A MODULAR APPROACH TO DEVELOP STANDARDIZED HVAC CONTROL SYSTEMS WITH UNICOS CPC FRAMEWORK

W. Booth, B. Bradu, E. Blanco, M. Quilichini, M. Bes, M. Zimny, R. Barillere,
CERN, Geneva, Switzerland

Abstract

At CERN there are currently 200 ventilation air handling units in production, used in many different applications, including building ventilation, pressurization of safe rooms, smoke extraction, pulsion/extraction of experimental areas (tunnel, cavern, etc.), and the ventilation of the computing centre. The PLC applications which operate these installations are currently being revamped to a new framework (UNICOS CPC [1]). This work began 3 years ago, and we are now in a position to standardize the development of these HVAC applications, in order to reduce the cost of initial development (including specification and coding), testing, and long-term maintenance of the code. In this paper we will discuss the various improvements to the process, and show examples, which can thus help the community develop HVAC applications. Improvements include templates for the "Functional Analysis" specification document, standardized HVAC devices and templates for the PLC control logic, and automatically generated test documentation, to help during the Factory Acceptance Test (FAT) and Site Acceptance Test (SAT) processes.

OVERVIEW

The development of an HVAC (Heating, Ventilation and Air Conditioning) control system follows a standard development cycle including data gathering, requirements specification, application development, implementation, and testing, as outlined below.

DATA GATHERING

The data gathering process collects information on the hardware that is, or will be, on-site. The process control overview is shown in the Process & Instrumentation Diagram (P&ID), see Figure 1. The electrical wiring diagram provides the information on the specific signals available in the PLC. Early on in the migration of cooling and ventilation applications to the UNICOS CPC (Unified Industrial Control System: Continuous Process Control) framework, a standard input/output (IO) list format was adopted, which provides all the pertinent information regarding the IO (e.g. unit name, unit number, actuator, position, hardware type, address, etc), based on the electrical wiring diagram, in a convenient form (Excel spreadsheet), see Figure 2. These two documents form the initial inputs for the control system design and requirements specification.

.

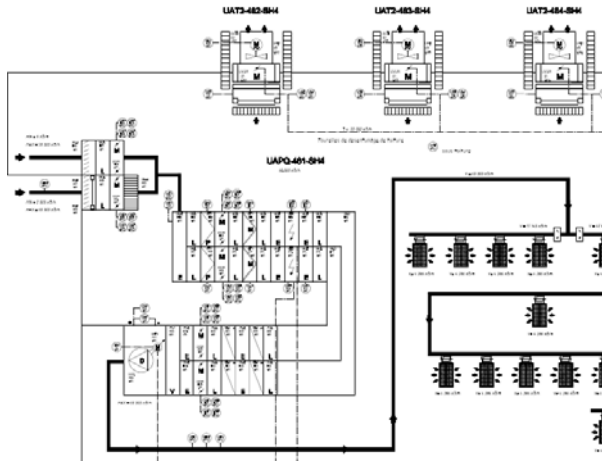

Figure 1: Example Process and Instrumentation Diagram (P&ID)

DeviceIdentification	Unit Name	UAPQ
	Unit Number	481_1
	Actuator	UMRM
	Position	B04
	Generic Name	UBLP
	Special Name	
DeviceDocumentation	Description	DAMPER POSITION EXTERNAL AIR 1 UVUM_B04_481.1
	GMAO	
	Remarks	Attention [2 10] V
FEDeviceIOConfig	Hard. Type	[0 10] V
	Address	PIW128
FEDeviceParameters	Range Min	0
	Range Max	100
	Unit	%

Figure 2: Example IO list format

REQUIREMENTS SPECIFICATION

The requirements of the control system are specified in the functional analysis (FA) document, which defines the functional breakdown of the control system into a hierarchy of individual units as specified in IEC 61512-1 or ANSI/ISA S88 [2]. After 3 years development of Ventilation plants, many different variations were found. Therefore, an analysis of the various ventilation control systems revealed that they are based on two key features: the Stepper (i.e. the state diagram) governing the operation of the individual unit, see Table 1, and the type of temperature regulation, see Table 2.

