ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV017

CERN SCADA SYSTEMS 2020 LARGE UPGRADE CAMPAIGN RETROSPECTIVE

Lukasz Goralczyk[†], Alexandros Foivos Kostopoulos, Brad Schofield, Jean-Charles Tournier
CERN, Geneva, Switzerland

Abstract

In this paper we report the experience from a large-scale upgrade campaign of SCADA control systems performed during the second LHC Long Shutdown at CERN. Such periodical upgrades are dictated by the ever evolving SCADA WinCC OA system and the CERN frameworks evolution used in those control systems. These upgrades concern: accelerator control systems, e.g. quench protection system, powering interlocks, magnet alignment; control systems devoted to accelerator facilities such as cryogenics, vacuum, gas and other global technical infrastructure systems as well as the CERN electrical distribution system.

Since there are more than 200 SCADA projects covering the CERN accelerator complex and technical infrastructure, any disruption requires careful coordination, planning and execution with process owners. Having gained experience from previous campaigns and reaching a new level of automation we were able to make visible improvements by shortening the required time and reducing the personnel required. Activities, lessons learned and further improvements are presented as well as a comprehensive statistical insight of the whole campaign.

INTRODUCTION

Software upgrades are an inherent process in modern computing which also applies for present-day SCADA (Supervisory Control And Data Acquisition) applications and the infrastructure in which they run. Upgrades are a necessity to keep systems secure, provide new features and avoid software obsolescence.

At CERN there are numerous industrial control systems in different domains such as machine protection, cryogenics, cooling and ventilation and the electrical distribution system. Together they account for a total of around 200 SCADA applications. These applications are often critical for the operation of CERN's accelerator complex and experiments. The large number and broad scope of these applications poses several challenges with respect to upgrades, a key challenge being the creation of an upgrade plan which does not interfere with the operation schedule.

At the end of 2018 CERN entered its second Long Shutdown (LS2) period that normally lasts around two years (in 2020 due to COVID-19 pandemic, it was extended). This was a perfect opportunity to perform a large-scale campaign to upgrade SCADA applications to the latest software packages, as well as to perform some housekeeping activities (e.g. server maintenance). In most cases, operational constraints are not as strict as is normally the case outside of long shutdown periods.

In this paper we give an insight into the upgrade campaign that took place in 2020. We start by giving details about the motivation behind the upgrades, planning goals and the scope. Differences with the previous campaign, carried out in 2017, will be also discussed. Improvements implemented from that campaign, as well as new enhancements are a topic of another chapter. We follow by presenting details about our experiences – the good and the bad. We conclude with proposals for further improvements.

For details about the SCADA service at CERN as well as software and infrastructure details, the reader is referred to [1].

MOTIVATION, SCOPE AND PLANNING

For the 2020 campaign the motivation was:

1. Siemens/ETM has decided to stop providing support for the version 3.15 of WinCC OA [2], the software used for most of the SCADA applications at CERN. It was, therefore, necessary to upgrade all the CERN WinCC OA-based SCADA systems to the latest version of WinCC OA.
2. New features that came with the updated JCOP [3] and UNICOS frameworks [4] depended on newer releases of WinCC OA.
3. Relaxed intervention restrictions given by the LS2 period.

Planning activities started several months before LS2 began. Similarly to the 2017 upgrade campaign all control domain representatives were interviewed to gain information about the planning restrictions, acceptable downtime and any additional requests. It was also an opportunity to present how the upgrades would be executed.

With this data gathered, an initial schedule was proposed and later agreed upon with the representatives. The upgrade calendar had to take into account some restrictions as: criticality, overlap with other upgrades, operation schedule and availability of experts.

Despite the long duration of LS2, the initial assumption was to finish all upgrades in less than 6 months, with the majority of applications upgraded in the first quarter of 2020. The COVID-19 pandemic forced us to put all pending upgrades on hold, abandon the initial schedule and, later, adjust it again conditioned by the COVID-19 protective measures (e.g. teleworking). This resulted in the upgrades being spread throughout the year.

[†] lukasz.goralczyk@cern.ch

The initial scope of the upgrade was around 250 applications, but with some major improvements and the consequent simplification in the number of applications in one of the largest control project at CERN, the Quench Protection System, the scope was reduced to around 200.

COMPARISON WITH 2017 CAMPAIGN

There are several differences when compared with the previous SCADA applications upgrade campaign; these are summarized in Table 1.

Due to the relatively minor number of changes in the new WinCC OA version, the JCOP and UNICOS frameworks did not require as many adaptations as in the 2017 campaign. This allowed for a shorter testing period. One major change was the switch from the ISO8859-1 character encoding to UTF-8. Care had to be taken to ensure that the upgraded applications text was displayed correctly, as well as not to cause any internal problems.

