LHC-GCS: A FRAMEWORK FOR THE PRODUCTION OF 23 HOMOGENEOUS CONTROL SYSTEMS

G. Thomas[1], K. Azarov[1], R. Barillère[1], S. Cabaret[2], N. Kulman[1], X.Pons[1], J. Rochez[1],
[1]CERN, Geneva, Switzerland, [2]UPJV / LMBE-ESIEE, Amiens, France – CERN, Geneva, Switzerland.

ABSTRACT

The LHC experiments' Gas Control System (LHC GCS) [1] aims to provide the LHC experiments with homogeneous control systems (supervision and process control layers) for their 23 gas systems. To ease the production of these control systems, it has been decided to develop a library of components, the LHC GCS framework, and to adopt a model-driven automatic code generation approach.

The first milestones of the project have been achieved. The components of the LHC GCS framework have been developed as well as the code generation tools. A first control application has been built and put into production, a second is in preparation.

This paper describes the components which have been developed to ease the production of the supervision and process control layers of the control applications. It identifies in particular the additions and extensions to the UNICOS [2] libraries and how the framework could be re-used for other applications. The paper also explains how the LHC GCS framework is associated with the LHC GCS automatic code production [3].

INTRODUCTION

The development and maintenance of the control systems of the four LHC experiments require non-negligible resources and effort. The Joint COntrol Project (JCOP) [4] has been set-up to find common solutions for the four LHC experiments. In the context of JCOP, LHC GCS has been initiated to provide control applications for all the LHC experiments' gas systems.

The LHC experiments require gas mixtures for 23 of their sub-detectors. A single CERN group has been mandated to provide the experiments with gas systems. This group has designed a modular architecture in order to build and to maintain homogeneous gas systems. Although these gas systems are very similar, they are not identical. They are all built from the same set of functional modules but the number and type of these differ from one system to another. In addition, some of the functional module devices are optional.

The LHC GCS instances are turn-key control applications which provide end-users with a consistent look and feel. As they must be developed and maintained with a small team, it has been decided to first produce libraries and tools, the LHC GCS framework, and then to develop all instances from this framework.

STRATEGY

As described in [5], the nature of the process led to the selection of industrial tools and technologies for the implementation of all layers of the LHC GCS instances: A Supervision Control And Data Acquisition (SCADA) system for the supervision layer (ETM's PVSSII), Schneider Programmable Logic Controllers (PLC) for the process control, standard protocols for the middleware and field buses for the access to the devices.

While collecting information from developers of similar systems, UNICOS [7], a framework to develop industrial applications, has been identified as a tool offering solutions to most of the LHC GCS requirements. It has thus been decided to collaborate with the UNICOS team to develop and maintain common UNICOS tools and to base the LHC GCS framework on the UNICOS one.

The LHC GCS framework consists of the Object components to handle specific hardware devices, SCADA components to provide data-driven solutions for the implementation of all the functionality of the supervision layer and PLC components for data-driven generation of the LHC GCS instances' specific process control code. The components are designed in such a way that they can be integrated in a model-based approach [3] which allows the automatic code generation of a complete LHC GCS instance.

ARCHITECTURE

The LHC GCS framework architecture has been driven by the UNICOS architecture. UNICOS-based applications, and therefore LHC GCS instances, are typical three-layer control applications consisting of supervision, middleware and process control. The PLC code of a UNICOS application is structured around a hierarchy of objects which have their respective 'proxies' in the PVSS layer. The middleware handles the exchange of commands and status between the PLC objects and their PVSS proxies.

