ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA111

EASING THE CONTROL SYSTEM APPLICATION DEVELOPMENT FOR CMS DETECTOR CONTROL SYSTEM WITH AUTOMATIC PRODUCTION ENVIRONMENT REPRODUCTION

I. Papakrivopoulos[1,2], G. Bakas[1], U. Behrens[3], J. Branson[4], P. Brummer[2,5], S. Cittolin[4], D. Da Silva Gomes[2,6], G. L. Darlea[7], C. Deldicque[2], M. Dobson[2], N. Doualot[2,6], J.R. Fulcher[2], D. Gigi[2], M.S. Gladki[2], F. Glege[2], G. Gomez-Ceballos[7], J. Hegeman[2], W. Li[3], A. Mecionis[6,8], F. Meijers[2], E. Meschi[2], R.K. Mommsen[6], K. Mor[2], S. Morovic[4], V. O'Dell[6], L. Orsini[2], C. Paus[7], A. Petrucci[4], M. Pieri[4], D. Rabady[2], K. Raychino[2], A. Racz[2], A. Rodriguez Garcia[2], H. Sakulin[2], C. Schwick[2], D. Simelevicius[2,8], P. Soursos[2], A. Stahl[3], M. Stankevicius[6,8], U. Suthakar[2], G. Tsipolitis[1], C. Vazquez Velez[2], A. Zahid[2], P. Zejdl[2,6]
[1]National Technical University of Athens, Athens, Greece
[2]CERN, Geneva, Switzerland
[3]Rice University, Houston, USA
[4]University of California San Diego, San Diego, USA
[5]Karlsruhe Institute of Technology, Karlsruhe, Germany
[6]Fermi National Accelerator Laboratory, Batavia, USA
[7]Massachusetts Institute of Technology, Cambridge, USA
[8]Vilnius University, Vilnius, Lithuania

Abstract

The Detector Control System (DCS) [1], [2] is one of the main pieces involved in the operation of the Compact Muon Solenoid (CMS) experiment at the LHC. The system is built using WinCC Open Architecture (WinCC OA) and the Joint Controls Project (JCOP) framework [3] which was developed on top of WinCC at CERN. Following the JCOP paradigm, CMS has developed its own framework which is structured as a collection of more than 200 individual installable components each providing a different feature. Every one of the systems that CMS DCS consists of, is created by targeting and installing a different set of these components. By automating this process, we are able to quickly and efficiently recreate systems both in production, but also, to create development environments identical to the production ones. This latter one results in smoother development and integration processes, as the new/reworked components are developed and tested in production-like environments. Moreover, it allows the central DCS support team to easily reproduce systems that the users/developers report as being problematic, reducing the response time for bug fixing and improving the support quality.

INTRODUCTION

Control and real-time monitoring are essential for the successful operation, wellbeing assurance and efficient data taking of the CMS detector. This is where the DCS comes in play. Similar systems are used in all the LHC experiments but the design, structure and implementation differ between them. In CMS the DCS is implemented as a big distributed system [4] where each node, which is called a system or a project, plays a distinct role either providing general infrastructure or having a dedicated role interfacing and interacting with a single subsystem of the experiment.

The DCS community in CMS consists of a central team that is responsible for the control system that handles all the common and generic infrastructure of the experiment and a set of sub-detector teams, each of them being responsible for one or more systems that handle the specific needs of every individual subsystem of the experiment. Apart from being in charge of the general infrastructure, the central team provides support to the rest of the experiment in control system related topics as well as tools that can be reused by the other teams or implement common experiment-wide functionality. Finally it provides administration and operation of all the control system related server infrastructure of the experiment in terms of OS and generic software like WinCC, OPC servers etc.

The experiment's control system is designed in a way of small reusable software entities that are called components. As mentioned above, each system and as an extension its role, is defined by the set of components that are installed into it. All systems start from a minimal initial point differing only by a little and diverge from one another with the installation of the components taking their final state and altogether forming the control system of the experiment. Out of the 200 individual components around half are provided by the central team while the rest are system specific and are developed by the sub-detector teams. Most of the central components, offer generic functionality that can be used and even extended while others implement a specific feature, like the communication with a certain type of device. This modular architecture allows for better maintenance of the system, but at the same time makes it highly dependent on the ability to constantly be able to install components. Furthermore, it renders CMS independent of the specific

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA111

state of a project at a certain point. This way, the system as a whole or any of its parts can be re-created from scratch whenever needed e.g. in case of hardware failure, offering high availability.

The sub-detector systems are created and maintained by the central team but they have installed software that is developed by the sub-detector teams. It is important that the software delivered by the various teams is tested and its installation is uneventful. Moreover in case of updates and improvements in the control system software, the corresponding components are reinstalled in order for the new feature to become part of the system. One of the steps in the software development lifecycle is the creation of projects and the installation of newly developed features or components to these projects.

