HOW TO SUCCESSFULLY RENOVATE A CONTROLS SYSTEM? - LESSONS LEARNED FROM THE RENOVATION OF THE CERN INJECTORS' CONTROLS SOFTWARE

Grzegorz Kruk, Stephane Deghaye, Olga Kulikova, Valery Lezhebokov, Marine Pace, Pablo Pera Mira, Eric Roux, Jakub Pawel Wozniak, CERN, Geneva, Switzerland

Abstract

Renovation of the control system of the CERN LHC injectors was initiated in 2007 in the scope of the Injector Controls Architecture (InCA) project. One of its main objectives was to homogenize the controls software across CERN accelerators and reuse as much as possible the existing modern sub-systems, such as the settings management used for the LHC. The project team created a platform that would permit coexistence and intercommunication between old and new components via a dedicated gateway, allowing a progressive replacement of the former. Dealing with a heterogeneous environment, with many diverse and interconnected modules, implemented using different technologies and programming languages, the team had to introduce all the modifications in the smoothest possible way, without causing machine downtime. After a brief description of the system architecture, the paper discusses the technical and non-technical sides of the renovation process such as validation and deployment methodology, operational applications and configuration tools characteristics and finally users' involvement and human aspects, outlining good decisions, pitfalls and lessons learned over the last five years.

INTRODUCTION

During the 80s and the 90s the high-level controls system used in the CERN Proton Synchrotron (PS) complex was based on a 2-tier architecture. Most of the Graphical User Interfaces (GUIs) were implemented in the C/C++ programming language using the X/Motif widget toolkit. The processes running on the front-end computers (FECs) were based on a framework called GM and communicated with higher layers via a custom RPC protocol, both developed in-house.

While being relatively simple, this solution had many drawbacks and limitations, for example lack of a subscriptions mechanism, making it necessary to pull data from the FECs, weak protection of the latter from the increasing number of clients and a very basic settings management.

Toward the end of the 90s, X/Motif was on the way to become obsolete and finding developers skilled in this technology was increasingly difficult. Work started on a new Controls Middleware (CMW) [1] library and on a new front-end framework called the Font-End Software Architecture (FESA) [2]. At the same time the decision was taken to implement the new high-level controls system using object-oriented methodology and the Java programming language. Work began to port the existing X/Motif applications to Java, replacing the legacy protocol with CMW in the hardware access layer. However, with most of the efforts focused on the LHC, no major architectural modifications were made, leaving the system with the long-standing issues described previously.

With growing maintenance costs and difficulties in introducing new functionality, in autumn 2007 a new project called Injectors Controls Architecture (InCA) [3] was mandated to homogenize the controls software across CERN accelerators.

INJECTOR CONTROLS ARCHITECTURE

InCA is a platform integrating specific applications developed for the LHC injector accelerators with modules implemented for the LHC, as well as new components required to fulfil the specific operational needs of the PS complex.

Architecture

InCA is based on a classical 3-tier architecture (Figure 1). At the bottom, there are the FECs, dedicated to the real-time control of the hardware, managed by three different frameworks: FESA, Function Generation Controller (FGC) [4], controlling the power converters and the legacy GM framework, being progressively replaced by FESA.

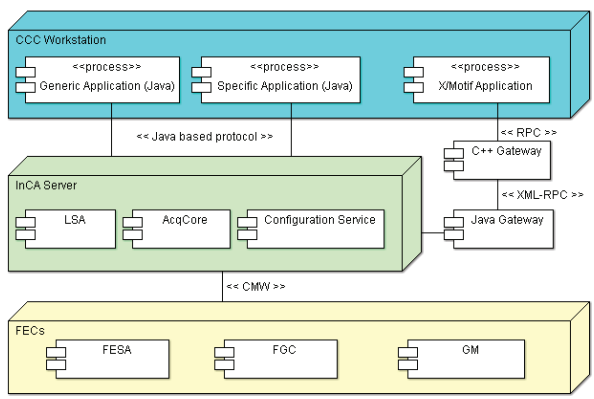

Figure 1: Main components of the Injectors Controls Architecture (InCA).

In the middle tier there are components providing high-level services. Among them the LHC Software Architecture (LSA) [5], responsible for the settings management, the Acquisition Core (AcqCore) responsible for the monitoring, processing and redistribution of

ISBN 978-3-95450-139-7

hardware values and the Configuration Service that provides efficient retrieval of configuration data.