Table 1: Upgrade Campaign Comparison

Campaign	2017	2020
WinCC OA	3.11 → 3.15 major change	3.15 → 3.16 minor change
Character encoding	ISO8859-1	UTF-8
Operating system	SLC6[1] → CC7	CC7 minor version
Timing and planning	EYETS[2] strict	LS2 relaxed
Resources	7 persons	4 persons
Tooling	Graphical user interface (GUI)	Command line interface (CLI)

In the previous campaign, a major change in the operating system (OS) was a hurdle as it was a considerable change, significantly extending upgrade time, requiring extra staff and a notably larger validation period. The campaign in 2020 had only a minor OS version change and its validation with a newer WinCC OA software was shorter and smoother.

The LS2 period was longer than EYETS (End of the Year Extended Technical Stop) allowing for a more relaxed planning, although the initial assumption was to finish the upgrades in the first half of 2020, with the majority of them in the first quarter. It's worth noting that despite the fact that a number of industrial installations were not operational, many of their control systems were still in production and required utmost care in planning and execution.

It's important to mention that other aspects of the process have not changed; these include:

1. Upgrade schedule to strictly follow directives from control domain representatives.
2. Individual upgrades execution composed of many individual steps: around 70.
3. Close cooperation with controls representatives to ensure proper operation after the upgrade.

[1]Scientific Linux CERN

[2]Extended Year-End Technical Stop

TOOLING IMPROVEMENTS

New tooling around the upgrade process was built on the previous collection of Python and bash scripts and libraries that interacted with WinCC OA.

Two new main tools were developed: *wccadm,* a Python 3 library to interact with WinCC OA and *wccauto,* a simple Python 3 framework to execute predefined steps. *wccadm* substituted a previous, similar, library which was based on unsupported Python 2. It aims to be a more reliable and complete interaction environment for WinCC OA. A major effort was made to make it easy to develop with as well as to have a richer logging support. Well formed and informational (debug) messages turned out to be crucial in the validation and, later, in the execution phase of the upgrade.

wccauto is an environment to develop specialised modules and execute them in a predefined order. Each module implements one particular function, which can be something as simple as displaying a message or, a more complex functionality like saving the SCADA application health status. *wccauto* is also an execution environment, where a predefined set of steps (modules), called plans, are run in an orderly manner (examples of steps could be 'display a message', 'copy a file', 'start the application'). It provides a command line interface and collects detailed logs from the upgrade. A typical production plan consisted of around 70 steps. Depending on the needs, a plan could be adjusted for non-standard applications.

The new tools exclusively use a command line interface (CLI); this differs from the previous approach which provided a graphical user interface (GUI). The switch simplified the execution as only a secure shell (ssh) connection was needed. It also allowed automatic offline tests performed on backups of production SCADA applications; a special non-interactive plan was used for that purpose.

Some of the upgrade steps that previously had to be executed manually were automated, leaving only a handful of operations requiring attention. The majority of the steps would execute automatically one-by-one. The tools would always provide feedback messages of what is being currently done. In case of a problem, a clear error message would be displayed with, if possible, a proposed solution. A number of pre-flight checks were done before the upgrade to detect problems early and save precious time later.

In addition to *wccadm* and *wccauto*, some supplementary programs were created including one to take and compare screenshots of synoptic views before and after the upgrade. This tool was used to detect differences in panel rendering between the old and the new version of WinCC OA, including problems related to the character encoding switch from ISO8859-1 to UTF-8. At least one bug in a panel widget, used by many applications was detected and fixed, as well as multiple panels in the PSEN application, used to supervise CERN's electrical network.

Each upgrade would produce a detailed log that would be recorded in the ticketing system. It also provided a source for post-upgrade analysis of the process. Bottlenecks could be identified and further improvements could be proposed.

CHALLENGES

Such a large scale upgrade, spanning across many control domains, poses several challenges, both at the preparation phase as well as during the execution period.

The creation of the schedule started by meeting with all control domain representatives. The purpose was to present early enough the upgrade, explain the impact on the supervision layer and the availability of the control system to the operation team and learn if there were any operational restrictions and when the upgrade should not be done.

After collecting this initial data a draft of the schedule was created. It grouped SCADA applications by their domains and planned them such that all of them would be done before mid of 2020. Any restrictions and particular wishes were taken into account. Special gaps were introduced to allow rest time for the team and as a safety net in case of unforeseen problems. Naturally the schedule would evolve with time as teams would make adjustments.

