ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA006

AUTOMATION OF THE SOFTWARE PRODUCTION PROCESS FOR MULTIPLE CRYOGENIC CONTROL APPLICATIONS

C. Fluder*, V. Lefebvre, M. Pezzetti, A. Tovar-Gonzalez, CERN, Geneva, Switzerland
P. Plutecki, T. Wolak, AGH University of Science and Technology, Kraków, Poland

Abstract

The development of process control systems for the cryogenic infrastructure at CERN is based on an automatic software generation approach. The overall complexity of the systems, their frequent evolution as well as the extensive use of databases, repositories, commercial engineering software and CERN frameworks have led to further efforts towards improving the existing automation based software production methodology.

A large number of process control system upgrades have been successfully performed for the Cryogenics in the LHC accelerator, applying the Continuous Integration practice integrating all software production tasks, tools and technologies. The production and maintenance of the control software for multiple cryogenic applications have become more reliable while significantly reducing the required time and effort. This concept has become a guideline for development of process control software for new cryogenic systems at CERN.

This publication presents the software production methodology, as well as the summary of several years of experience with the enhanced automated control software production, already implemented for the Cryogenics of the LHC accelerator and the CERN cryogenic test facilities.

INTRODUCTION

Cryogenic systems, an integral part of the infrastructure of the most important accelerators and experimental facilities at CERN, require complex industrial process control systems for operation. Their large scale and evolving requirements, functional (change requests) and environmental (updated control software components), are big challenges for developers, who have to produce error-free, robust and safe control system software.

From the very beginning the large scale of the control system for the Cryogenics of the LHC accelerator forced the use of automatic code production in the development process. The existing CERN code generation tools were adapted to cover the requirements of the control system for the Cryogenics of the LHC, which became fully operational for the first time in 2008. Experience after months of operation led to a review and optimization of the process functional analysis [1]. As a result the second major release was successfully deployed in 2010, ensuring the operability of the cryogenic infrastructure during the first run of the LHC accelerator.

The amount of changes to implement during the LHC's Long Shutdown 1 (LS1), a 2-year consolidation and maintenance work period (2013 - 2015), using the UNIfied COntrol System (UNICOS)-based complex and time-consuming software development process triggered work towards further improvements and more complete automation of some parts of the process, i.e. the software building and testing [2]. Introducing the Continuous Integration (CI) methodology to the development of the software for the LHC accelerator allowed to enter the second run of the LHC with a very reliable control system software, and with significantly improved and more efficient development process, allowing to address any modifications much safer and faster than before. Both have contributed to improving of the overall reliability and availability of the Cryogenics [3], what was especially important while facing new challenges with the beams of higher luminosity in the LHC accelerator [4].

The experiences in development and the use of the first version of the CI system along with new requirements and new projects led to further evolution of the CI system for control system software of the Cryogenics. These new developments and their applications are discussed below.

THE CONTINUOUS INTEGRATION SERVICE FOR THE CONTROL SYSTEM SOFTWARE FOR THE CRYOGENICS OF THE LHC TUNNEL

Building software for large-scale control system applications would be very difficult, if not impossible, without specialized tools facilitating the process. Automated code generation have been used in development of control system applications at CERN since many years [5]. The UNICOS framework with its Continuous Process Control package (UNICOS CPC, UCPC) is a standard for building programmable logic controller (PLC) based applications at CERN and other laboratories [6]. The framework, by providing a library of generic device types, a methodology and a toolset to design and implement industrial control applications [7], simplifies and unifies the way control system software is implemented. The architecture, the communication layer, the main data structures, the execution flow of the control applications and also development workflow is the same for all the projects and varies only in very project-specific areas.

Still, even using the framework, for very large scale control systems producing control system software is a complex and long process, requiring many time-consuming steps to be performed manually by the developer. During the LS1 it became clear that in the case of the control system for the Cryogenics of the LHC accelerator, the largest cryogenic system at CERN (and in the world), it would be very difficult to rebuild reliably the software implementing all required modifications without further automation of the pro-

* czeslaw.fluder@cern.ch

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA006

cess. Small, seemingly transparent modifications of the hardware database, changes in evolving framework or changes in the logic could have devastating impact on the software produced if not tested incrementally, in iterations and in isolation. Introducing too many changes to a single build complicates debugging, makes difficult to find causes of the problems, and in result it consumes time of the developers.