Finally, in the top tier, there are client applications accessing the middle-tier services, consisting of generic applications provided by the InCA team that allow control and surveillance of all the equipment in a standard way, and many specific applications, developed by the operations crew, dedicated to a concrete type of equipment or operational scenario.

The overall architecture choice was correct but comprehensive performance tests showed the low-level libraries could not cope with the load that the AcqCore exerted while monitoring and republishing all hardware parameter values. Therefore we implemented *subscriptions on demand* – a mechanism that creates new subscriptions from InCA server to the FECs when requested for the first time and stops them when the last interested client application had been closed for a predefined amount of time.

Dealing with Legacy Applications

By the time of the first operational deployment of InCA in the PS machine in 2010, all the generic applications had been implemented in Java and integrated with InCA. There was however an important number of specific applications used operationally still implemented in X/Motif. As the migration of these applications to Java was not feasible in time for the operational deployment of InCA, two dedicated gateways were provided to allow integration between these applications and the InCA server, as seen in Figure 1.

Instead of directly sending new settings to the FECs, the RPC calls from these applications are redirected to a dedicated process (implemented in C++), which subsequently forwards the calls to a Java process using the XML-RPC protocol. The Java gateway calls the InCA server as any other Java client.

This solution has proven to be reliable. However due to the three additional hoops (two gateways and the InCA server), the interaction with the FECs became an order of magnitude slower, with possible delays up to a few seconds. Despite these delays, we decided to not invest additional time on optimizations due to the tight deadlines to complete crucial features before the first operational deployment. A positive side effect is the incentive it has given the operations crew to rapidly renovate these applications in Java.

DEVELOPMENT PROCESS

To properly manage such a large project we needed a structured methodology. First we studied the Rational Unified Process (RUP) but we concluded that it was too heavy for our needs. We looked then into agile methodologies and settled on Scrum [6], which gave structure to the development process while being lightweight.

Each four-week development cycle, shown in Figure 2, ended by a demo meeting where the new features were presented in front of all developers and representatives of the operations crew.

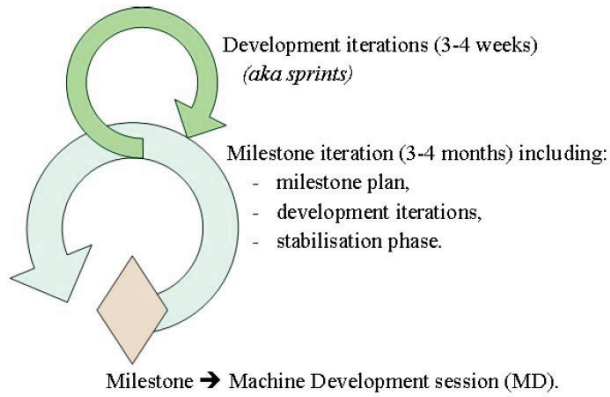

Figure 2: InCA development cycle.

Although this methodology has many positive elements, with time we realized that it was not ideal to our environment.

What worked well for the InCA team were the planning meetings, organized at the beginning of each iteration, allowing all the developers to have an overview of the features that would be worked on next. The iteration meetings, held twice a week, improved knowledge sharing, allowing close follow-up of the progress and a more efficient resolution of many issues arising during development. Also the demo meeting, being a small milestone, played a meaningful role in motivating the team to complete the planned work on time.

On the other hand, for the Scrum methodology to work well, all members of the team need to be relatively easily interchangeable i.e. all developers know and can work on all parts of the project. Due to different levels of knowledge about existing components and different areas of expertise among the InCA developers, several Scrum principles could not be applied properly. For example it was difficult to fully engage participants during the planning and demo meeting, when items outside of their core responsibilities were discussed. In addition, support issues and activities related to other projects that some of the developers were involved in, heavily interfered with planned tasks. This required the developer to often switch context and meant a change of priorities for features foreseen for the iteration.

After the first deployment of InCA in the PS machine, we started to adjust the development process into a form of Scrum-ban [7], i.e. a mixture of the Scrum and Kanban [8] methodologies. This is more suitable for maintenance projects with frequent and unexpected user requests and support issues.

