Software migration of the CMS ECAL Detector Control System during the CERN Large Hadron Collider Long Shutdown II

R. Jiménez Estupiñán[1,*], *L. Marchese*[1], *D. Di Calafiori*[1], *G. Dissertori*[1], *W. Lustermann*[1], *L. Djambazov*[1], *J. Fay*[2], *E. Auffray*[3], *D. Bailleux*[4], *D. Jovanovic*[5], *P. Adzic*[5], *P. Milenovic*[5]
on behalf of the CMS Collaboration

[1]ETH Zurich, Switzerland
[2]Univ Lyon, Univ Claude Bernard Lyon 1, CNRS/IN2P3, IP2I Lyon, France
[3]CERN, Geneva, Switzerland
[4]University of Notre Dame, Notre Dame, Indiana, USA
[5]University of Belgrade, Serbia

Abstract. During the second long shutdown (LS2) of the CERN Large Hadron Collider (LHC), the Detector Control System (DCS) of the Compact Muon Solenoid (CMS) Electromagnetic Calorimeter (ECAL) is undergoing a large software upgrade at various levels. The ECAL DCS supervisory system has been reviewed and extended to migrate the underlying software toolkits and platform technologies to the latest versions. The resulting software will run on top of a new computing infrastructure, using the WinCC Open Architecture (OA) version 3.16 and newly developed communication drivers for some of the hardware. The ECAL DCS has been configured and managed from a different control version system and stored with more modern encoding and file formats. A new set of development guidelines has been prepared for this purpose, including conventions and recommendations from the CMS Central DCS and CERN Joint Controls Project (JCOP) framework groups. The large list of modifications also motivated the revision and reorganization of the software architecture, which is needed to resolve and satisfy additional software dependencies. Many modifications also aimed to improve the installation process, anticipating in some cases works for the next long shutdown upgrade.

1 Introduction

The Electromagnetic Calorimeter [1] is one of the detectors of the CMS experiment, sitting at the CERN LHC. The DCS software of the ECAL detector monitors and controls a large

[*] R. Jiménez Estupiñán: jraul@phys.ethz.ch

number of equipment using the WinCC OA [2] platform. During the LHC long shutdowns, the architecture is typically reviewed and extended to include features that require extensive testing and long validation periods. The second long shutdown (LS2) has been extended by one year. This extension allowed us to implement large software updates, as described in the CMS ECAL DCS upgrade plan [3], as well as to carry out software migrations to bring the systems up to date, in line with the latest versions of the control platform and frameworks. We present in this document the main software updates, focusing on the migration of different software toolkits and platforms in terms of maintainability, compatibility and their impact in the existing architecture.

2 Software migrations

In parallel to the tasks foreseen for the LS2, the ECAL DCS team had to accommodate a large portion of time to migrate the existing architecture to run in newer platform versions. An initial offline migration is carried out, followed by a deployment in pre-production and a final switch between versions.

2.1 Migration of the version control system

CERN announced the discontinued support for the software versioning and revision control system known as Apache Subversion (SVN) [4]. The CMS collaboration has started a migration campaign to port the DCS source code from SVN to the Git distributed version control system [5], using the CERN GitLab hosting service. This type of migration has required not only the planning and reorganization of the repositories, but also the integration of the new toolkit and the migration of all historical records. After considering different options, it has been decided to break down the original single SVN repository into a hierarchy of individual Git repositories, organized according to their functional purpose.

