ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP066

VISUALIZATION TOOLS TO MONITOR STRUCTURE AND GROWTH OF AN EXISTING CONTROL SYSTEM

O. Pinazza[†,1,2], A. Augustinus[2], P. M. Bond[2], P. Chochula[2],
A. Kurepin[3], M. Lechman[2,4], D. Voscek[2]
[†]corresponding author; [1]INFN, Sezione di Bologna, Italy; [2]CERN, Geneva (CH);
[3]affiliated with an Institute Covered by a Cooperation Agreement with CERN, Geneva, (CH);
[4]University of the Witwatersrand, Johannesburg, South Africa

Abstract

The ALICE experiment has been in operation for 15 years at the LHC. During its life several detectors have been replaced, new instruments installed, and readout technologies have changed. The control system has therefore also had to adapt, evolve and expand, sometimes departing from the symmetry and compactness of the original design.

ALICE is a large collaboration, where different Institutes and Universities from over 40 Countries contribute to the development of the detectors and their control systems. For the central coordination it is important to maintain the overview of the integrated control system to assure its coherence. Tools to visualize the structure and other critical aspects of the system can be of great help and can highlight problems or features of the control system such as deviations from the agreed architecture.

This paper describes how existing tools, such as graphical widgets available in the public domain, or techniques typical of scientific analysis, can be adapted and help assess the coherence of the development, revealing hidden weaknesses and highlighting the interdependence of parts of the system.

INTRODUCTION

ALICE [1] is one of the four big experiments at the LHC. Operational since LHC started in 2007, ALICE has taken proton and heavy-ions collisions data during Run1 (2009-2013) and Run2 (2015-2018). During Long Shutdown2 (LS2, 2018-2022) ALICE has undergone an important upgrade consisting of the installation of new detectors: Inner Tracking System (ITS), Muon Forward Tracker (MFT), Fast Timing triggering detectors (FDD, FT0 and FV0); a new readout system for the main Time Projection Chamber (TPC) detector, and a brand new software model called O^2, combining offline and online systems [2].

The ALICE Detector Control System (DCS) [3] has successfully accompanied these developments and adapted to new technologies and requirements, while at the same time ensuring the continuity of operations and control of different devices.

In order to verify the coherent and safe development of the different systems and the integration of new detectors, we have introduced data analysis and visualization techniques, which also enabled us to assess the interdependence and safety of the systems.

THE ALICE CONTROL SYSTEM

The ALICE DCS is strongly based on SIMATIC WinCC Open Architecture (WinCC) [4], as well as the other CERN experiments, but it is not limited to it. The WinCC software core is running on 63 worker nodes connected to the 15 ALICE detectors, interfaced by 17 WinCC multiuser operator nodes, where experts can login to access monitoring and control interfaces; in the backend, 19 more nodes host the WinCC central services, 20 linux nodes run drivers and custom software, together with a database farm, several fileservers, gateways and other specialized nodes, and several different embedded systems.

While WinCC offers a wide choice of graphical widgets allowing to develop fancy panels, to monitor the detectors' hardware and control their devices, it lacks sometimes tools and interfaces to represent the connection and interdependence between different operating system and to unveil the complexity of the overall DCS.

This type of information can help in revealing unexpected interdependencies and weaknesses, or design flaws. It is characterized by being rather static, compared to the mutability of typical DCS parameters, and can be produced whenever required, occasionally, from an external host.

The data presented here has been obtained after the LS2 upgrade. Some of the images were inspired by books on the graphic representation of hierarchical data using trees or spheres [5]. All data is extracted with *python*, Microsoft Powershell and WinCC CTRL++, and visualized with graphical libraries for *python* and *javascript,* available online [6, 7]. It will be interesting to observe the variations in time, especially when new subsystems will be added to the DCS during LS3, after the next big ALICE upgrade [8].

PROJECTS INTERDEPENDENCE

One of the features of WinCC is the capability to interconnect systems through the DIST managers, thus allowing access to datapoints in a remote project. While the DIST mechanism is overall rather efficient, its monitoring and control is more obscure. It can happen that, under some special circumstances like a timeout during a project restart, or systems restarting in random order after a general reboot, two or more WinCC projects that are supposed to be connected, in reality are not. This situation can stay unnoticed and be difficult to emerge. To limit this kind of problems and prevent the malfunction of a system from

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP066

affecting others, it is good practice to avoid unnecessary dependence between systems, and to maintain a snapshot of the expected and authorized interconnections.