Software engineering is addressing these issues with the Continuous Integration practice coming from Agile development methods (originally - eXtreme Programming, XP). At the time (LS1) the use of the CI was very limited in the field of industrial automation (at least in case of PLC-based systems). Partially it was a consequence of closed/proprietary character of the development environments for PLCs, not providing ready to use, flexible and scriptable ways to automate software builds. Another cause that might have prevented adoption of the CI by PLC-based industrial controls could be difficulties in implementing testing environments, that, for a more complete testbed, require sophisticated and expensive hardware simulation devices replacing inputs and outputs of the original (unique!) hardware from production systems.

In case of such large-scale system the potential gains from finding a way to overcome the difficulties and optimize the development process were sufficient to use some time and resources to investigate possible solutions. Availability of libraries with programming interfaces (APIs) providing means to use the Siemens SIMATIC® environment without human interaction allowed to develop custom automation tools, i.e. the SIMATIC Step7® command-line interface (s7cli) [2]. The hardware simulation box for SIMATIC (SIMBA Profibus®) and developed UNICOS testing framework allowed to exhaustively test the most critical functionalities (i.e. alarms and interlocks, communication) in an automated way.

The possibility to execute the project build tasks automatically (i.e. those most time-consuming and the most frequent) opened a way to integrate all the software frameworks, developed tools, and automated tasks, and thus led to construction of an (almost) complete Continuous Integration solution for UNICOS CPC control applications for Siemens Step7 PLCs [2]. The CI service was built using Jenkins [8], chosen for its flexibility concerning various types of jobs on multiple platforms and wide possibilities to integrate with miscellaneous services - in general for its flexible, extendable architecture and functionality (using variety of plugins). The service made use of a number of standard IT services (i.e. Git service - repositories, Atlassian JIRA® - issue tracking, CERN Virtualization Infrastructure - virtual machines with the CI service and for build jobs execution). At the time (LS1) dedicated virtualized build infrastructure consisted 16 virtual machines with Microsoft Windows® (Server 2008®) and Linux (SLC6) systems with all the required software (i.e. Siemens SIMATIC Step7, Oracle Java® 1.7, Apache Maven, UNICOS Application Builder (UAB), Oracle Instant Client® 11, Python 2.x).

The service, configured in a way that it reflected project's and team's workflow, had significant impact on the software development process and the result - which in short was very high quality and very robust application software for 18 critically important control systems for the Cryogenics of the LHC accelerator. It also meant entering the Run 2 of the LHC with a development environment allowing to address any change requests in more efficient and secure way.

Review and Improvements

After the first three years of software development using the existing CI system one of the drawbacks of the Jenkins system became visible - a configured, heavily customized by using many plugins system became difficult to upgrade. The configurations of Jenkins itself and of many of its important plugins (like, for instance, those responsible for security credentials and authentication) were incompatible with newer versions. In consequence such upgrade could break configured complex build jobs, making the system unusable and not easy to repair. Therefore, it was worth considering setting up a new, clean instance of the service. Having gathered many experiences during years of exploitation of the system it was also an opportunity to apply additional improvements.

Functionality The work on modernisation of the system had several stages. First, the Jenkins itself and underlying plugins had to be updated to the latest stable version (Jenkins-CI 2.10). The available plugins were assessed in order to provide new and expand existing functionalities.

After the review of the plugins, it was decided to take a more generic approach creating build jobs structure and configuration. To achieve this some carefully selected plugins were used, for instance the Matrix Project plugin. The plugin allows the developer to execute multiple similar build jobs within a single one by defining a set of parameters with predefined, valid values. Each parameter is treated as an axis of a matrix containing all possible combinations and the job using the plugin is executed against each of the combinations.