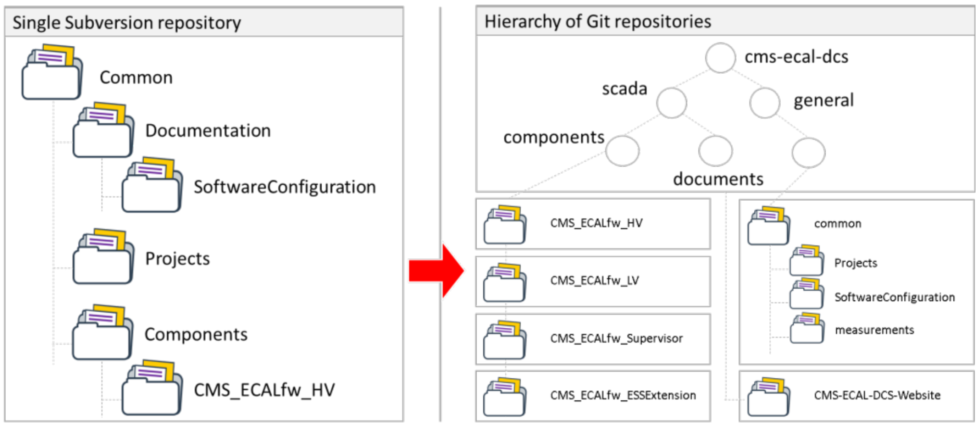

Fig. 1. Migration from a single Subversion code repository to a hierarchy of Git repositories.

The pre-existing repository was re-organized while transferring all the historical records from one platform to another. After a thorough analysis, we have passed from a single monolithic repository to a hierarchy of 35 different repositories, with 27 of them hosting individual software components. In addition to this, multiple applications with access to the repositories have been adapted to use the new toolkit command line interface to preserve the existing functionality.

2.2 Migration from WinCCOA version 3.15 to 3.16

The details for the WinCCOA migration were released during the first half of 2020. Among other changes, the different CMS detectors were requested to adapt their DCS software in order to make it compatible with the latest release of the control platform. On the contrary of previous migrations, the change of version from WinCCOA 3.15 to the version 3.16 has required a complex transition between encoding types and source file formats. Most of these changes aim to make the DCS software compatible with upper versions of the platform. Therefore, a proper planning and execution of the different migration steps are crucial to minimize the code-freezing transition period between versions.

2.2.1 Change of the character encoding from ASCII to Unicode

The version 3.16 of WinCCOA only supports the UTF-8 encoded projects. In lower versions, projects were created using the ISO-8859-1 (Latin1) encoding by default, not compatible with most of western European languages. By contrast to the UTF-8, ASCII characters are represented by single bytes and can be stored and treated within the code using a character data type. In the UTF-8 encoding, all symbols including all language special characters, can be treated using a dynamic multi-byte representation from 1 to 4 bytes per character. Only a small sub-set of the UTF-8 encoding is backwards compatible with the first half of the ASCII set of characters (codes from 0 to 127 are identically represented in both encoding formats). Even if not enforced by the JCOP framework [6] convention, the usage of only ASCII compatible characters, excluding the language support features, is still considered a good practice. The first step towards this migration was a consistency check to verify if the ECAL DCS code was compliant with this rule. All source files (libraries, scripts and panels) have been analysed to identify problematic pieces of code. As shown in the Table 1, we identified 11 out of 304 files containing non-ASCII compatible characters. Comments written by different developers including accents or special characters from languages different than English required straightforward changes.

Type of source file	Encoding formats	Number of files
Control libraries and scripts	ASCII	77
	UTF-8	0
	Unknown	2
User Interface (panels)	ASCII	216
	UTF	8
	Unknown	1

Table 1. Summary of files using different encoding formats.

Most of the ECAL DCS functionality is programmed using the native WinCC OA Control (CTRL) Language. The original standard library functions for string processing in CTRL language are byte-wise and remain unchanged between both versions for backward compatibility. A new set of Unicode standard functions has been tested and proven to be compatible also with ASCII character set. This compatibility feature has motivated the introduction of Unicode functions, whose deployment has been carried out even before the migration to 3.16 all across the ECAL DCS code. All components have been upgraded, tested and deployed in the legacy 3.15 systems. This strategy has allowed to install the same component versions in both the 3.15 and 3.16 versions, minimizing the code-freezing period since the development of features proceeded in parallel for both versions. In summary, the code has been refactored to introduce 11 new types of standard functions, making a total of 773 replacements across the ECAL DCS code.