From all the above it is evident that project installation and recreation is a key tool on the design of the DCS in CMS. It is of the outmost importance that the system can be re-installed from scratch at any given point. This means that for every change made in any of the components the installation procedure needs to be replicated in order to make sure that the newly developed features are installed without a problem and that they do not interfere with existing code.

SYSTEM CREATION AND ADMINISTRATION

A solution that plays an important role in providing this functionality is the fwConfigurationDBSystemInformation tool provided by the JCOP framework team. It consists of a schema for an oracle database, a WinCC script and a User Interface (UI). The schema which is responsible for holding the data is organized in such a way that allows the association of the data in the various tables. The user can register hosts, projects as well as specific information about them and components. Then tree structures can be created by assigning projects to hosts and components to projects. The UI is used as the interface between the user and the database in order to view and alter its contents and of course create tree structures like the ones mentioned above. The script is the interface between each project and the database, enabling data flow from the project to the database and the other way around.

The logic behind the tool is that an instance of the script runs in every project and updates the database with the project's information every time a user-defined interval has elapsed. Since the user has stored the project's information, any discrepancy between the project and the database can be identified and reported. Additionally the project can store its local component related information. This consists of which components and when they were installed and if the operation was carried out successfully again allowing for discrepancy reporting and fixing.

This tool is used extensively by the central DCS team of CMS for monitoring and administering all the control system projects. This tool can operate in two modes with the one specified above being the one called "local". This means that the database is used as a means for the project to store its information and the actual project is the master. Whatever happens in the project manually by the user is stored in the database. A second mode exists which is called "central". In this mode, the database is used as a reference point and all stored information in it is passed to the project. In both modes the aforementioned script is used as the link between the project and the database, storing information in the database in the local mode and changing the project settings in the central one. In the central mode, apart from generic project information, components are also installed and deleted based on the database contents. In fact this is the mode used in CMS for administering the production machines. Moreover, since apart from the central team the rest of the community involved in the control systems does not have direct access to the control system machines, through this tool they are allowed to interact with their systems. An interface is provided to them through which they can modify the contents of the database and thus install or re-install components in the production systems.

An extension to this functionality was created in CMS. A whole suite of batch files was developed in order to use the database stored information to also create projects, not just administer them. That is how the production systems in CMS are created whenever this is needed. Either during a normal installation, e.g. migration to new hardware or new version of OS etc. or in case of recovering from failure. This greatly eases the creation of projects as it can be done with the execution of a single script.

PRODUCTION SYSTEM REPRODUCTION

As it has already been stated, the need for system creation is constant and could appear at any time, either in production or in a development system. It was decided to extend the tool and make its use possible also during the development process. Though this would mean that whenever information is changed in the database the production systems may be affected. To avoid such scenarios, a set of changes needed to be implemented in order to avoid interference between the data stored in the database for the production systems and the ones stored for the development systems. This would allow the team to give access to the tool also to new members that are not experienced and would otherwise not be granted access in order to not interfere with the production systems.

The chosen solution was the following. Two databases where used instead of one, connected to each other with a database link, and a set of three schemas. The first schema is the production one that already existed. Let's call it schemaA for the shake of this example. So schemaA exists in databaseA. In a second database, let's call it databaseB a second schema was created that was named schemaB. This would hold all the development environment data. A third schema was created, schemaA, in databaseB which would be empty without any tables but contains only views and triggers for redirecting the data to the correct schema. For every table in the normal schema that comes with the

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA111

tool a view was created in schemaA in databaseB. The view was constructed in such a way that it queries both other schemas, schemaA in databaseA and schemaB in databaseB. This way the tool can see all entries, both the production and development ones. For every action that can be done in the normal schema tables i.e. INSERT, ALERT, DELETE a trigger was created in schemaA in databaseB that would redirect all traffic to schemaB in databaseB. This logic is displayed in Fig. 1. Following this approach, all the production data such as project and host info and all the registered components are available for use in the development environment.

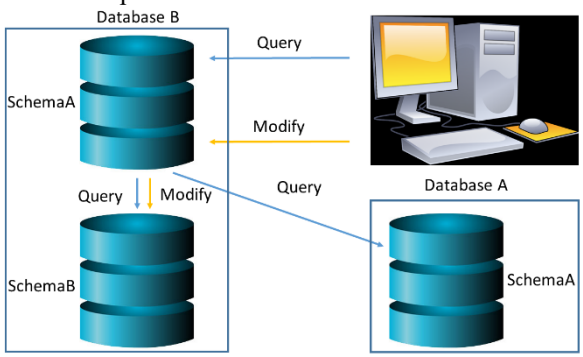

Figure 1: Schematic representation of the database configuration used.

