AN OVERVIEW OF THE LHC EXPERIMENTS' CONTROL SYSTEMS

C. Gaspar, CERN, Geneva, Switzerland

Abstract

The four LHC experiments (ALICE, ATLAS, CMS and LHCb), either by need or by choice have defined different requirements, use different equipment, and are operated differently. This led to the development of four quite different Control Systems.

Although a joint effort was done in the area of Detector Control Systems (DCS) allowing a common choice of components and tools and achieving the development of a common DCS Framework for the four experiments, nothing was done in common in the areas of Data Acquisition or Trigger Control (normally called Run Control).

This paper will present an overview of the design principles, architectures and technologies chosen by the four experiments in order to perform the main tasks of the Control System: Configuration, Control, Monitoring, Error Recovery, User Interfacing, Automation, etc.

INTRODUCTION

In general the Control System of an LHC experiment handles the configuration, monitoring and operation of all experimental equipment involved in the different activities of the experiment:

- The Data Acquisition System (DAQ): front-end electronics, readout network, storage etc.
- The Timing System: timing and trigger distribution electronics
- The Trigger: the hardware trigger components.
- The High Level Trigger (HLT) Farm: thousands of trigger algorithms running on a CPU farm.
- The DCS: sub-detector gases, high voltages, low voltages, temperatures, etc. and also experiment's infrastructure: magnet(s), cooling, electricity distribution, detector safety, etc.
- Interaction with the outside world: LHC Accelerator, CERN safety system, CERN technical services, etc.

The relationship between the Control System and other components of the experiment is shown schematically in Fig. 1. This figure shows that the Control System provides a unique interface between the users and all experimental equipment.

Some of the requirements that were common to the four experiments are:

- Distribution and Parallelism - Due to the large number of devices and IO channels, the acquisition and monitoring of the data has to be done in parallel and distributed over many machines.
- Hierarchical Control – The data gathered by the different machines has to be summarized in order to present a simplified but coherent view to the users
- Partitioning – Due to the large number of different teams involved and the various operation modes of the system, the capability of operating parts of the system independently and concurrently is mandatory.
- Automation – Standard operations and error recovery procedures should be, as much as possible, automated in order to prevent human mistakes and to speed up standard procedures.
- Intuitive User Interfaces – Since the operators are not control system experts it is important that the user interfaces are intuitive and easy to use.
- All other standard requirements in large Control Systems: Scalability, Reliability, Maintainability, etc.

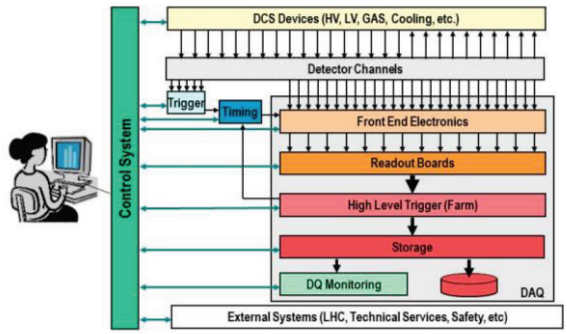

Figure 1. Scope of the Experiment Control System.

LHC EXPERIMENTS' COMMONALITIES

The Joint Controls Project

Around the end of 1997, a common project between the four LHC experiments and a CERN controls group (first IT/CO then EN/ICE) was setup. Its mandate was to: *"Provide a common DCS for all 4 experiments in a resource effective manner"* and in more detail to: *"Define, select and/or implement as appropriate the architecture, framework and components required to build the control system"*.

This project – JCOP (Joint COntrols Project) [1] – was very successful and it is often cited as an example; it resulted in a common architecture and a common framework used by all 4 experiments and all their sub-detectors and sub-systems (and also by other experiments and projects at CERN).

JCOP is still active promoting commonality and proposing and implementing common developments and upgrades in the area of Detector Control Systems.