Behind this analysis a more subtle problem related to the treatment of character arrays has been detected. Some applications require the treatment of strings as arrays of characters. There are pieces of code that process these arrays while converting and treating individual characters. Variables of character type and their position within the arrays are context dependent making the program likely to fail if not treated correctly. To identify these problematic pieces of code, we have created a small program to analyse the sources searching for the following language expressions: character type declarations, character literals, character arrays, strings as arrays, etc. Thanks to this approach, we have disclosed several important pieces of code that would have passed unnoticed and eventually failed.

2.2.2 User's interface reformatting into XML

The WinCCOA 3.16 version intends to adopt a new XML file format as the standard for storing the different graphical user interface file descriptors, commonly known as panels. While the previous file formats might still be compatible, it has been decided to explicitly convert all existing panels into the new type. A native tool has been used to convert all the panels. Since this XML format is also compatible with previous versions of WinCCOA, it has been decided to make an earlier deployment of the user interfaces in the production systems, permitting the identification of multiple rendering problems. After the initial conversion, the panels have been treated with a set of parsing tools to identify and correct invalid or redundant declarations.

2.3 Migration of the Control Framework from 8.1.2 to 8.4.0

One of the most important changes in the underlying software structures has been the migration to the version 8.4.0 of the CERN JCOP framework. In previous versions of this software framework, libraries and utilities were included as part of the global scope of WinCCOA. This means that most framework functions were accessible without declaring an explicit usage of their libraries. This feature was widely used across the CMS DCS, till the point that almost no source code file made an explicit inclusion of framework libraries. In the version 8.4.0 of the framework, it has been decided to discontinue this feature, mainly for performance purposes. This has forced the developers' community to update the source code to include framework libraries explicitly. However, this new paradigm has been proven to require a more programmatic approach because of the large number of code mentions and file sources.

We have used a combination of tools prepared by the JCOP and CMS Central DCS teams to identify and modify sources automatically. In more details, we have used a small utility called codePurger to construct the graph of function-calls across the ECAL DCS code repositories. This utility has been used to analyse and create a symbol's cross-reference database with all the framework function calls throughout the ECAL DCS code. As shown in Fig. 2, the symbol's database, later used by a java utility to look up for each framework function, has allowed the identification and modification of the sources that required an explicit library inclusion.

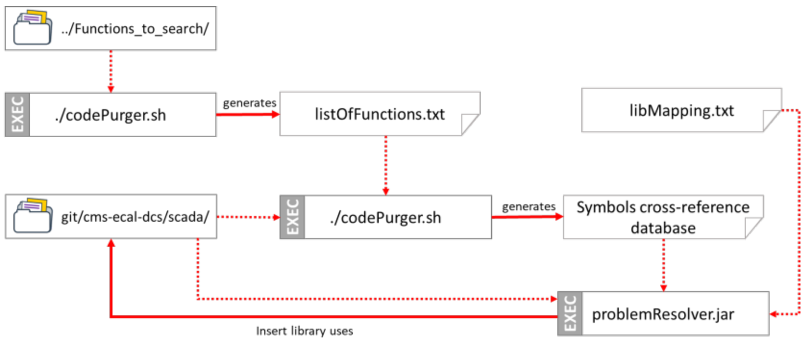

Fig. 2. Look up and automatic transformation process to include library uses.

The tools have processed a total of 120 different source files and performed 240 code transformations for purposes of library inclusions. After these automatic modifications, the code has been analysed and compared with previous versions for consistency. The result has proved the reliability of the automatic changes.

2.4 Migration to the OPC Unified Architecture

The OPC server is a fundamental software component that runs as part of the chain of components in a control system. The OPC Unified Architecture (UA) is the successor of the OPC Data Access (DA) that defines how real-time data can be transferred between a data source and a data client without having to know each other's native protocol. The ECAL DCS uses three types of OPC servers to monitor and control large collections of fieldbus devices:

1. Wiener OPC to access the Wiener Maraton power supplies feeding Low Voltage (LV) power to the ECAL Endcaps (EE) and Barrel (EB).
2. CAEN OPC to access the CAEN power supplies feeding power to the Preshowers and High Voltage (HV) power to the EE and EB.
3. CANOpen OPC to access the CERN made Embedded Local Monitor Boards (ELMB) for various applications.