Building control applications comprise a number of steps (configured as build jobs in the CI) to execute [2]. Another selected plugin, the MultiJob Plugin, allowed to create build jobs, which execution flow is defined by consecutive phases. Each phase may contain one or more steps (build jobs), executed sequentially or in parallel. Also, the jobs in the phases inherit the environment variables (parameters) defined in the corresponding MultiJob.

Incorporating this functionality allows to create a single job to execute the whole build chain - and to parametrize it. For the discussed CI system, it brought means to aggregate the jobs building control applications for the eight LHC sectors into a single, parametrized job. Its phases and subjobs correspond to the jobs in the old CI system (like "Code generation", "Compilation", "Testing/Validation"), but simple parametrization permitted to significantly decrease the number of jobs (what implies reduced maintenance efforts). The jobs using the plugin can be themselves executed by another

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA006

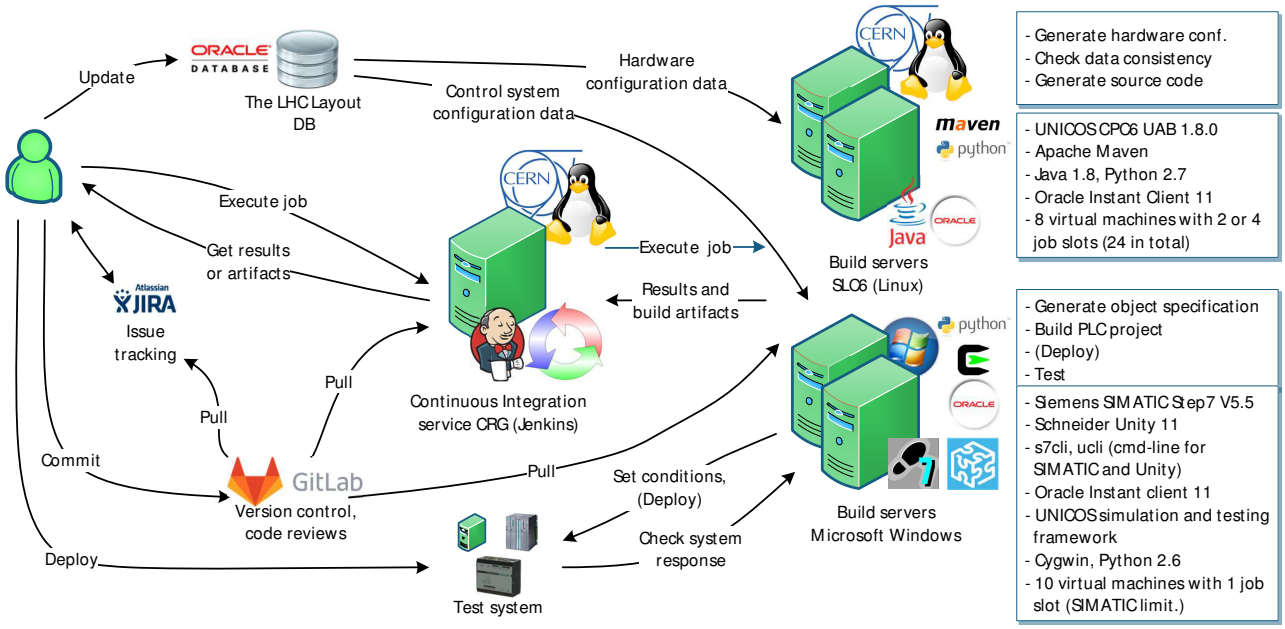

Figure 1: The updated architecture of the CI system.

MultiJob - currently all 16 applications for the LHC Tunnel can be built using a single job.

The possibility to easily parametrize the whole build chain and, in particular, let selecting a repository branch (Fig. 2), opened a simple way to build the applications using different versions of the UCPC framework. To achieve this the jobs were also reconfigured in a way that the logical content (build commands) was extracted to be used with a standard software build automation program, GNU Make, and placed in a configuration file stored in the repository. This facilitates and makes more secure addressing any changes in the build process - which since then can differ depending on the branch in Git repository.