Throughout the years, JCOP has spawned many important sub projects, for example:

- The Architecture Working group
- Technology Survey: evaluation, validation and selection of products for use by the Control System
- The Framework Working Group
- The Detector Safety System
- And several others

ISBN 978-3-95450-139-7

The JCOP Framework

The major outcome of the above working groups was the JCOP Framework (FW) [2]. It was defined as:

"An integrated set of guidelines and software tools used by Detector Developers to realize their specific Control System application. The Framework will include, as far as possible all templates, standard elements and functions required to achieve a homogeneous Control System and to reduce the development effort as much as possible for the Developers".

At the end of a very detailed evaluation process a product was selected as the basis for the framework. This product is a SCADA (Supervisory Control and Data Acquisition) System - PVSS II (now called WinCC-OA) [3], from ETM (now part of Siemens). PVSS II has a highly distributed architecture allowing the Control System to run distributed over hundreds of PCs. It provides a common interface to access all necessary equipment and it provides several tools to ease the life of building a control system, some of them are:

- Several drivers to access various types of devices (extendable to allow for user-devices)
- A run-time database for storing the data coming from the various devices, easily accessible for processing, visualisation, etc.
- Alarm Handling (generation, filtering, masking, visualization of alarms)
- Data Archiving, Logging, Scripting, Trending, etc.
- A very powerful User Interface Builder
- Several predefined Interfaces: Parameterisation, Alarm Display, Access Control, etc.

Within the Framework and in order to handle high level abstraction, PVSS II was complemented by another tool: SMI++ (State Management Interface) [4]. SMI++ is a toolkit for modelling the behaviour of large distributed control systems; its methodology combines three concepts: object orientation, Finite State Machines (FSM) and rule-based reasoning.

By combining these two products the framework offers tools to implement a hierarchical control system, in particular using a graphical user interface, shown in Fig. 2, which allows the configuration of object types, declaration of states, actions, rules, etc. as well as the definition and operation of the hierarchical control tree.

The framework was then complemented by numerous components to completely handle the most common types of equipment (very often using the OPC - OLE for process Control protocol) [5]; another communication protocol –DIM (Distributed Information Management) [6], to access any non-standard devices; access to a Configuration Database; System Overview tools to monitor the state of the control system itself; etc.

The JCOP framework was used to completely design and build the DCS of ALICE, ATLAS and CMS and in LHCb it was used across the whole experiment to implement the Experiment Control System and all its sub-systems. It is also used by other experiments at CERN like COMPASS or NA62 and/or other Common projects like the LHC experiment's Gas Systems, the Detector Safety Systems, etc.

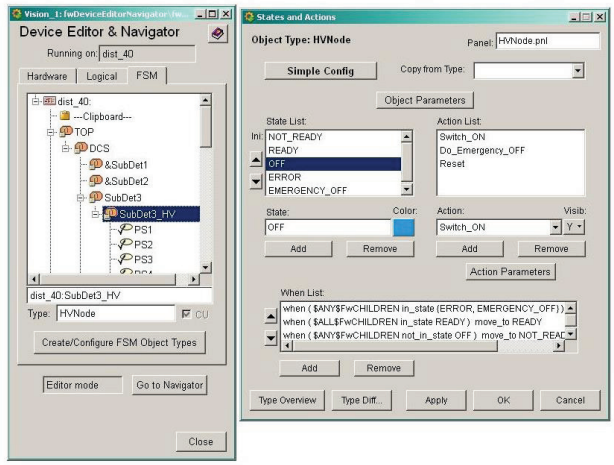

Figure 2: The Framework Device Editor Navigator.

LHC EXPERIMENTS' DIFFERENCES

At the time of starting the Control System's design it was considered that the DAQ and Trigger Control areas did not offer enough commonality to try to work together among the experiments. So each experiment set up a Control team to work in this area and in particular, to design and implement their Run Control (the highest level interface to the control of the experiment). Not surprisingly each experiment arrived at a completely different solution. In fact since the original requirements were quite similar, the architectural choices and even the list of components are actually very similar in the four experiments. What is quite different is the emphasis given to different design principles and above all the choice of tools, products or paradigms used to implement each one of the components.