The migration to the OPC UA involves different collections of hardware and software components. Hardware addresses are encoded as part of the data structures known as datapoints. Datapoints are usually software representations of physical devices. Every software component is responsible for installing and configuring datapoints that belong to a specific application domain. The components affected by this migration are the following:

Component name	**OPC Server type**	**Description**
CMS_ECAfw_PTHM	CanOpenOPC	Precision temperature monitoring
CMS_ECALfw_EE_HVM	CanOpenOPC	Endcaps HV monitoring
CMS_ESfw_HVM	CanOpenOPC	Preshowers HV monitoring
CMS_ECALfw_HV	CaenOPC	ECAL HV control
CMS_ESfw_FrontEnd	CaenOPC	Preshowers LV and HV control
CMS_ECALfw_LV	WienerOPC	ECAL LV control

Table 2. List of CMS ECAL DCS components affected by the migration of different OPC servers.

The first step towards the migration of the OPC has been the massive reconfiguration of the address space for every component. To do this, the latest releases of the JCOP framework includes a migration tool for every type of OPC server. The tool performs all the necessary transformations in the address space. In this new version, the OPC UA servers can be configured to run several instances, allowing for a better separation of concerns and moving from a monolithic single server approach into a multiple server configuration. This particular feature has allowed a further sub-division of the OPC server's configuration for each software component and the advancement of future modifications foreseen only during the major ECAL upgrade for the LS3 (e.g., the removal of the ECAL Endcaps and their corresponding LV and HV parts).

3 Architectural changes

The migrations described in the previous sections has an impact on the software architecture. The migration to a higher version of the JCOP framework has revealed differences in the resolution of component's dependencies at installation time. Also, the adoption of new libraries and utilities have motivated a revision and a subsequent improvement of several components, as the ECAL DCS installation toolkit.

3.1 ECAL DCS installation toolkit

The installation of DCS software components is one of the most critical operations of the software, directly affecting the availability and maintainability of the project. The unattended installation of components is the traditional method of deploying software in the CMS production environment. In the JCOP framework, each component is responsible for providing all the needed information for the component to work, executing instructions during the installation and satisfying subsequent component's dependencies. The ECAL DCS makes use of an advanced installation toolkit, which combines several years of knowledge and multiple features from the JCOP and the CMS installation toolkit. The ECAL PC configuration toolkit establishes a well-defined interface with a highly customizable instruction sequence to perform the installation of any ECAL DCS component. The ECAL installation toolkit has been modified to accommodate new features for redundancy support and the configuration of the new OPC UA servers. The new version of the toolkit is one of the cornerstones of the ECAL DCS migration. The toolkit version defines the compatible features, forcing the version staging of all dependent components.

3.2 Removal of the distributed connection between Tracker and ECAL DCS

In the WinCCOA environment, control systems can be configured as a network of autonomous distributed systems. The preferred topology for interconnecting machines in the CMS DCS is the spanning tree structure. The top nodes can control the lower layers, usually avoiding topology loops. The CMS DCS is a large distributed system formed by tens of control machines. The Central DCS nodes are on the top of the hierarchy and detectors are controlled from dedicated sub-trees. The Preshower and the Tracker pixel detectors share some parts of the cooling infrastructure. This means that operations on the shared infrastructure have an impact on both detectors and need to be monitored by both control systems. This motivated a distributed connection between them. Distributed connections have many technical implications beyond the pure network organization. However, a badly programmed application can create problems across the two distributed systems and disrupt each other operations. As a result of a shared effort, both DCS teams have worked to find an alternative solution. A well-defined interface between the Tracker and the ECAL DCS has been created using the CERN's Data Interchange Protocol (DIP) [7]. This interface has allowed the Tracker DCS to publish relevant Cooling information for the Preshower. As shown in Fig. 3, the ECAL DCS now subscribes to this data using a dedicated DIP interface, permitting us to remove the distributed connection between the systems.