Concerning the repository management system itself - the existing CERN's Git repository management system, Gitolite, became deprecated due to creation of a new service based on GitLab®. All repositories had to be moved to the new system and according changes were applied in the CI system. GitLab also replaced Atlassian FishEye®, software allowing to visualize and analyze changes in the source code (Fig. 1).

Optimizations The activities to review and upgrade the existing system were triggered directly by the necessity to migrate the service to the new CERN's Cloud infrastructure, which was in the period of transition from a custom virtualization system based on Microsoft's Hyper-V® to OpenStack®. However, initial tests performed on migrated worker nodes revealed a potential issue - due to the way the SIMATIC is using the disk input/output (I/O), performing many small writes with very frequent file buffer flushes, build times for jobs operating on a PLC project[1] would become unacceptable (2-3 times longer than before, with previous hypervisors offering very fast I/O). Requesting the use of better performing (and more expensive) hypervisors could be an (expensive) solution but in this case - using a typical temporary storage for building PLC projects requiring not more than 200MB - it did not seem an optimal solution. The best way to optimize performance of a virtual machine (VM), sharing I/O bus and disks with others, is using system's memory as much as possible. In case of the file storage (and software not using system file buffers effectively) this means creating a disk in the VM's memory (RAM). Microsoft Windows systems do not provide such possibility natively and it is necessary to use additional 3rd party software. At the time (2015) two free products were available: SoftPerfect RAM Disk and ImDisk. The first one was chosen for (slightly) better performance[2].

Using RAMdisk for building SIMATIC PLC projects improved performance of the CI system (even comparing to the previous system with faster I/O). The average time to compile two applications for a single sector of the LHC (ARC and LSS) was 3h, with RAM-disks - around 1h.

Optimizations were made also in organization and configuration of the build tasks. For instance, the most time-consuming task, exportation of the source code files from a SIMATIC PLC project, was optimised by introducing more parallelisation in comparison to the previous Jenkins setup.

Gains All the improvements described above proved to be very useful in further development of the control applications for the LHC Tunnel. It allowed to work efficiently on required modifications and have updates ready to deploy in a very short time. For instance, during 2016/17 Extended Year-End Technical Stop of the LHC accelerator all the 18

[1] It concerns operations like SCL source code importation or compilation.

[2] Nowadays, newer versions of SoftPerfect RAMDisk comes with a commercial license and are not free any more.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA006

LHC accelerator control applications had to be upgraded (and therefore - rebuild and redeploy) to the newer version of the UNICOS framework. The possibility to use more branches without the necessity to alter Jenkins' configuration allowed to generate and to validate multiple sets of control applications and to develop scheduled logic modifications simultaneously for the production (released) and the development (testing) version of the UNICOS CPC.

Another example of a complex update of the control applications was the implementation of the feed-forward control in order to compensate the beam effects on the beam screen temperature. It was needed to maintain stable cryogenic conditions with beams of higher luminosity in the LHC accelerator [4]. The update required applying many modifications in the UNICOS database (containing information about control system objects), what implied changes in many source code files. The modification itself and also improvements coming on the fly were deployed without any problems.

For a large scale control system, like the one for the Cryogenics of the LHC accelerator, some modifications have to be deployed and validated in the production environment just in one part of the system (one application or a sector) prior to applying them everywhere. The possibility to use multiple branches made feasible building and deploying control applications independently (e.g. with different modifications), what was difficult to achieve in the former configuration. Validated modifications are propagated to all applications and deployed in all sectors of the LHC. Particular modifications can be selected ("cherry-picked" in Git), then built and deployed in short time.

The use of improved CI system allowed to build up an efficient workflow for developing large scale control applications and thus to reduce time and efforts - and therefore human resources required. In a way, they were replaced by a larger pool of worker nodes of the build system (Fig. 3).