Design Principles

Different experiments have followed different design principles and put emphasis on different requirements. In ATLAS the system was designed to be hierarchical and provide a high level of abstraction, in CMS the first design choice was to have a web-based system, while ALICE tried to design highly customisable and flexible components and LHCb put the highest emphasis on having an integrated and homogeneous control system.

Architecture & Scope

The high-level architectures of the four experiments' control systems are very similar. All are divided into a DCS tree encompassing the various sub-detectors and sub-systems and a Run Control tree overseeing the electronics of the various sub-detectors and the central DAQ & Trigger systems. Fig. 3 shows as example the ATLAS architecture, CMS's architecture is quite similar. In ALICE and LHCb there is an "Experiment Control System" (ECS) above all other central systems (DCS, DAQ, Trigger and HLT), either directly (ALICE) or

ISBN 978-3-95450-139-7

through a Run Control level (as shown in Fig. 4 for LHCb).

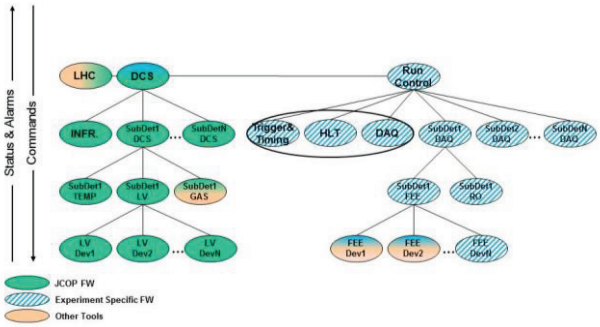

Figure 3: ATLAS Control Architecture.

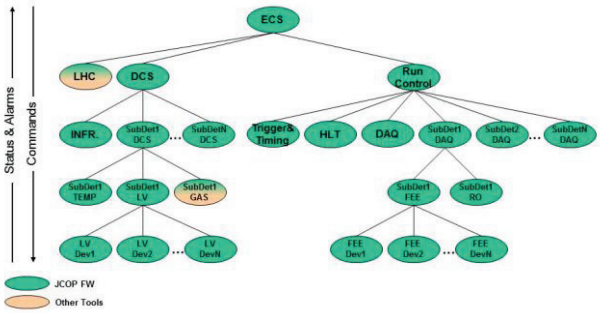

Figure 4: LHCb Control Architecture.

CONTROL SYSTEM COMPONENTS

Several frameworks are in use in the LHC experiments: apart from the JCOP framework there are experiment specific frameworks in three out of four experiments (as in Fig. 3). These have different names and slightly different functionality in each experiment:

- ALICE: uses DATE (Data Acquisition and Test Environment) [7].
- ATLAS: Provides a set of services for higher level control [8] and a DAQ framework: the Rod Crate DAQ Framework.
- CMS: has two complementary frameworks: RCMS (Run Control and Monitoring System) and XDAQ (DAQ Software Framework)[9].

Even though these Frameworks provide quite different tools and components, some functions are present across all Frameworks as they are basic components of any Control System:

- Communications: For acquiring data, sending commands and in general for exchanging messages between processes.
- Finite State Machines: For the description of system components and for the synchronization and sequencing of operations
- Expert System Functionality: For error recovery, operator assistance and/or system automation
- Databases: For storing configuration data, for archiving historical data, etc.
- User Interfaces: For Visualization and for System operation
- Many other services like process management, resource management, etc.

Communications

All Frameworks provide communication mechanisms. Within communications we can distinguish three types of data flow:

- "Control" data: These are normally short messages, bidirectional traffic mostly commands in one direction and status messages in the other direction.
- "Configuration": These messages can contain large amounts of data mostly in the direction control-system to hardware (or software process).
- "Monitoring": These can also be large messages normally in the opposite direction .i.e. hardware (or software process) to Control System. Furthermore Monitoring data may need to be "archived" or made persistent either for short periods or even permanently, so that the system status can be analysed or trouble-shot in case of problems.