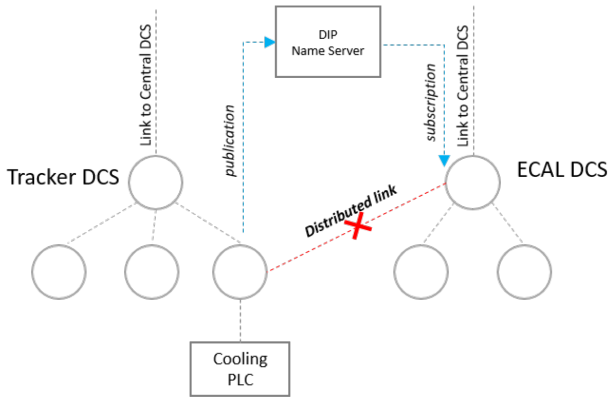

Fig. 3. Replacement of a distributed link by DIP publications, between the ECAL and Tracker DCS.

3.3 Reorganization of the notification system

The notification system is a crosswise functionality in the DCS. ECAL DCS components use a central tool for configuring alarm-based notifications, known as the CMSfwAlertSystem. This tool allows the system to import a recipient list from the CERN user's management interface known as e-groups [8]. E-groups can be used as a mailing list or to grant users access to different resources. Our team has participated in the extension of this tool by helping implement a further e-group integration.

In the previous schema, the tool required a periodic synchronization to operate. The new tool is capable of resolving real-time all recipients contact information (email addresses and GSM phone numbers). Components are typically responsible for adding or altering groups of notifications and to link them later to a recipient list. Originally, the ECAL DCS implemented a notification-centred mechanism, meaning that notifications and recipients were grouped into entities called notification groups whose recipients were synchronized once or twice a day. These notification groups were used by multiple components for sending messages to the same group of users, complicating the overall system maintenance. To solve this issue, the whole notification system has been re-implemented across 9 different components using the features mentioned above. The new notification system is now organized by application domains, each of them installed and used exclusively by a single component.

Thanks to the new e-groups integration features, components remain unaware of the recipient list while delegating its management to an external web-application. The notification domains are exposed as independent e-groups, to which users can subscribe or unsubscribe directly without altering the software.