The ALICE Controls Coordination (ACC) policy foresees that a detector project can connect to all nodes belonging to the same detector, if needed, and to some special central nodes, maintained by the ACC itself. Connection to the central nodes is necessary to participate to the data taking operations, and to be managed by trained shifter operating from the ALICE control room. Centrally managed nodes are also setup to collect, rework, archive and publish data related to the environment, the infrastructure, and parameters provided to/from the LHC. By offering this type of service centrally, access to the most important parameters can be ensured through centrally maintained channels.

In order to monitor the distributed system, we have checked the status of *allowed* and *active* connections between nodes: while allowed connections are listed in a centrally maintained configuration file, active connections can be obtained inside the WinCC environment.

Figure 1 is the map of DIST connections in ALICE, built in *javascript*. Data about the allowed and active DIST connections is generated by a WinCC script in a *json* format, and the D3.js library for web visualization is used. All projects are aligned outside the circle, and incoming/outgoing connections are drawn in blue/red lines.

A quick look to the visualization confirmed us that all *active* connections are limited to projects belonging to the same detector (grouped together alongside the circle), or to the central projects, which are especially designed for this purpose. A more careful comparison between the expected and the actual configurations confirmed that all systems comply with the authorized interdependency map.

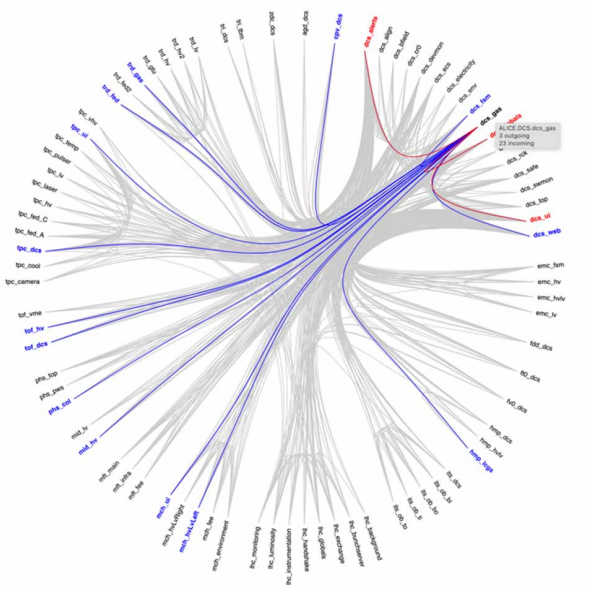

Figure 1: DIST connections wheel, exposing the connections to/from the gas system, as an example. All lines represent the expected and authorized connections. Blue lines are incoming connections and red lines are outgoing connections.

WINCC PROJECTS COMPLEXITY

The control system of each of the 15 detectors forming part of ALICE is developed on one or more worker nodes, depending on the complexity and partitioning of the detector. WinCC and other basic software is installed on each node; on top of that, detector experts develop custom files as:

- panels, XML (or WinCC PNL) files representing the graphical interface with hardware and procedures
- pictures, graphical widgets and maps and included in panels
- libraries, C-like code (WinCC ctrl++ language) containing reusable detector specific code
- scripts, C-like code (WinCC ctrl++ language) executing on-demand or automatic procedures
- and SMI codes, the Finite State Machine logics regulating the operational hierarchy.

Together with the local file system, each detector group can also use a network file system shared between all its nodes, where the final version of the developed files can be stored to be reachable from the central systems. This shared system has the advantage of being centrally maintained, backed up, and its availability is guaranteed by system administrators.

Using Microsoft Powershell scripts, the number of files developed by the experts in each group was evaluated, both in the local system and in the shared file system. Figure 2 shows that some of the more complex detectors are actually controlled with a limited amount of software, while some of the newer detectors have developed enormous amounts of software. The use of the shared system is also not optimal, despite the advantages it offers, like backup and high availability. As a result of these observations, we decided to take advantage of the upcoming winter shutdown for a more targeted information campaign, to enhance the deployment of custom software and improve the file system occupation.