ISBN 978-3-95450-148-9

Overview of AHU Steppers

The result of the analysis of the variation of steppers among the existing applications is shown in Table 1. This demonstrates that 2/3 of the applications can be classified as type 1 through 4. As a result of this analysis, the standard requirements for type 1 through 4 were captured, as a template, that the designer can then use to create FAs in the future.

Table 1: AHU Stepper Type Summary

AHU stepper	Description	#
Type0	No stepper	1
Type1	Start-Up phase	26
Type2	Start-Up+Post-Ventilation	8
Type3	Cold-Start-Up+Post-Ventilation	10
Type4	Post-Ventilation	2
Specific	x	21

The template specification includes the state diagram, or stepper, an example of which is shown in Figure 3, as well as the definition of the operating states, and the table of actuator behaviour, as well as standard interlocks for the unit and the equipment.

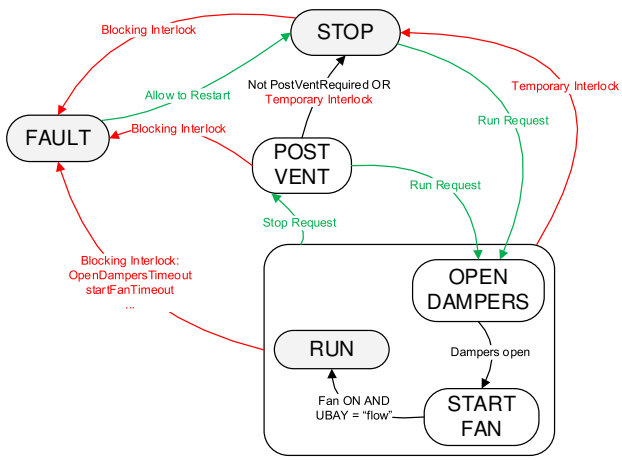

Figure 3 : State diagram for 'Type 2' AHU

Overview of Types of Temperature Regulation

The second key feature of a ventilation control system is the type of temperature regulation. After analysing the applications which have been deployed, it was found that 2/3 can be classified into a few different types, thus allowing us to standardize the implementation.

Table 2: Temperature Regulation Summary

Regulation	Description	#
Type0	No Temperature regulation	7
Type1	1 Cascade for Ambient+Supply with split range with set point filtering on ambient	19
Type2	1 Single Loop for Supply	8
Type3	Type 1 + De-humidification on cooling valve	14
Type4	3 cascades in parallel for supply	1
Type5	3 single loops in parallel for supply	2
Specific	x	17

IMPLEMENTATION

Once the requirements for the ventilation control system have been clearly specified in the Functional Analysis document, the development phase starts. Normally this phase of the project can take up to 4 weeks, depending on the complexity of the project.

The first phase is building the UNICOS CPC specification (spec) defining all the objects in the control system, beginning with the Ventilation Spec Tool as described in the next section.

The next phase is creating the specific logic for the various equipment, per the requirements.

Finally, the PLC program is tested in various phases, initially in simulation on a real PLC in the lab, Factory Acceptance Testing (FAT) and finally on site on the real PLC, Site Acceptance Testing.

Spec Creation with Ventilation Spec Tool

Based on the IO list mentioned above, a tool called the Ventilation Spec Tool has been developed. It takes an IO list and creates an initial version of the UNICOS CPC specification (spec) which defines all the devices in the control system. Note, this is very similar to the Cooling Spec Tool, see [3], except it accounts for the unique naming convention employed in Ventilation systems.

Completing the UNICOS CPC Spec

Once the user has an initial version of the specification, the control engineer completes it by hand, with the control devices needed for the project. These comprise interlocks, regulation loops, operator parameters and calculations, as specified in the Functional Analysis. This is one of the most tedious parts of the control development. There are a few possible ways of automating this, none of which are entirely flawless. The interlocks/alarms could be generated directly from the Functional Analysis, but this is not perfect because of the lack of formal language and signal names which do not match the IO list. Alternatively, a predefined

ISBN 978-3-95450-148-9

list of alarms for a given equipment could be an alternative, but this tends to be incomplete because the alarms at the unit level (i.e. the master of a group of actuators) cannot be easily generalized.

Once the user has a completed version of his specification, he can then generate all the instances and standard logic for the PLC and create a preliminary version of the control code.