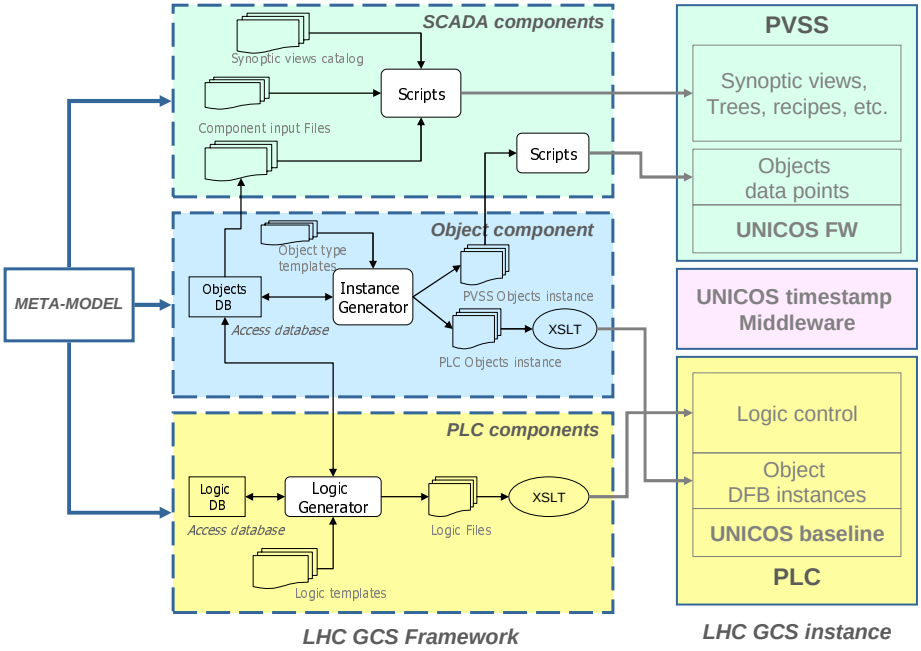


Figure 1: The components of LHC GCS framework.

The core of the LHC GCS framework is the Object components which are based on a common UNICOS tool (Instance Generator) to which LHC GCS types have been added. These types have been designed according to the UNICOS patterns.

The LHC GCS framework SCADA components ease the development of PVSS features such as synoptic and trend views, trees of views, etc. They obviously depend on the Object components. They consist of inputs files which describe in detail the pieces of information to handle and scripts which create the PVSS displays and data points from these files. The input files can be produced manually (with text or XML editors) or can be extracted automatically from databases.

The LHC GCS framework PLC specific components are based on the UNICOS tool (Logic Generator), they help in the production of the application-specific PLC code (e.g. closed loop controls, interlock handling, etc.). They consist of generic source files which are pre-compiled for the PLCs. The pre-compilation requires parameters that are either manually entered or extracted from a database [3].

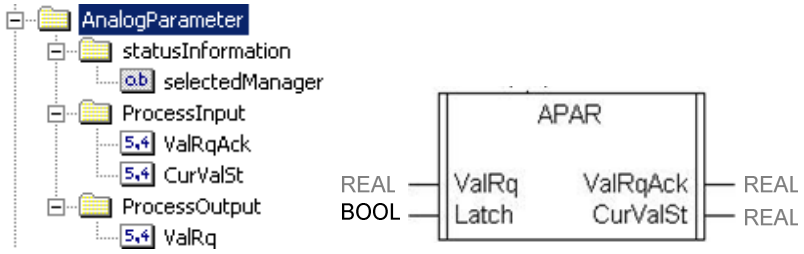
OBJECTS COMPONENT

As described in [2], the UNICOS framework Object components consist of libraries to implement many types of objects (e.g. I/O, field objects, and high level control objects). Each of the objects is evaluated in the PLCs and has a proxy in the PVSS layer. These proxies are represented by means of icons (widgets) and operated by means of dedicated displays (faceplates); they have been developed for each class of objects. A communication layer implements the exchange of object status and commands between PLCs and PVSS. In order to have consistent PLC and PVSS layers, they are generated from a unique database using a data-driven tool, the UNICOS Instance Generator.

New classes of objects have been introduced for the LHC GCS applications. They deal with specific gas devices (e.g. Mass Flow Controllers) and with specific functionality (e.g. objects for recipes).

For instance, the xPAR object classes have been defined for the recipe component which is described later in this paper. They are simple objects with no connexion to physical devices. They are

used as a data-transfer mechanism between the SCADA and the PLC layers. This component is composed of 3 types: APAR, DPAR, WPAR for respectively analog, digital and word parameters. Figure 2 shows the representation of the APAR object type in PVSS (defined by a DP Type) and in the PLC (defined as a DFB).



(a) **(b)**

Figure 2: APAR class in PVSS (a) and in PLC (b).