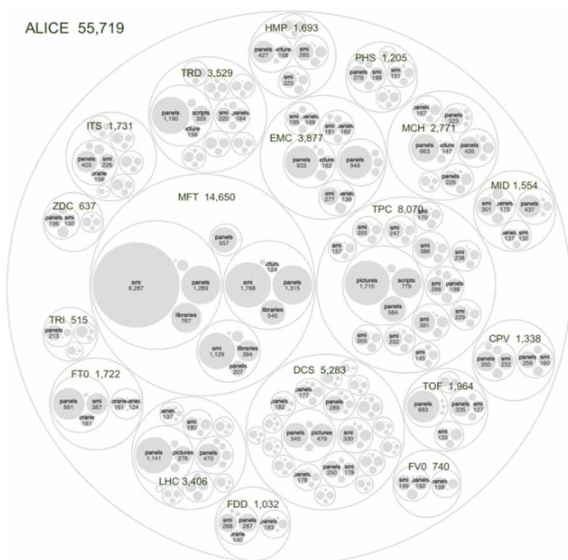

Figure 2: Number of custom files developed by detector experts.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP066

FRONTEND SOFTWARE

The WinCC framework offers a wide variety of drivers to connect to the most common commercial devices, but in HEP it is common to develop low level electronics with custom protocols. The introduction of the O^2 readout system has required the development of a new board, called Common Readout Unit (CRU), able to manage physics and condition data on optical lines, and interfaced by a new driver (ALF) and a frontend software called FRED [9].

FRED is developed and maintained by central DCS, and allows access to the CRU without the need for extensive software development. Detectors' customization is facilitated by the use of API and config files. Both central FRED and detectors' extensions are maintained in a common GIT repository site.

In order to check the extent of customization by detectors, a comparison was made between the lines of code developed for the FRED core and the detector branches available on the GIT site. Through a *python* script, we scrolled through the entire software repository, calculated the number of lines of code for the different file types, and collected the numbers in a *json* file. Using the D3.js graphics library [6] and CodeFlower [7] we realized a representation of the GIT site.

In Fig. 3 we have reported together the visualizations of the ALFRED core and the development for the single detectors. Some systems have developed just a minimal customization, profiting of the flexibility of configuration files (small red branches visible on the lower part of *fred*); while others have extensively developed code using the API (complex branches with many bubbles), or huge single files where HV settings are stored (the big red bubble).

From this picture we can understand the need for the different systems and assist in the optimization of further developments.

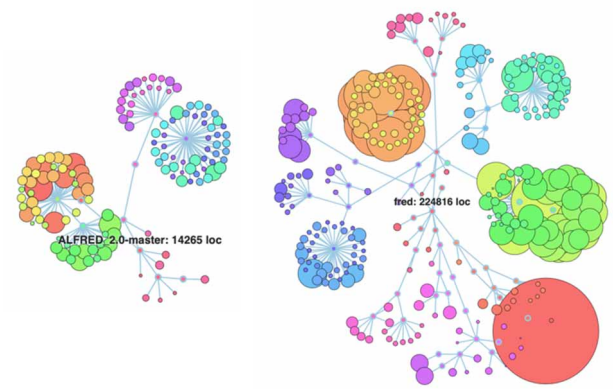

Figure 3: ALFRED core (left) and FRED detectors' repository (right). Bubble dimension represents the number of code lines.

THE FINITE STATE MACHINE

Hierarchical control of ALICE and its detectors is accomplished with the help of a Finite State Machine (FSM). Realized in SML/SMI++, this logical structure allows to simplify the operations through simple keywords representing the device states and the actions to be performed [10].

The ALICE FSM is the skeleton of DCS operations performed in the control room: through this hierarchy the DCS shifter can interact with detectors' parts and devices, even without a specific knowledge of the details.

To represent the full ALICE FSM in a graphical way, we have listed all nodes and branches in a *json* file, and used the D3.js library to visualize it in a flower-like way. In Fig. 4, we have more than 17000 nodes, 11500 of which are hardware devices (the leaves). Three systems represent 70% of the whole set, and their complexity is sometimes source of problems in the overall operation. By looking at this figure, and evaluating the structure of such complex branches, we were able to assist developers and address them to some optimizations.