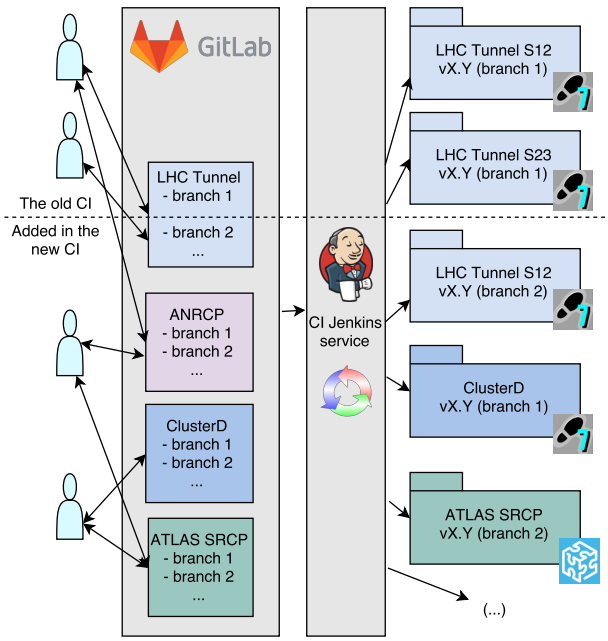

Figure 2: The workflow with the updated CI system.

CONTINOUS INTEGRATION FOR NON-LHC CRYOGENIC CONTROLS PROJECTS

The first use of CI systems for a non-LHC Siemens application was in 2015 and concerned migration to UNICOS of the 10 horizontal magnet test benches (cryogenic feedboxes - CFBs) in the CERN's magnet test facilities (SM18). Those 10 applications share the same hardware design and process logic, thus building them consists of a large number of repetitive tasks that can be automatized using a CI system and automation tools, in addition to UNICOS python templates. The CI system was set up with the same base as the existing one for the LHC tunnel (dedicated Jenkins instance, use of Git branches for CFB specific generation files, with tools like s7cli etc.). This strategy allowed to successfully deploy the 10 applications in a short period of time avoiding manual execution of many time-consuming tasks. It eases also the deployment of logic updates that have to be propagated to the 10 CFBs.

In 2017, the CI system was updated in order to follow consolidation initiated for the LHC tunnel and to standardize the use of CI tools. For this, the dedicated Jenkins instances were migrated to the one used for the LHC and Makefiles stored in the Git repository were introduced. This common structure allows to have one pool of OpenStack machines shared between all CI systems, and results in an easier maintenance of the machines and applications.

Following the successful use of CI for the SM18 CFBs and experience gained during work on the LHC Tunnel, it was decided to develop a system for the generic use of CI tools in order to facilitate the production of new applications. The system was named Cryo-apps and is composed of two parts: a Python library and an automatic build solution based on Git and Jenkins. The key feature of the Python library is that it separates the complete logic generated automatically by the UNICOS Application Builder and the logic which has to be completed by the developer. This step allows the developer to create an application without the necessity of writing Python templates for Siemens code generation and introduces simple procedure to regenerate the sources without losing the already written code, i.e. to reuse the code exported from an older version of the application. In addition, the system provides a set of predefined Python functions to generate commonly used Siemens chunks of code, like automatic generation of Cernox® curves and interpolation functions. Some other tools have also been developed in order to ease the automatic built of the code, for instance to assign symbols dynamically (UAB / Step7) and to generate compilation files.

For the automatic build part, a common Git structure is defined. It natively contains all the necessary tools for the automatic generation of a Siemens project: s7cli, the mentioned Python library and a generic Makefile. Then any application just inherit from the common Git and only project specific files have to be added. The Jenkins build job

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA006

does not need any specific configuration since the process is based on the generic Makefile.

Cryo-apps has been successfully used for several projects of different size and importance since its first version in 2016. The first projects using cryo-apps were the High Field Magnet (HFM) and ClusterD - the new magnet test stations for SM18 test hall. They are built as twin systems and therefore they share the same Python code templates. The HFM was the first one deployed and was used as a beta test for the newly developed cryo-apps system. Following the successful deployment of the HFM, it was decided to continue with ClusterD, and then to use cryo-apps for other Siemens application such as NA62 and HIE-ISOLDE cryomodules. This application is a great example of how one can fully benefit from the use of the CI tools to generate and build applications. Indeed, the application is composed of up to 6 cryomodules that share the same logic and equipments, making it one of the largest non-LHC cryogenic control system applications in CERN. It needs frequent updates since the number of cryomodules increase every year - currently 3 cryomodules are in place and the test phase is still in progress. Therefore, the CI system makes the development process more efficient.

