising speed of search results.

The workflow in Fig. 2 aims to illustrate the concept used for the query optimiser logic. When receiving a query via HTTP GET request the back-end checks first of all, if the caller is only interested in active alarms. If so, it can

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA118

leverage from the local in-memory cache which is keeping an up-to-date list of all active alarms. Depending on the other filter criteria, it might even be possible to extract the result set without database query. This is for instance the case when filtering active alarms by priority or name. More complex search queries are realised against the database by passing directly the RSQL [11] search string, which used to encode the search criteria on the front-end, to the Spring Data JPA [12] layer. This flexible and powerful approach avoids writing a lot of additional search logic and brings good performance results.

Before returning the result set to the requester it is necessary to synchronise the current alarm state with the local cache as the reference database is not providing this information due to the fast update rate.

FRONT-END

Developing the user interface as a web application allows us to use a single application to target several clients, from desktop to mobile devices. It is developed using Vue.js [13] front-end framework and makes use of Vue's centralised state management (Vuex [14]). The Single Page Application (SPA) is composed of the four following components:

- Search Interface
- Interactive Map
- Alarm List
- Single Alarm Page

Figure 3 illustrates the font-end workflow implemented to display live alarms from a search request. The user starts by entering some search parameters in the search form that are sent to the ALIS server as two separate requests. The first is a GET request to the REST API that returns an array con-taining all the requested alarms. In parallel a second request is sent to the WebSocket service to start a subscription to status changes for the same list of alarms. The WebSocket client is implemented with the Atmosphere framework [10] and is responsible for providing live updates. Any change in status triggers an update that is sent to the client. Both re-quests store their results in the central store. Vue/Vuex takes care of automatically notifying the active component so that the data driven user interface immediately updates to reflect the changes in the data source. Separating the status man-agement from the display layer allows for a more reactive user interface since alarms don't need to be fetched again from the server when switching between different display modes. The default interface displays the alarms in a list, sorted by time, and colour coded depending on their priority level. Thanks to the integration with CERN's GIS portal [15] alarms that contain geographical location information can be displayed on a map. A third page is provided to display detailed information of a single alarm and requires a new call to the server to fetch the information not distributed via WebSocket. Those additional data are merged into the alarm object in the store so that subsequent access to the detailed page don't require more calls to the server.

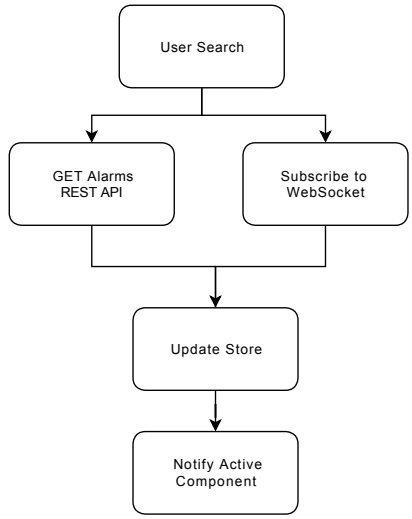

Figure 3: Front-end Workflow.

Search parameters are encoded into an RSQL [11] string before sending them to the server. RSQL is particularly suited for communicating with REST APIs and allows us to provide the user with an easy way to save and share complicated search queries in the form of URLs.

OUTLOOK

The current implementation of ALIS is based on existing data sources and data structures. The data on the different information displays is generated and maintained by tools outside the ALIS framework, which makes it difficult to provide timely updates for changes in the alarm descriptions, their causes/consquences or actions to be done. This data is today entered via the MoDESTI procedure or by the current TI alarm console (Phoenix), which is realised as Java desktop application and needs to be maintained separately.

Currently, operators are still using a separate desktop application as main alarm console (Phoenix), which is linked to ALIS for providing more information. As the ALIS functionality is overlapping and extending the Phoenix functionality, it is envisaged to integrate the missing parts directly in its framework.

The ALIS architecture based on web technology allows to integrate information from different data sources quite easily. Modules that connect to external systems like the asset management system can show dependencies and the state of neighbouring installations. Direct access to existing online analysis tools for equipment will be integrated to complete the live information for the operators and equipment specialists in order to ease troubleshooting and repair.

In the future, the presented system could be extended for a generic SCADA web use case following the same architectural principle. By distributing live data via WebSocket, SVG images can be animated by making use of modern JavaScript libraries such as Snap.svg [16].

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA118

CONCLUSION

Using the current implementation of the ALIS system was accepted from the beginning by the users. Improvements and missing features were added timely by the development team. Tests showed a very good robustness and fast availability of live data. This confirms the chosen architecture and encourages to add more modules and functionality to provide this service to an extended number of users. The availability of ALIS on all web-capable devices enlarges its usage to teams in the field and provides live status information for nearby equipment. By using modern technologies, it should also be easier to find qualified personnel for maintenance and future extensions.

REFERENCES

[1] M. W. Tyrell, *The LEP Alarm System*, in *Proc. ICALEPCS'91*, Tsukuba, Japan, Nov. 1991, pp. 254-259.

[2] M. Brightwell, M. Bräger, A. Lang, and A. Suwalska, "A Customizable Platform for High-availability Monitoring, Control and Data Distribution at CERN", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper MOPMS037, pp. 418–421.

[3] M. Bräger, M. Brightwell, E. Koufakis, R. Martini, and A. Suwalska, "High-Availability Monitoring and Big Data: Using Java Clustering and Caching Technologies to Meet Complex Monitoring Scenarios", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, paper MOPPC140, pp. 439–442.

[4] C2MON project
https://github.com/c2mon/c2mon

[5] A. Suwalska, M. Brightwell, M. Bräger, E. Koufakis, R. Martini, and P. Sollander, "Integration, Processing, Analysis Methodologies and Tools for Ensuring High Data Quality and Rapid Data Access in the TIM* Monitoring System", in *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, paper TUPPC029, pp. 615–618.

[6] R. Martini, M. Bräger, J. L. Salmon, and A. Suwalska, "Tools and Procedures for High Quality Technical Infrastructure Monitoring Reference Data at CERN", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 1036–1039. doi:10.18429/JACoW-ICALEPCS2015-WEPGF141

[7] Spring Boot
https://spring.io/projects/spring-boot

[8] Apache Tomcat
http://tomcat.apache.org

[9] ArcGIS
https://www.arcgis.com

[10] Atmosphere framework
https://github.com/Atmosphere/atmosphere

[11] RSQL parser for Java
https://github.com/jirutka/rsql-parser

[12] Spring Data JPA
https://spring.io/projects/spring-data-jpa

[13] Vue.js framework
https://vuejs.org

[14] Vuex state management pattern + library
https://vuex.vuejs.org

[15] CERN GIS portal
https://gis.cern.ch

[16] Snap.svg JavaScript library
http://snapsvg.io