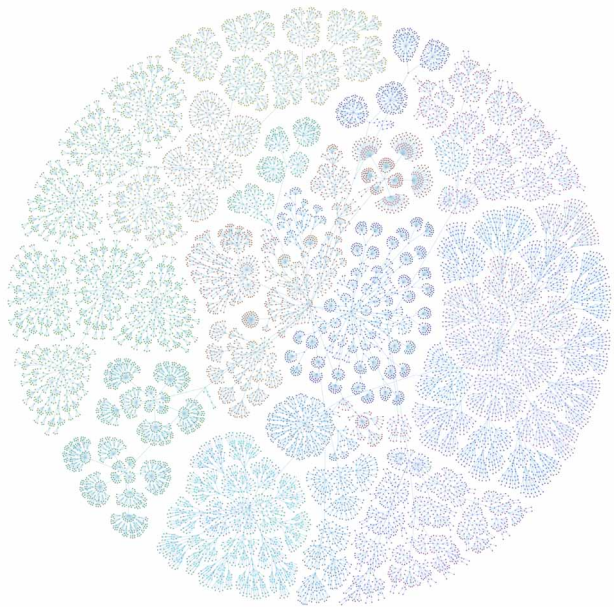

Figure 4: representation of the ALICE FSM in a flower structure.

DOCUMENTATION FOR OPERATORS

Another visualisation realised this time in *python*, is the analysis of the documentation made available by the experts for DCS shifters. Operators in control room supervise alarms and other events generated by detectors' nodes, and consult specific help files where they should find instructions on how to intervene on the faulty system.

To realise Fig. 5, we created a word cloud according to the occurrence of the words themselves. It is interesting to notice that the most common words refer to the intevention of experts ("call", "oncall", "expert", ..), more than giving instructions to be realized directly by the shifter.

In carrying out this word analysis, we also noticed the massive presence of expert names and telephone numbers, which often turned out to be obsolete.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THPDP066

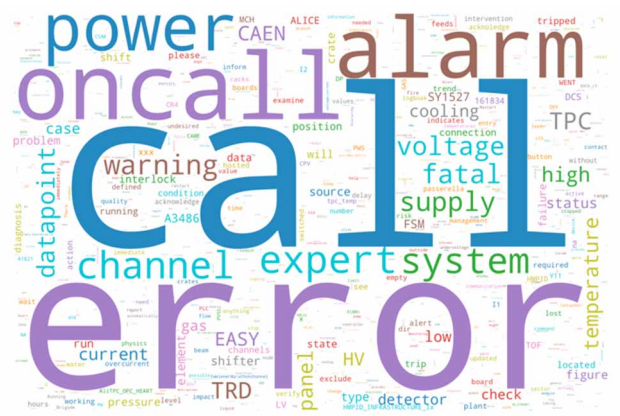

Figure 5: the most recurrent words in the shifter online documentation, accessible as alert help file.

CONCLUSION

This document describes the visualisations of various parameters related to the ALICE DCS. These are alternative techniques to the daily monitoring carried out by experts or developers. However, they can be useful to complete the knowledge of a large control system, to follow its evolution over time, and to evaluate targeted interventions when parts of the system become too large or diverge from the common line.

REFERENCES

[1] The ALICE Collaboration, "The ALICE experiment at the CERN LHC", *J. Instrum.*, vol. 3, p. S08002, Aug. 2008. doi:10.1088/1748-0221/3/08/S08002

[2] The ALICE Collaboration, "Upgrade of the Online-Offline computing system", CERN- LHCC-2015-006, ALICE-TDR-019, 2015.

[3] The ALICE Collaboration, "Technical Design Report of the Trigger, Data Acquisition, High-Level Trigger and Control System", CERN-LHCC-2003-062, 2003.

[4] SIMATIC WinCC Open Architecture, https://www.siemens.com/global/en/products/automation/industry-software/automation-software/scada/simatic-wincc-oa.html

[5] M. Lima, *The Book of Circles: Visualising Branches of Knowledge*, Princeton Architectural Press: 2017; *The Book of Trees: Visualising Spheres of Knowledge*, Princeton Architectural Press: 2014.

[6] D3, the Javascript library for bespoke data visualization, https://d3js.org

[7] CodeFlower, http://www.redotheweb.com/CodeFlower

[8] The ALICE Collaboration, "Letter of intent for ALICE 3: A next-generation heavy-ion experiment at the LHC", CERN, CERN-LHCC-2022-009, 2022. doi:10.48550/arXiv:2211.02491

[9] Milan Tkacik *et al.,* "FRED—Flexible Framework for Frontend Electronics Control in ALICE Experiment at CERN", *Processes*, vol. 8,p. 565, 2020. https://doi.org/10.3390/pr8050565

[10] Clara Gaspar, "Hierarchical Controls, https://lhcb-online.web.cern.ch/ecs/fw/FSMConfig.pdf