The first successful experiences using cryo-apps and its genericity led to consider it as a standard to build every future Siemens application (such as ClusterG Infra and B180 new magnet test area).

FUTURE PLANS

Future developments will most likely be focused on integrating more use cases, in particular those using different hardware and software components. For the moment, a prototype CI was created for the first Schneider Unity®-based projects. While already helpful, further developments on automating some stages of the process and supporting tools are necessary for better coverage of the workflow. With upcoming multiple upgrades scheduled over the next two years - these will most likely have a high priority in the near future.

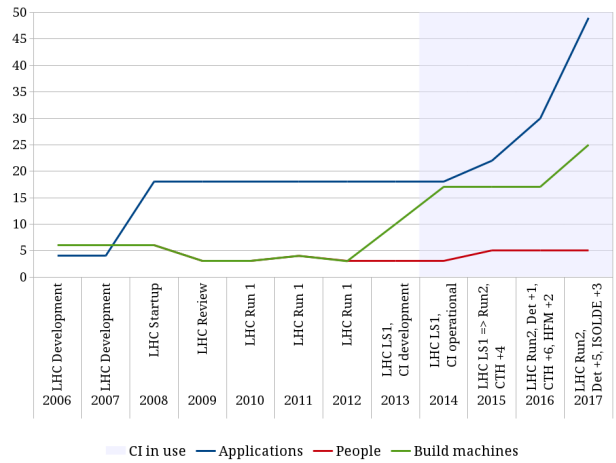

Figure 3: Applications, people, computing resources and CI usage over time.

CONCLUSIONS

The automation of the software production process in CERN Cryogenics has become an essential part of the creation of the necessary software for the control applications. The results and expectative of the update have shown that the time and resources needed to implement it were more than worthy to obtain a system which is better configurable, faster and more reliable than the previous one. The necessity of the maintenance and updates of the different components of the Continuous Integrations service is overshadowed by the benefits the system provides.

Thanks to its modularity, it can be configured to be adapted to different types of applications and even cope with different versions simultaneously. That way it is possible to modify and test an application while generating a working version of the installed software. The rise of the speed gained increasing the number of machines and optimizing them allow a faster error detection within the programming code reducing the time to get a working application. It also improved quality assurance of the control applications produced. Human error is minimized while launching the different jobs of the Continuous Integration service as all of them are already predefined. All the benefits combined permitted to evolve from maintaining 18 applications to 49 with possibility to be increased in the near future.

REFERENCES

[1] P. Gomes *et al.*, "The control system for the cryogenics in the LHC tunnel [first experience and improvements]", in *Proc. ICALEPCS 2009*, Kobe, Japan, 2009, paper WEP061.

[2] C. Fluder *et al.*, "Improved software production for the LHC tunnel cryogenics control system", in *Physics Procedia*, vol. 67, pp. 1134-1140, 2015.

[3] K. Brodzinski *et al.*, "LHC Cryogenics – Run2 (2015) summary and perspectives", Evian, France, 2015.

[4] B. Bradu *et al.*, "Compensation of beam induced effects in LHC cryogenic systems", in *Proc. IPAC2016*, Busan, Korea, 2016, paper TUPMB048.

[5] Ph. Gayet *et al.*, "Application of object-based industrial controls for cryogenics", in *Proc. EPAC 2002*, Paris, France, 2002, pp. 2013-2015.

[6] P. Modanese *et al.*, "Cryogenic Control System Migration and Developments towards the UNICOS CERN Standard at INFN", in *Physics Procedia*, vol. 67, pp. 163-169, 2015.

[7] B. Fernandez Adiego *et al.*, "UNICOS CPC6: Automated code generation for process control applications", in *Proc. ICALEPCS2011*, WTC Grenoble, France, 2011, paper WEPKS033, pp. 871-874.

[8] "Jenkins, An extendable open source continuous integration server", https://jenkins.io/