The various experiments have different ways of handling these different types of data:

- JCOP FW/LHCb: Within the JCOP Framework most control type data is handled by the SMI++ toolkit which uses the DIM Publish/Subscribe mechanism. The configuration data is handled by PVSS and it is sent to specific devices (hardware or software) using the appropriate drivers, in LHCb, for example, the largest amounts of Configuration and Monitoring data are in the DAQ area and are sent/received via DIM.
- ALICE ECS also uses SMI++ (but outside the JCOP/PVSS framework) and hence DIM for control messages. DIM is also used directly for some Configuration and Monitoring but ALICE has the particularity that most sub-detector DAQ electronics are configured via the DCS hence via the JCOP FW (and in most cases via DIM).
- ATLAS uses CORBA [10] for all Communications, within two packages: "IPC" (Inter Process Communications) for Control and Configuration and "IS" (Information Service) for Monitoring. In ATLAS some sub-detector electronics are also configured via the DCS (JCOP FW)
- CMS uses Web Services [11]. These are used within the RCMS high level framework for Control, within the XDAQ framework for Configuration and within XMAS (the XDAQ Monitoring and Alarm System) for Monitoring.

Within the DCSs and in LHCb, PVSS II (temporarily) and its archive mechanism in Oracle (permanently) is used as a repository for monitoring data. This is also the case of the ATLAS IS (although only transiently) and the CMS XMAS system.

The three most used communication mechanisms in the DAQ area are DIM, CORBA in the ATLAS IPC and the CMS Web Services (XML/Soap), they all use the Client/Server model and mostly a Publish/Subscribe

ISBN 978-3-95450-139-7

mechanism. It is difficult to compare them in terms of performance, but DIM is a thin layer on top of TCP/IP, IPC is a thin layer on top of CORBA, both provide a simple API, a Naming Service and some error-detection and recovery. As advantages they are both quite efficient and easy to use. As drawbacks, DIM is home-made while CORBA is not so popular anymore. As for Web-Services they are a standard, modern, protocol but their performance can be a drawback due to the XML overhead. In the DCS area OPC DA (Data Access), is widely used. It is an industry standard but its main drawback is the link to Windows, this will be overcome in the new platform independent standard: OPC UA (Unified Architecture).

Finite State Machines

All experiments use Finite State Machines in order to model the system behaviour. The SMI++ toolkit is the most used since it is an inherent part of the JCOP FW and hence used in all DCS Systems, for the complete modelling of LHCb's Experiment Control System and also used in ALICE as a stand-alone tool. ATLAS has had several iterations of their FSM toolkit, the first version used CHSM (Concurrent Hierarchical State Machines) which used its own statechart specification language, the second version used CLIPS [12] (a tool for building expert systems) while the current version is home-made in C++. CMS built two FSM toolkits, one in Java for RCMS and one in C++ for XDAQ. The approach of each experiment to how to design, implement or distribute the FSMs for the various sub-systems is also different:

- In ALICE the FSM for all sub-systems was provided centrally but can be different from one sub-system to another.
- In ATLAS the FSM for all sub-systems was provided centrally and they all have to be the same.
- In CMS FSM templates were provided centrally, sub-systems implement specific Java or C++ code.
- In LHCb FSM templates were provided centrally, sub-systems can modify the template using a graphic editor.

In general most experiments decided on a few, coarse-grained states to model their Run Control operations. Assuming that most sub-systems can work in parallel, generic actions can be sent down from the top and the top-level needs no or very little knowledge of the sub-systems' internals.

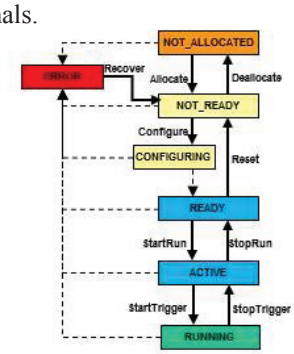

Figure 5: LHCb Run Control FSM.