DEPLOYMENT METHODOLOGY

To prepare for the first operational deployment in the PS, every 3-4 months, we organized dedicated Machine Development (MD) sessions. During these one-day sessions, InCA was deployed in a full scale on the

ISBN 978-3-95450-139-7

operational accelerator. The goal of these sessions was to validate a set of features in the operational environment.

The tests were carried out by both the operations crew, performing functional tests according to prepared scenarios, and the InCA team, doing detailed checks of generic applications and executing non-functional tests such as verifying the performance and scalability of the system. All the problems spotted during these sessions were noted down and fixed before the next MD day.

The MD sessions played a key role in validating the overall system. They also allowed the operations crew to gain confidence in the new system before the operational deployment. But even though they were carefully planned, due to their limited duration it was difficult to test all possible use cases, considering different types of beams and diverse groups of users. In addition we focused on operational tools and scenarios, giving less attention to specialist applications such as those used by the Radio Frequency (RF) experts. As a consequence we experienced some problems within the first weeks after the operational deployment that could have been avoided. These problems were fortunately not critical and could be quickly resolved.

Before the final deployment we also organized several training sessions to familiarize the users with the new system and to train them with the new set of tools.

We have applied the same strategy for all subsequent InCA deployments on the other accelerators, adjusting the procedure according to the feedback from the previous sessions.

InCA Mode

Even with several testing sessions in the operational environment, due to the importance of the system, we had to be prepared for unforeseen critical problems that could block operation for a significant period of time. To mitigate such risks, we designed and implemented the InCA client libraries in a way which allowed to quickly disable the use of the InCA services and to switch back to a non-InCA mode in which the applications worked as they did before the operational deployment of InCA.

To bypass the InCA server, it was sufficient to modify a dedicated JVM property or an environment variable and restart the Java or X/Motif application. In addition, using a single configuration file kept in a network location, we were able to toggle the InCA mode globally for all applications.

The global switch has never been used, however the local ones turned out to be very useful for diagnostics as they allowed comparing behaviour with and without the InCA server involvement.

OPERATIONAL SUPPORT

InCA is a critical system used 24/7 to control most of the accelerator's equipment. More serious problems could stop operation and delivery of the beam to various experiments and to the LHC. Therefore it was essential to put in place a reactive support. This was especially important within the first months after the operational deployment. We decided to involve all InCA developers in the support to avoid the same people to be called in systematically. The support was organized in weekly shifts.

Each week, one member of the team is responsible for the diagnostics and resolution of all problems, playing the role of a *front person*. In case he is not able to diagnose or solve the problem by himself, he redirects the issue to the appropriate developer and ensures a proper follow up. At the end of each week, a support meeting takes place with all the developers and some of the user representatives, where the last 7 days' issues are discussed and explained to the whole team.

Thanks to this organization, most of the issues are handled directly by the support person, offloading time from the other developers and minimizing the number of interruptions they would be exposed to otherwise. The support meeting improves the knowledge sharing and decreases the diagnostic time in case of similar issues appearing in the future.

One area where we could have improved is the training of the participating developers. A more thorough training would have given everyone a more detailed knowledge of the components and layers of the system, especially those they were not directly involved with. Without this, sometimes the support person could not even perform the initial diagnostics without the involvement of the responsible developer.

GRAPHICAL USER INTERFACES

The greatest control system, providing rich functionality and being fast and reliable, will not be successful without good GUIs.

Users perceive the quality of the overall system through the graphical tools that they use in their daily work. These tools must not only be free of bugs, but also intuitive and easy to use for the occasional and advanced users. If this is not the case, instead of being helpful they might become a source of frustration or even a cause of operational errors.

Proliferation of Applications

One significant source of issues was the number of different applications used to perform settings-related operations e.g. to initialize, change, copy or rollback settings. Historically different developers implemented them at different moments in time, having SPS and LHC requirements in mind and not covering the LHC injectors' needs. Other tools existed in the PS complex in the pre-InCA times and were only slightly adapted to use InCA for settings management. Because of this situation, the operations crew was sometimes confused about which application should be used to perform a given task.

When this problem became apparent, work started on a single and coherent settings management tool, covering the requirements of the operations crew of all the concerned accelerators. Successive versions of this tool were deployed into production in 2011 and 2012, replacing progressively the existing applications.

ISBN 978-3-95450-139-7

Functionality of the remaining applications will be included in a new version planned for early 2014.