One serious, unforeseen external event that interfered with the schedule was the start of the COVID-19 pandemic. At the very beginning, as events quickly developed, we decided to put planned upgrades on hold (with some exceptions). Later we cancelled the current schedule and changed the modus operandi to a more ad-hoc planning. In the end a new schedule was created and agreed with representatives for the remaining SCADA applications. Due to lockdown and the resulting restrictions of on-site presence at CERN, it became clear that it would be necessary to perform upgrades remotely. Initially we approached this with great caution. With time we gained confidence and could resume the upgrades, doing them almost exclusively remotely. See Fig. 1.

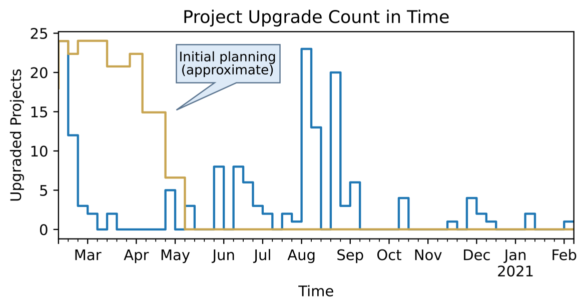

Figure 1: Comparison of upgrade counts in time between initial and adjusted planning.

At any point in time the schedule had to take into account fewer staff availability compared to the previous campaign. It was desired that the load should not reach 100% in order to give time for other parallel activities like support, operations or other developments. Thanks to the improved tooling and documentation the process was not as taxing as with the previous upgrade. For numerous applications the upgrade could run mostly in the background.

WHAT WENT WELL

Despite changes in the schedule and fewer committed resources, we succeeded in upgrading all the SCADA applications.

Developed tools and procedures proved to work well and be very reliable. They provided an intuitive interface with clear instructions on what to do and displaying the currently executed operation. In case of problems, a clear and understandable message would be displayed. Complexity of the process was well hidden from the user. Even after a longer break there was no need for an earlier procedure review as following the tool simple instructions was enough. It is important to mention that the same procedure (a plan, which as mentioned before, is a series of simple steps to be executed) was applied to the majority of the upgrades, thus avoiding possible confusion from constant changes. Parallel upgrades were possible and performed whenever possible thus greatly reducing the impact on supervision unavailability.

The ability to easily change the upgrade plan allowed us to adopt the process to non-standard SCADA applications (see Fig. 2). Additionally any reports of possible improvements would be easily accommodated in the upgrade plan.

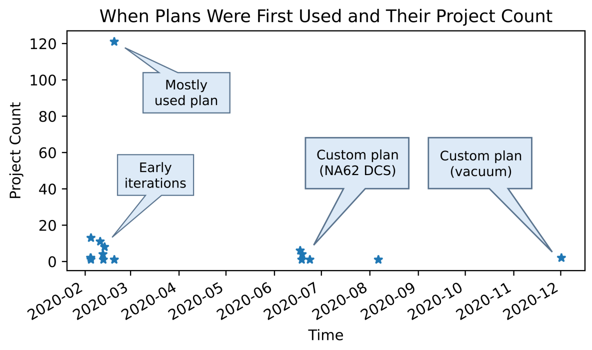

Figure 2: The number of projects using an upgrade plan (vertical axis) vs the time each plan was first used.

A key factor to the success of the campaign was the massive offline testing of the upgrade procedure done before the execution phase. It gave us the confidence especially needed for newly developed tools. A number of problems were detected thanks to detailed logs collected during the test run.

The announced total time needed for a single application upgrade was set to 4 hours initially, based on the experience of the 2017 campaign, but was then reduced to 2 hours, due to improvements made to the tooling. After the expected initial learning phase, the time was further reduced to 1 hour. See also Fig. 3.

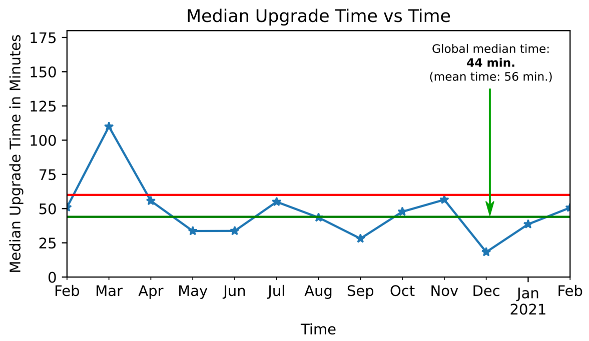

Figure 3: How upgrade time changed throughout the whole campaign.

The LS2 period gave us the comfort of a more relaxed upgrade schedule. Despite ongoing interventions we could still find time to provide support and respond to other client requests.