At the same time it was decided to extend the tool so that it would be possible to copy structures. For example, as we mentioned previously, during development it is crucial to install new or revised components in an environment that simulates what is used in the production control system. Copying structures in the database would allow us to quickly instruct the tool to create a copy of a production project in one of the development machines. To do so, the production data would have to stay hidden for modification but at the same time be visible because they would have to be copied. Using the copy structure functionality in the database and then the batch file utilities, a production environment could be recreated as quickly as possible.

EXTENDING COMPONENT TARGETING

One of the main features of the original tool is bookkeeping for the JCOP components. It provides a native way of storing the list of components that are installed in the project as well as the list of components that should be installed. Components are organised using groups. The user defines a group that contains a set of components, and targeting it to a project automatically assigns all the components of this group to the project. This allows for easier targeting of multiple components and dependency handling as we can ensure that bunches of components are targeted at the same time.

A limitation on this approach was spotted on CMS in the case of major software updates. An example of this would be a new release of the JCOP framework where all its components have to be reinstalled. At the same time, all the CMS specific ones that are affected by the changes in the framework need to also be modified and reinstalled. In cases like this, new groups have to be created and populated with the corresponding components. This is because groups do not support multiple versions of the same component as this could lead to inconsistencies, because two versions of the same components would be targeted at the same time. In addition to this, the installation scripts running in each project have to be stopped because inconsistencies would be detected between the project and the database. Also, since the projects in CMS are operated in central mode, not listed components would be deleted causing problems to the system.

A new approach was implemented, keeping the old design of groups but extending it in order to overcome the above limitations. The solution was to use a “tag” that is assigned to each component in the group and to the project. This way, an extra constraint is used in order to associate components to projects. Instead of automatically assigning each component of a group to a project, now only the components that are inside the group and have the same tag as the project are assigned to it. An eventual upgrade will then follow these steps: the group is modified by inserting the new components with an updated tag; at the same time the project’s tag is updated to match the one of the new components; then everything would be automatically targeted without the need to un-target things or stop scripts.

BROADENING THE USERBASE

As discussed above, the installation process is very important in CMS. This is not only true for the central DCS team that is responsible for maintaining the production systems and making sure that they operate non-stop but also for the members of the development community in CMS that create control system related software and are not in the central team. They have to make sure that what they develop is in an operational state and can safely be installed in the production systems. To ensure this, they have to be able to simulate the installation process, which means testing it in conditions that are as close as possible to the production ones. This need became even more important during the long shutdown 2 (LS2) – taking place in 2019 and 2020 – as CMS is replacing all the DCS servers and in parallel is following the general CERN update of WinCC to its newest version. To accompany the changes in WinCC version the JCOP team will also release a new version of its framework that will be compatible with the new WinCC version.

This combination of upgrades will require a lot of testing and system recreation during the development period from the whole CMS community, and of course, in the end, a total system upgrade all of it with minimal or - if possible - no downtime. The tool mentioned above with all the upgrades done would be essential throughout this procedure, saving a lot of time on the process. To ensure that each of the teams work fully focused on their own space and that no distraction or interference occurs between the various groups in CMS, it was decided by the central team to include the idea of ownership in the objects in the database, more specifically in computers, projects,

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA111

and groups of components. Therefore the workflow of each team is optimised, because their work environments are isolated and easily reproducible. This will play a key role in the process of the upcoming upgrades in the CMS infrastructure, both hardware and software, ensuring a faster development lifecycle and a better testing environment, minimising the possibility of undesired and unpredictable behaviour during software deployment.

CONCLUSION

The creation procedure of new projects is a part of the everyday work on the maintenance and improvement of the control system in CMS, either because a system has to be recreated to reproduce a potential problem, or a new feature has to be tested in an environment that simulates the production one. This is also true in cases of major software upgrades or on hardware failure and upgrades. The extensions of the fwConfigurationDBSystemInformation tool presented in this paper have highly enhanced the level of support provided by the central team, both in cases of every day support and debugging as well as in the case of system upgrades. Furthermore, making the tool available to the whole CMS community has reduced the development difficulties and the time consumed by the members of the subsystem communities during the testing period of development. The tool is expected to play an important role in the upcoming hardware and software migrations in CMS DCS in the next couple of years, as well as in the upcoming ones as the system is continuously evolving with time.

REFERENCES

[1] R. Gomez-Reino *et al.*, "The Compace Muon Solenoid Detector Control System", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'09)*, Kobe, Japan, Oct. 2009, paper MOB005, pp. 10-12.

[2] G. Bauer *et al.*, "Status of the CMS Detector Control System", *J. of Phys.*, Conference Series, vol. 396, Part 1, 2012.

[3] O. Holme, M. Gonzalez-Berges, P. Golonka and S. Schmeling, "The JCOP Framework", in *Proc. 10th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'05)*, Geneva, Switzerland, Oct. 2005, paper O3_005.

[4] R. Arcidiacono *et al.*, "CMS DCS Design Concepts", in *Proc. 10th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'05)*, Geneva, Switzerland, Oct. 2005, paper P1_062.