Fig. 5 illustrates as an example the LHCb Top-level Run Control FSM. ATLAS and CMS FSMs are quite similar.

In ALICE the top-level needs to synchronize more detailed operations across sub-systems so the top-level FSM needs more states, around 20 to 25, 15 states from "ground" state to "RUNNING".

Expert System Functionality

All experiments saw the need for some form of expert system functionality. The approach is normally: "we are in the mess, how do we get out of it?" by opposition to "we are in the mess, how did we get there?" and none of the systems has the capability of "automatic learning", in all cases all "rules" are coded by experts. Expert systems are used for advising the shifter (in ATLAS and CMS), automated error recovery (ATLAS, CMS, LHCb and more modestly in ALICE) and to completely automate standard operations (LHCb). The tools used are:

- In ATLAS: CLIPS is used for error recovery. There are central and distributed (domain specific) rules. The system is used only by CLIPS experts, they can implement sub-system rules on request. A different tool is used for the "Shifter Assistant". This is based on "Esper" [13], a component for Complex Event Processing. Esper allows dealing with large volumes of high-frequency time-based event data. ATLAS is now moving away from CLIPS and more towards the Esper approach.
- The CMS RCMS framework provides expert-system functionality implemented in Java: asynchronous notifications can be received from sub-systems allowing each node to automatically handle problems. A separate, complementary tool for shifter assistance: the "DAQ Doctor", uses the Perl scripting language.
- In LHCb and in the experiments' DCSs SMI++ is used. Since the tool is quite simple to use (due to its graphic PVSS II interface), it is used directly by sub-system experts to synchronize and automate their domain specific operations. In LHCb, at top-level, central rules integrate the various sub-systems.
- ALICE uses SMI++ too, but automatic error recovery is only performed in a few specific cases.

There are two distinct decision making or reasoning approaches: Centralized or Decentralized. In the Centralized approach all rules are in one single repository and there is one central engine that has access to all rules and all necessary data, this is the case of Esper for example. In the Decentralized approach each sub-system deals with its own local problems, hierarchically, possibly in parallel. This is the case of SMI++ and also the ATLAS CLIPS and the CMS RCMS implementations.

User Interfacing

Many types of User Interfaces are used within each control system: there are alarm screens and message displays to warn the operators about problems, there are many monitoring displays providing information about the most important areas of the experiment and there are

ISBN 978-3-95450-139-7

operation interfaces allowing the operator to interact with the system, of which the most important are the Run Control and the DCS Control. Again here different tools were used by the four experiments:

- LHCb and the four DCS systems use the JCOP Framework and its very powerful PVSS Graphic Interface Builder. As an example the ALICE DCS User Interface is shown in Fig 6.
- ATLAS used Java in order to build a modular Run Control, for which central and sub-system developers can develop and contribute their own modules.
- CMS uses WEB tools, Javascript + HTML, to design the Run Control (Fig. 7).
- ALICE's Run Control uses Tcl/Tk.

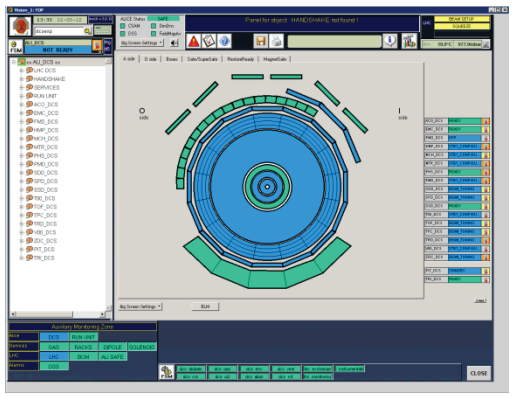

Figure 6: ALICE DCS Interface.