In addition template files and libraries have been developed for each object class to enable their automatic generation by the Instance Generator and the PVSS scripts.

SCADA-SPECIFIC COMPONENTS

The purpose of the SCADA components is to replace the interactive configuration phase of the supervision layer. The SCADA specific part of the LHC GCS framework is made of 6 data-driven components described hereafter:

- The synoptic view component allows the production of similar but not identical views;
- The trend view component produces all trend views of a LHC GCS instance.
- The view trees component configures the hierarchical organization of the synoptic and trend views of an LHC GCS instance.
- The recipe component provides tools for the configuration and operation of recipes of an application.
- The anomaly component is a solution to build synoptic views displaying the alarm objects which can raise the interlock of a given higher-level object of the control application.
- The alert summary component allows the configuration of PVSS alert summaries for these high level objects.

All components are built upon a similar principle (Figure 3). They consist of scripts which produce displays and/or data points in the PVSS DB from LHC GCS instance-specific input files (e.g. synoptic view templates, objects description files). They are built on top of the Object components.

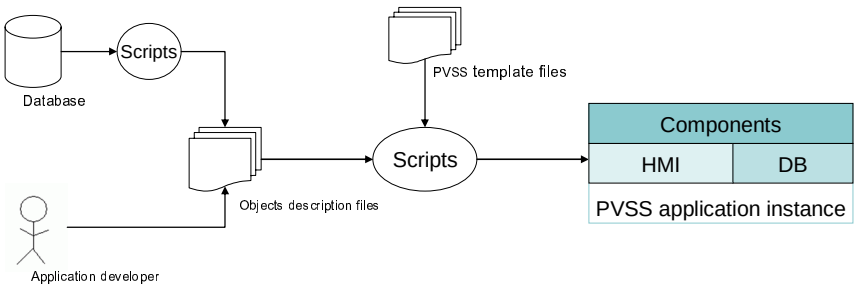


Figure 3: LHC GCS SCADA framework component

The LHC GCS framework components can be re-used in UNICOS-based control applications. The following paragraphs describe three components which are the most interesting.

RECIPE COMPONENT

Recipes are a common feature of SCADA systems. They basically provide operators with a means to select predefined collections of values to parameterize the control system. PVSS supports the concept by means of two data point types: "_rct" for recipe type and "_rcp" for recipes. A recipe type defines which parameters (e.g. commands, input values or set points) will be given a value:

myRecipeType = {temperatureSetpoint, valve1Setpoint, pressureControllerValue};

Several recipes can be instantiated for a given recipe type:

- myRecipe1 of myRecipeType={120ºC,20%,120mbar};
- myRecipe2 of myRecipeType={500ºC,80%,200mbar};

The aim of the GCS recipe component is to extend the PVSS recipe facility: to automate the creation of recipe types and recipes for a given LHC GCS instance, to guarantee that all recipes have been correctly downloaded to the PLC and provide the operator with feedback from the peripheral devices.

To handle the recipe types and recipes the application developer is provided with dedicated PVSS panels. During runtime he can interactively define recipe types and recipes for his control application using these panels. However, a typical LHC GCS instance requires about 30 recipe types, some of which can contain hundreds of parameters.

The automatic creation consists of scripts which instantiate the _rcp and _rct of a given LHC GCS instance into the PVSS database. They need an input file listing all parameters and default values of the required recipe types. The script is executed at configuration time by a simple user action which is faster and less error-prone.

The PVSS "activate" recipe command, guarantees that the recipe values are sent to the PLCs when operators select a given recipe. However, PVSS does not implement any feedback to check that these values have been received by the PLCs. The LHC GCS solution is based on **x**PAR object classes which implement a simple way to download recipe values (without requiring any operational mode) and provide feedback as in any UNICOS standard objects. The feedback is used to guarantee that the PLCs will not use a set of recipe elements which has been partially downloaded. After the activation, the feedback values are checked against those which were sent. If the transfer has been successful, PVSS sends a command to the PLC to enable the use of the values sent.