Complexity and Ergonomics

Another source of trouble was the complexity of the tools. Some of them, such as the generic *Function Editor*, provide very rich functionality, starting from basic operations to sophisticated, expert-oriented options. Developing such tools, with requirements coming from different accelerators and users, turned out to be much more challenging than we initially assumed. The main difficulty was not the implementation but the visual design, the flow between various views and the way different options were presented. With the initial version of the *Function Editor*, many users felt lost in the number of options, not knowing how to perform the simplest operations. Even though we provided a comprehensive help documentation available directly in the application, most of the users preferred a more intuitive GUI with small contextual help tips.

We realized when reviewing this tool, and also when designing other applications, that we needed to stay in close contact with the users. To show the users how each aspect would look like and getting feedback from them before starting the real implementation, we used Balsamiq Mockups [9], a rapid wire-framing tool that allows easy creation of graphical sketches reflecting the GUI to be implemented.

The usage of this tool facilitated discussions with users and speeded up iterations until a satisfactory design of the GUI was found.

Configuration Tools

Many operational aspects of the existing control system required proper configuration in the database. It was agreed that the operations crew would take this responsibility over. With InCA, many new features were introduced, requiring additional configuration, making this task more complex. With the main priority put on providing the necessary functionality in the operational applications, the importance of appropriate configuration tools was neglected. The existing tools were not adequate and contained a mixture of basic and advanced options. As they started to be used regularly by the operations crew, the number of wrong configurations started to increase, contributing to about 30% of all reported issues.

To resolve this problem we decided to completely review the configuration tools. The goal was to bring the number of existing options to a minimum, by automating configuration tasks or using reasonable default values, and to make a clear distinction between the available (visible) options to regular users and to experts.

The redesign has been completed in 2013 and new configuration tools will be available to the operations crew after the Long Shutdown in 2014.

HUMAN ASPECTS

An important aspect of the renovation process was the acceptance of the system by the users' community. The first operational deployment of InCA confronted substantial resistance from the operations crew, for several reasons.

One reason was the fact that homogenization meant moving from tools tailored to the needs of the individual accelerators towards more generic applications. In addition, InCA introduced several new concepts compared to what already existed and changed slightly the way the existing functionality could be used. The result of these changes was that the operations crew had to get accustomed to a new set of tools and to certain extend also had to change their habits.

Other reasons were some missing functionality, still under development, and teething problems in the new tools, which lowered the trust in the system.

One of the key factors in rebuilding the confidence was to maintain a close contact with all groups of users. Reactive follow up of issues, a continuous presence in the control room, listening and understanding individual requirements and explaining any difficulties in implementing them helped in increasing the bidirectional understanding and facilitated the acceptance of the new system by the users.

CONCLUSIONS

We successfully renovated the control system, homogenizing it across the whole CERN accelerators complex while respecting the specificities of individual accelerators and user groups.

Facing a mixed reception of InCA by the users after the first deployment, we significantly improved all the aspects of the system during the last three years, progressively gaining their trust. Many more improvements are being developed now to be ready for restart of all accelerators in 2014.

Since the PS deployment in 2010, InCA has been deployed in the Booster and Linac2 in 2011, in SPS and ISOLDE in 2012 and preparation is well on track for the deployment in 2014 on the two remaining machines: AD and CTF3.

REFERENCES

[1] K. Kostro et al., "The Controls Middleware (CMW) at CERN", ICALEPCS'03, Gyeongju, Korea
[2] M.Arruat et al., "Front-End Software Architecture", ICALEPCS'07, Knoxville, Tennessee, U.S.A.
[3] S. Deghaye et al., "CERN Proton Synchrotron Complex High-Level Controls Renovation", ICALEPCS'09, Kobe, Japan
[4] Q. King et al., "Evolution of the CERN Power Converter Function Generator/Controller for Operation in Fast Cycling Accelerators", ICALEPCS'11, Grenoble, France
[5] G.Kruk et al., "LHC Software Architecture (LSA) – Evolution Toward LHC Beam Commissioning", ICALEPCS'07, Knoxville, Tennessee, U.S.A.
[6] en.wikipedia.org/wiki/Scrum_(development)
[7] en.wikipedia.org/wiki/Scrum_(development)#Scrum-ban
[8] en.wikipedia.org/wiki/Kanban_(development)
[9] http://balsamiq.com

ISBN 978-3-95450-139-7