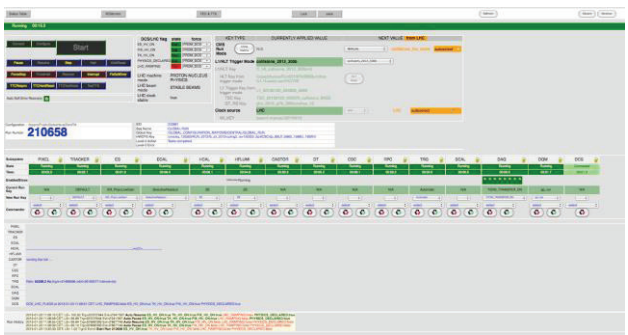

Figure 7: CMS Web-based Run Control.

OPERATIONS

All four experiments run 24 hours a day, 7 days a week during LHC running periods (several months a year). The number of operators on shift at any point in time is quite different in the different experiments:

- ALICE: 4 - Shift Leader, DCS operator, ECS + DAQ operator and Data Quality + High Level Trigger operator
- ATLAS: 8 - Shift Leader, DCS operator, Run Control operator, Trigger operator, Data Quality operator, plus three sub-detector operators
- CMS: 5 - Shift Leader, DCS operator, Run Control operator, Trigger operator and Data Quality operator
- LHCb: 2 - Shift Leader (DCS + Run Control operator) and Data Quality operator

SIZE AND PERFORMANCE

Even though different experiments made different choices, the size of the Control Systems is comparable. The amount of computers (PCs) needed to control the various parts of the experiment is summarized in Table 1.

Table 1: Size of Control System in PCs

	DAQ	DCS
ALICE	1	~100
ATLAS	32	130
CMS	12	~80
LHCb	~50 (+ ~50 HLT)	~50

Table 2: Some Selected Performance Numbers

	ALICE	ATLAS	CMS	LHCb
Cold Start to Running (min.)	5	5	3	4
Stop/Start Run (min.)	6	2	1	1
Fast Stop/Start (sec.)	-	<10	<10	<10
DAQ Inefficiency (%)	1	<1	<1	<1

Needless to say that all four Experiments' control systems work perfectly as can be seen in Table 2, in particular looking at the DAQ Inefficiency row.

ACKNOWLEDGMENTS

Many thanks to the colleagues in the LHC Experiments and in the EN/ICE group for their input and assistance, in particular: A. Augustinus, V. Barroso, F. Carena (ALICE); G. Miotto, S. Schlenker (ATLAS); F. Glege, A. Petrucci, H. Sakulin (CMS) and F. Varela (JCOP).

REFERENCES

[1] D. R. Myers et al, "The LHC experiments Joint COntrols Project, JCOP", Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems, Trieste Italy (1999)
[2] S. Schmeling et al, "Controls Framework for LHC experiments" Proc. 13th IEEE-NPSS Real Time Conference, Montreal, Canada (2003).
[3] PVSS-II/WinCC-OA http://www.etm.at/index_e.asp
[4] B. Franek and C. Gaspar, "SMI++ - an object oriented Framework for designing distributed control systems", IEEE Trans. Nucl. Sci. 45 4 1946-50 (1998).
[5] OPC http://www.opcfoundation.org/
[6] C. Gaspar et al, "DIM, a portable, light weight package for information publishing, data transfer and inter-process communication", Comp. Phys. Comm. 140 102-9 (2001).
[7] F. Carena et al, "The ALICE experiment control system", Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems, Geneva, Switzerland (2005).
[8] G. Lehmann Miotto et al, "Configuration and control of the ATLAS trigger and data acquisition", Nuclear Instruments and Methods A, 623, Issue 1, Nov 2010, p. 549-551 (2010).
[9] G. Bauer et al, "First Operational Experience with a High-Energy Physics Run Control System based on Web Technologies", IEEE Trans. Nucl. Sci. 59 4 1597-1604 (2012).
[10] CORBA http://www.corba.org/
[11] WebServices http://www.w3schools.com/webservices/
[12] CLIPS http://clipsrules.sourceforge.net/
[13] Esper http://esper.codehaus.org/

ISBN 978-3-95450-139-7