The component has been implemented using PVSS standard technologies. For the automatic instantiation of the recipe elements in PVSS, a configuration panel has been implemented to gather the input file (file describing the recipes of a LHC GCS instance) and scripts written in the PVSS scripting language to process the input file and create the data point elements in the PVSS database. The input file is coded in ASCII CSV format. The file can be written manually or automatically extracted from a database [3]. The PVSS panels provided have been used for recipe editing. The PVSS activation panels have been customized to deal with the xPAR download mechanism and to display feedback information. The download mechanism has been written in the PVSS script language.

The LHC GCS recipe component is delivered as a single component package (panels, scripts, etc). It is installed on top of the UNICOS and GCS components using the JCOP framework [6] installer tool and can be used in any control application providing that the same control technology is used.

WINDOW AND TREND TREE COMPONENTS

The window and trend trees are UNICOS framework components which have been implemented according to the JCOP hierarchy management [6]. They allow the organization of views (trends or synoptic) of a control application in a hierarchical way. Operators are provided with a Microsoft file explorer-like window (Active X) to navigate through the views. Window and trend trees have a data-structure type; they are composed of nodes, sub-nodes and leaves. Nodes and sub-nodes in the trees are either static elements or can be attached to a file (synoptic views or trend views for respectively the window and trend). Each node can have several children (leaves) which point to views.

There is one window and trend tree per UNICOS-based application. Each tree is configured interactively by the application developers. As a typical LHC GCS instance can have large trees, the LHC GCS add-on consists of providing the application developer with tools to ease the configuration of these trees for any LHC GCS instance. The principle relies on input files describing the tree hierarchies and scripts to translate the information of the input files and write it to the corresponding PVSS data points. The input file can be written manually or generated from a database and contains the information about the tree structure (node and children relationships) of a given LHC GCS instance.

Then standard PVSS scripts and JCOP framework functions are used to parse the file and create the data points in the PVSS database. The standard UNICOS active X window is then used to display and navigate through the trees.

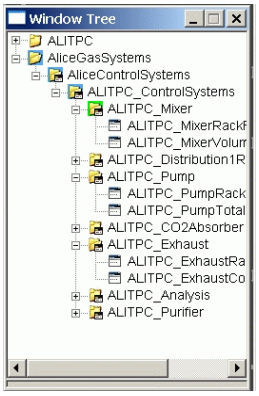


Figure 4: Window Tree of Alice TPC

The LHC GCS add-on is composed of a set of PVSS libraries. It can be re-used in control applications provided that the UNICOS framework is installed in the application. As described in [3] the files are not produced manually but from a database using an XML generator (tool written in C# language and compiled using .NET framework 1.1).

ANOMALY COMPONENT

This component automates the creation of panels displaying the alarm object widgets of all alarms which are logically linked to PCOs (See PLC logic component section). PCO objects are the nodes of the UNICOS objects hierarchy. The application specific code is organized around them, for instance by associating each of them with a Finite State Machine. When an alarm object is associated with a PCO it can trigger state transitions when active and/or raise PCO interlocks. The alarm objects widgets are then sorted according to their nature in the anomaly panel of each PCO. A typical LHC GCS instance has about 15 of these panels but a larger one may have 50 of them.

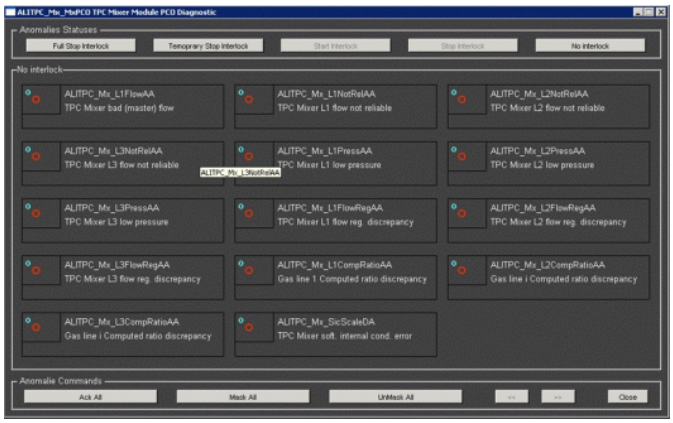


Figure 5: Anomaly panel of Alice TPC Mixer.