Logic Development Using User Templates

The next step is to create specific logic for the individual units and actuators, based on the requirements. In the baseline UNICOS CPC framework, all 'user defined logic' is, as the name implies, left up to the user to define, either in the IDE (Integrated Development Environment) provided by the PLC supplier (either Unity for Schneider PLCs or Step-7 for Siemens PLCS), or by means of 'templates', which allow more advanced programming capabilities of the PLC logic by means of the use of scripting languages such as Python. However, these templates are completely open for the user to create. In the case of the cooling and ventilation applications, this open approach, resulted in each programmer (there have been up to 10 individuals programming cooling and ventilation PLC applications over the last 3 years) adopting his own approach and the code was somewhat heterogeneous, and more difficult to maintain as a result.

Therefore, more recently, new templates have been adopted for each given type of UNICOS CPC field object. The advantages of these new user templates is shown in Figure 4. They include standard implementations of the various features needed to program ventilation units. For example, work time counters, and "anti restart" protection, which ensures that an equipment cannot be turned back on immediately after being switch off, to avoid overheating. The programmer just has to enable a feature by adding a keyword to his specification file, and in some cases providing a few additional UNICOS objects for storing counters, or temporary variables. This avoids unnecessary work and programming errors; once you have defined a given function, it can easily be re-used elsewhere.

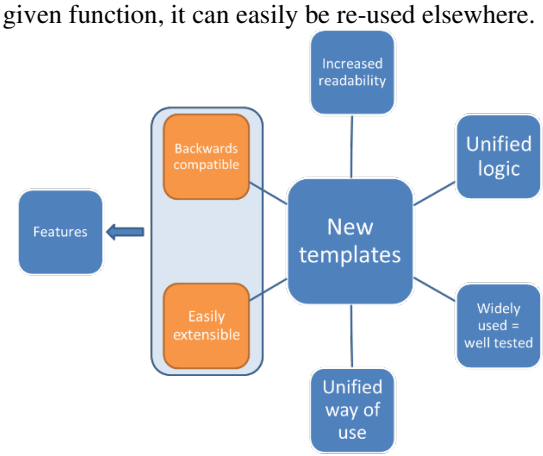

Figure 4: Advantages of new user templates

In addition, the templates make it simple to specify the operation of equipment as a function of the stepper state. As explained above, a significant piece of the functional design, is the state machine for a given unit. Each state has a name, and using the HVAC templates, it is easy to specify in which state a valve should be open or closed, or in which state a fan should be regulating, or at a fixed speed. This connects the programming with the functional specification, and therefore simplifies the overall complexity of the system.

In the future generating the initial logic directly from the Functional Analysis could be possible, once a formal way for designers to specify their requirements is available.

SCADA Synoptics

The final step before testing is the development of SCADA synoptic views, using Siemens WinCC OA, the standard solution adopted at CERN. The synoptics are developed based on the P&ID following internal standards, with additional input from the operators where necessary. The process of generating synoptics from the P&ID could be automated now that the P&IDs are being developed in more modern tools.

In the majority of cases, there is a requirement to develop a local operation facility based on industrial touch panels. The constraint is 100% availability, as the user needs to operate the plant on-site even in the case that network access is not available. This represents significant additional effort and it would be preferable to generate these touch panel views directly from the WinCC OA synoptics, but this is currently not possible.

TESTING

Once the PLC control system logic is implemented and the SCADA synoptics created, the test phase starts. First in a simulation in a lab PLC, a static simulation is built, using a custom-made PLC function to simulate the IO of the actual hardware. This involves modifying the PLC code slightly in order to wrap-around the actuator outputs to the actuator inputs, with some simple dynamics. Also normally-closed fault inputs are artificially forced, and delays are added to other inputs (pressure and flow switches, etc.) as a crude approximation of the dynamics of the installation.

Once a working simulation is built and the code is loaded on a PLC in the lab (this is quick and can be done in less than an hour) the Factory Acceptance Testing using the FAT commissioning file as outlined below is started.