Finally, the rich logs produced during the upgrades gave us an interesting insight on the whole process, such as where the bottlenecks were, and what the most common errors were in the process. The analysis will prove useful in indicating what to improve in the future.

WHAT DID NOT GO WELL

Not everything went as planned and there were a couple of obstacles we needed to overcome. This section presents the lessons learned from events that didn't go well.

The execution planning was done using an Excel spreadsheet. This was initially a good choice because of its simplicity and ability to view it online using existing infrastructure. Data present in the spreadsheet was later used to create tickets in a tracking system (Atlassian Jira [5]), to enable traceability and integration with a shared calendar (Outlook). This combination was not well suited for constant, large changes. Applying changes to a selected set of tickets was time consuming. A solution to this problem would be to use a dedicated tool that would interact with the ticketing system and a shared calendar. It could also automatically create an initial schedule given some basic rules.

Despite the effort put into the automation of the process, manual steps still existed. List of top 4 slowest steps can be found in Table 2, majority of them is manual. Their automation was either not possible or too costly in terms of time. Automation not only saves time, but also prevents mistakes – we experienced that a couple of times. After the upgrades were completed some of those manual steps were automated. For changes in external systems we contacted other teams to help us with the automation.

In several instances, multiple applications were upgraded in parallel. This was due to various reasons, like a request from the control domain representative or the fact that applications hosted on the same server must be upgraded at the same time. In most cases, this parallelism saved time. It was, however, very onerous for the person performing the upgrade, and could negatively affect the timing, even at the presence of any single problem. The lesson learned was to put a limit on the number of parallel upgrades handled per person; this could be an input for a future automatic scheduling tool.

Table 2: Top 4 Slowest Upgrade Steps (Median)

Upgrade Step	Manual	Time in Minutes
Update of application startup scripts	Yes	6
Pre-upgrade scripts	Yes	3.5
Post-upgrade backup	Yes	3.5
JCOP and UNICOS components upgrade	No	2.5

We also encountered a problem with the SCADA software we use – WinCC OA. Despite extensive testing and validation, a few weeks after the first applications were upgraded, a memory leak caused a crash. The problem manifested itself under very specific conditions. Upgrades were temporarily put on hold and a fix was quickly provided and applied to the already upgraded applications. Intentional gaps in time, that were strategically inserted in the planning from the beginning, helped mitigate the impact of various unexpected delays on the overall schedule.

FUTURE IMPROVEMENTS

After the last application was upgraded we already had a good idea about future improvements.

The key to further reduction in the upgrade time is the automation of the remaining manual steps. This will also open the door to an interaction-less upgrade. The revert (start from scratch) operation should also be automated – in specific cases, when an upgrade needed to be repeated, this was done manually.

The most natural evolution of our tools would be to incorporate Ansible [6], an IT automation solution, which perfectly fits our needs. It provides, out of the box, the functionality of *wccauto* and much more. Rich features and plugins can make the process even more resilient to problems and scale better.

These advancements in the tooling should allow us to do offline upgrades, a method in which the process is performed outside of the production system. This method was previously proposed in [1]. The planned use of SCADA applications inside containers could give us similar opportunities.

We should also consider using a dedicated scheduling software, that, based on resource constraints, will automatically create/adjust the planning of the upgrades. This could be either an in-house developed or an existing commercial or open-source software, and it would be a great aid during the initial phase as well as later in the campaign, when changes are unavoidable.

CONCLUSION

The CERN SCADA systems upgrade in 2020 followed a steady incremental progress. Flexibility and adaptability in the planning were needed in order to accommodate the operational restrictions posed by the different domains.

The effort placed on automation paid off, by enabling more efficient, agile and error-free upgrades.

The success of the campaign was visible on the overall decreasing intervention times per upgrade and, in particular, on the less human resources needed. The improvements introduced in the tooling were essential to reach this achievement.

The experience gained during the previous campaign in 2017 constituted the base of this accomplishment.

REFERENCES

[1] R. Kulaga et al., "Large-Scale Upgrade Campaigns of SCADA Systems at CERN - Organisation, Tools and Lessons Learned", in Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17), Barcelona, Spain, Oct. 2017, pp. 1384-1388. doi:10.18429/JACoW-ICALEPCS2017-THPHA021

[2] Simatic WinCC Open Architecture (previously PVSS) SCADA Software from ETM (Siemens subsidiary), http://www.etm.at

[3] JCOP Framework, http://jcop.web.cern.ch

[4] UNICOS Framework, http://unicos.web.cern.ch

[5] Jira issue and project tracking software from Atlassian, https://www.atlassian.com/software/jira

[6] Ansible, https://www.ansible.com/