This component is implemented by means of PVSS scripts which read a file in XML format listing the alarm objects and their relations with the PCOs. At configuration time the scripts add the widgets in the appropriate windows of the panel and adjust its size.

PLC LOGIC COMPONENT

In addition to the libraries implementing the LHC GCS classes in the PLC, a component has been developed to ease the development of the application-specific code required for the closed-loop controls, the interlock detection, etc.

The PLC code of a UNICOS application is organized around a hierarchy of objects: low-level objects (e.g. valves, heaters, PID controllers) are driven by higher level objects (PCO). Each low-level object is driven by means of a dedicated routine (called sections in the PLC development environment). The implementation of a PCO requires a set of 7 sections. Each of these 7 sections has a specific purpose. PCO sections are either generic ones implementing objects connection pattern (e.g. propagation of ioError along the parent child relationship) or application specific sections (e.g. the Finite State Machine of a gas mixer, the interlock evaluation of a pump module). A UNICOS design pattern describes (see [2]) how a complete PLC application is structured with these routines.

The basic principle of the PLC logic component is then to produce a set of re-usable files, establish a relationship between these files and the objects of a LHC GCS instance and let a tool gather them to build the full PLC code according to the UNICOS PLC code structure. A file contains the logic required to drive a low level object of one of the 7 routines of a PCO.

A file typically contains variables to handle object names; it can then be used for several gas systems. For instance when a device is fitted for a given purpose (e.g. the pump bypass valve), in these gas systems, it is driven according to the same principle. When a unique file can not be used, one can produce several files and associate them appropriately with the objects.

To avoid multiple similar files, the file format allows the introduction of test conditions or queries to database in order to add or remove lines of PLC code. A unique file can then hold all the require variations of a section. These more complex files are typically useful to implement the 7 PCO sections. One can for instance specify by means of parameters if optional objects are fitted, and introduce these parameters in a test condition. The queries are typically useful to generate, in a PCO section, a line of code for each child object of a given type.

The component has been implemented using standard technology. The files are in XML format embedding PLC code (e.g. the standard Structure Text PLC language) and Visual Basic functions. The tool is based on a Microsoft Access database in which the objects of a LHC GCS instance are associated with files. The parameters represent the properties of the relationships. This database can be populated manually or by high level generators [3]. Visual Basic scripts pre-compile the files to produced intermediate XML files. The test conditions and queries are evaluated during this phase. Finally the files produced are combined into a single XML file using the XSLT transform which is then imported in the PLC development environment.

CONCLUSION

A first version of the LHC GCS framework implementing all necessary functionality has been released. This version has been used to produce the first of the 23 LHC GCS instances. The second is in preparation while the gas system hardware is being installed.

The LHC GCS framework allows time saving in the generation process of a control application. In addition it guarantees homogeneity of the code produced for all LHC GCS instances. By replacing a lot of interactive configuration phases, it reduces the number of configuration errors. Although they are used in combination with a model-driven [3] approach for the LCH GCS project, most of the framework components can be re-used in other UNICOS-based control projects.

REFERENCES

[1] The LHC GCS project, URL: http://itcofe.web.cern.ch/itcofe/Projects/LHC-GCS/

[2] P. Gayet et al., “UNICOS a framework to built industry-like control systems, Principles and Methodology”, ICALEPCS’2005, Geneva, Switzerland, October 2005. [WE2.2-6I]

[3] G. Thomas et al., “LHC GCS: A model-driven approach for automatic PLC and SCADA code generation”, ICALEPCS’2005, Geneva, Switzerland, October 2005. [FR2.1-6O]

[4] The Joint Control Project, URL: http://itco.web.cern.ch/itco/Projects-Services/JCOP/

[5] R. Barillère et al., “LHC GCS: A homogeneous approach for the control of the LHC experiment gas systems”, ICALEPCS’2003, Gyeongiu, Korea, October 2003.

[6] O. Holme et al., “The JCOP Framework”, ICALPECS'2005, Geneva, Switzerland, October 2005. [WE2.1-6O]

[7] C.H. Sicard et al., “Deploying the UNICOS Industrial Controls framework in multiple projects and architectures”, ICALEPCS’2005, Geneva, Switzerland, October 2005. [WE3A.2-6O]