FAT / SAT Commissioning File

During the design, development and test process, the amount of time testing and that of implementing the requirements are approximately the same. Therefore, it is imperative that an exhaustive test document is established, which covers all the requirements. Thus a commissioning file for use during Factory Acceptance Testing (FAT) in the lab and Site Acceptance Testing (SAT) on-site is generated. This file gives an overview of the application, including all inputs/outputs (IO), all actuators (including the parameterization, if any), all parameters which can be

ISBN 978-3-95450-148-9

modified during plant operation, a list of all the alarms which are sent to the CERN control centre (CCC), as these will be visible to the operators who monitor these applications 24/7, and a list of all the regulation loops.

Also, broken down by individual unit of operation (for example, an extraction unit), this file includes the list of alarms, see Figure 5, the stepper states, the various actuators, the parameters, and the commands, and where possible a comparison of the AF requirement against the actual code implementation (a traceability matrix). Thus the tester (who in most cases is someone other than PLC programmer) can easily navigate the various pieces of the program, in order to test, validate, and ensure compliance with the high-level functional requirements.

Name	UAPQ_481_AL6
Description	PROBLEME TEMPS D'OUVERTURE REGISTRES
FA Condition	Open Dampers Delay 180 s
Code Condition (for reference only)	DB_GRAPH_UAPQ.OPEN_DAMPERS_AR.X OR DB_GRAPH_UAPQ.OPEN_DAMPERS_AN.X
Type	FS
Master	UAPQ_481
Folio	4.10
Threshold	
Delay	UAPQ_481_AL6Dt
CCC Alarm	UAPQ_481_CCC_MAJ
Alarm activation	OK
Alarm Action	OK
Date	22/05/2015
Responsible	
Remarks	

Figure 5: Example of an alarm for a particular unit, with requirements, completed during FAT testing

In addition, the stepper is extracted from the code and displayed visually, in an automated way, in order to validate the implementation against the requirement. This was previously not possible without manually opening each stepper in the suppliers' IDE.

In fact, this commissioning file can be used for any UNICOS-CPC PLC program, and is delivered with the UNICOS-CPC resource package, but it has been developed in close collaboration with the Cooling and Ventilation group, based on their significant testing expertise developed during the migration of their old PLC applications to the UNICOS-CPC framework.

HVAC FOR LHC SURFACE BUILDINGS

For the renovation of the LHC surface building ventilation systems, the control systems of 20 surface ventilation buildings will be re-engineered, beginning with those which contain the surface cryogenics installations.

In order to expedite the control system development, a generic control system and applicable set of templates were developed. These are used to generate the 7 similar installations by replacing generic names with the specific installed equipment codes. This approach ensures that all the control systems are alike, and significantly reduces the time it takes to develop each application. This approach works well for applications which are identical. However, the remaining 13 surface installations, which will be completed in 2016, are slightly different and thus the current approach will have to be adapted. In general, over the past 3 years we have commissioned ~60 ventilation applications, and during the next 5 years, we expect to commission a further ~100, and hopefully these improvements will significantly reduce the development effort.

CONCLUSION

A noticeable improvement of the process of developing HVAC control systems with the UNICOS CPC framework has been done. This comprises improved templates for creating the requirements (Functional Analysis), implementing the PLC code (using user templates), and testing (using the commissioning file). However, there is still room for improvement such as enhancing the process to automatically generate more code, directly from the requirements if possible, and generate the SCADA synoptics directly from the P&IDs. Also the IO list could be generated automatically from the wiring diagram itself, thus simplifying the design process yet further. With these additional improvements, coding errors would be drastically reduced, and the effort required to develop the control system minimized.

ACKNOWLEDGMENT

We would like to thank the CERN Cooling and Ventilation (EN-CV) group and specially the operation and control teams for working closely with us over the last few years and sharing their control system expertise, in order to improve our development process.

REFERENCES

[1] B. Fernandez et al., UNICOS-CPC6: Automated code generation for process control applications" ICALEPCS'11, Grenoble, October 2011.

[2] IEC 61512-1 or ANSI/ISA S88. Batch Control - Part 1: Models and terminology. 1995.

[3] B. Bradu et al., "Re-engineering Controls Systems Using Automatic Generation Tools and Process Simulation: The LHC Water Cooling Case", THPPC076, ICALEPCS'13, San Francisco, USA (2013)

ISBN 978-3-95450-148-9