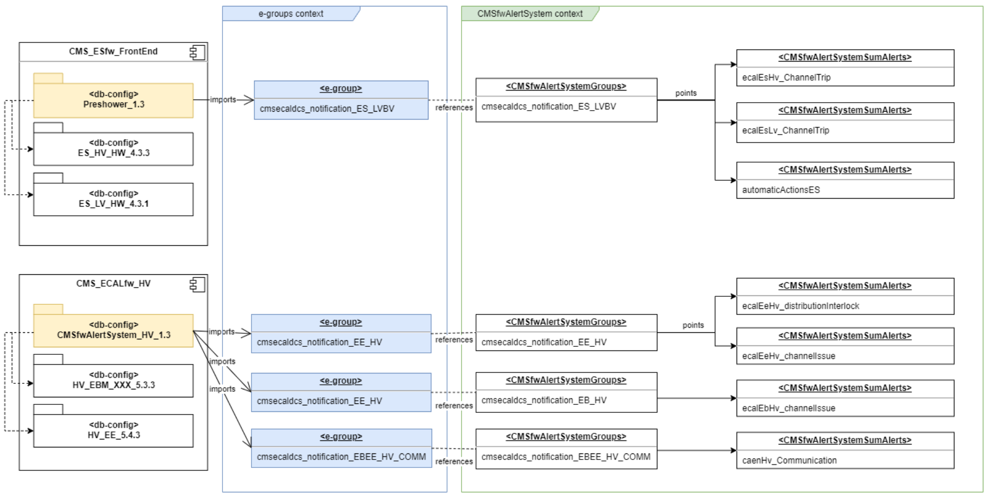
e-groups context
CMSfwAlertSystem context
CMS_ESfw_FrontEnd
<db-config> Preshower_1.3
<db-config> ES_HV_HW_4.3.3
<db-config> ES_LV_HW_4.3.1
imports
<e-group> cmsecaldcs_notification_ES_LVBV
references
<CMSfwAlertSystemGroups> cmsecaldcs_notification_ES_LVBV
points
<CMSfwAlertSystemSumAlerts> ecalEsHv_ChannelTrip
<CMSfwAlertSystemSumAlerts> ecalEsLv_ChannelTrip
<CMSfwAlertSystemSumAlerts> automaticActionsES
CMS_ECALfw_HV
<db-config> CMSfwAlertSystem_HV_1.3
<db-config> HV_EBM_XXX_5.3.3
<db-config> HV_EE_5.4.3
<e-group> cmsecaldcs_notification_EE_HV
<e-group> cmsecaldcs_notification_EE_HV
<e-group> cmsecaldcs_notification_EBEE_HV_COMM
<CMSfwAlertSystemGroups> cmsecaldcs_notification_EE_HV
<CMSfwAlertSystemGroups> cmsecaldcs_notification_EB_HV
<CMSfwAlertSystemGroups> cmsecaldcs_notification_EBEE_HV_COMM
<CMSfwAlertSystemSumAlerts> ecalEeHv_distributionInterlock
<CMSfwAlertSystemSumAlerts> ecalEeHv_channelIssue
<CMSfwAlertSystemSumAlerts> ecalEbHv_channelIssue
<CMSfwAlertSystemSumAlerts> caenHv_Communication

Fig. 4. Organization of the notification system for the ECAL HV and Preshower powering system.

4 Conclusion

During this long shutdown of the LHC, the ECAL DCS has been brought up to date to use some of the latest versions of the control platform, software frameworks, drivers and utilities. All code sources have been reviewed and thoroughly modified to accommodate to the new context. All code sources and their history along the past 12 years have been ported to the CERN Gitlab instance, a more modern and powerful version control system. The different code transformations imposed by the WinCCOA and JCOP migrations have been planned and organized to minimize the code-freezing period during the transition between versions. Decisions such as maintaining the ASCII backwards compatibility or the explicit inclusion of libraries have allowed partial deployments in the legacy systems, while continuing developing other features. The architecture has been reviewed to correct, adjust and improve relevant pieces of code. In many cases, these modifications have permitted to advance work and prepare the software components in view of the major ECAL upgrade for the next long shutdown, for instance the breakdown of the OPC server's configuration according to the physical organization of the detector. All these modifications have been validated using automatic deployment tools and automatic testing techniques, based on a set of powerful virtual machines. The new versions of the ECAL DCS software have been successfully deployed in a pre-production environment, running on the next generation of computing servers and waiting for the final validation.

The authors would like to thank the Swiss National Science Foundation for the financial support, as well as the CMS Tracker DCS team for their participation in the works described in this paper.

References

1. "The CMS electromagnetic calorimeter project: Technical Design Report", CERN-LHCC-97-033 CMS-TDR-4, CERN, Geneva, 1997.
2. SIMATIC WinCCOA, http://www.etm.at/index_e.asp

3. R. Jiménez Estupiñán et al., "CMS ECAL Detector Control System upgrade plan for the CERN Large Hadron Collider Long Shutdown II", in *Proc. 12th Int. Workshop on Emerging Technologies and Scientific Facilities Controls (PCaPAC'18)*, Hsinchu, Taiwan, Oct. 2018.
4. Apache Subversion, http://subversion.apache.org/
5. GIT Distributed Version Control System, http://git-scm.com/
6. JCOP framework, http://jcop.web.cern.ch/jcop-framework
7. W. Salter et al., "*DIP Description*" LDIWG (2004), https://cern.ch/dip/
8. CERN e-groups, https://aisdocs.web.cern.ch/display/ED/1.+Introduction