Renovation of the CERN Controls Configuration Service

Lukasz Burdzanowski, Chris Roderick
CERN, Geneva, Switzerland

ICALEPCS 2015

CERN Controls Configuration Service

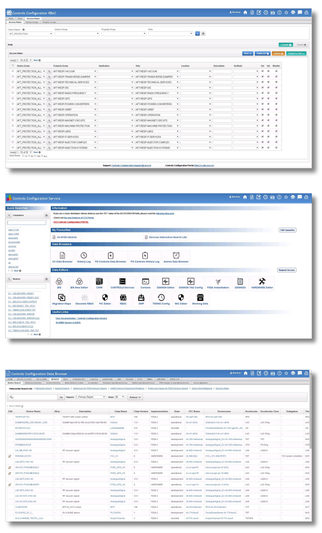
The CCS exists for more than 30 years, during which the scope, architecture, implementation technology and development methodology have kept evolving.

The criticality of the service for safe operation of the accelerators chain is high (though not required for their safe shutdown): The CCS is essential for proper accelerator configuration and start-up - especially during Technical Stops when equipment and other components of the Controls System undergo maintenance and upgrades.

database oriented architecture

Graphical User Interfaces

Numerous GUIs based on proprietary Oracle technologies: Application Development Framework (ADF) and Oracle Application Express (APEX).



high-level client Java APIs

database level client APIs

PL/SQL programmatic access used from both high-level application and for database-to-database intergration

Oracle database (RAC cluster)

The database is implemented using a relation model, with approximately 700 domain tables and ~7GB of core domain data (excluding binary, log and history data which collectively accounts for ~115GB)

data-driven multi-layer infrastructure

CERN Controls System

high-level software

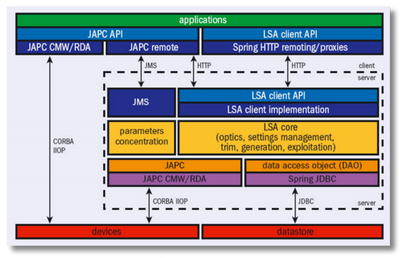
e.g.: high-level settings management, data acquisition and archiving

middleware layer

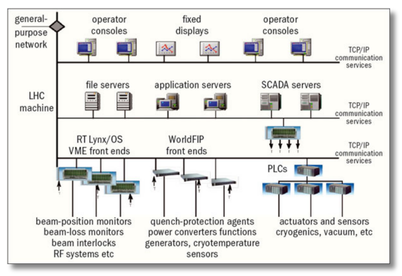
e.g.: read/write access to processes running on FECs and Role Based Access Control (RBAC)

low-level components

hardware and software, e.g.: timing infrastructure, equipment drivers, Front-End Computers (FEC), end-user developed C/C++ binaries representing operational “devices”



architecture of the Controls system as seen from the perspective of the high-level applications



overview of the Control system network topology

Brief history of the service

The scope of the CCS was initially limited to the PS (Proton-Synchrotron) complex controls system, meaning that the service and its database were oriented towards a concrete accelerator and its specific control system.

The first relational database was introduced in 1986. Over the years the scope grew following the evolution of CERN’s accelerator complex. 1995 marks the introduction of graphical user interfaces (GUI) based on Oracle Forms and PL/SQL Web Toolkit (OWA).

The first Java based data access API was implemented in 1999 facilitating access for high-level applications. Starting in 2006, another Oracle based GUI solution (ADF - a Java Server Faces implementation) was put in place to replace existing OWA and Forms applications. In 2009, APEX (a subsequent framework for building database-driven GUI’s) was adapted alongside ADF.

Renovation Strategy

In the middle of 2014 the first major service-wide renovation and overhaul has started - marking the beginning of a new chapter in its long history.

the corner stones

Suppression of the accumulated technical debt
Changes in the overall architecture
Adaptation of the Lean software development process

in summary

All of renovation aspects are closely related as suppression of technical debt is **essential** in order to **advance** the system **architecture**, while taking proper architectural and design decisions prevent further “erosion” in the system and limit existing technical debt.

The adapted software **development process** facilitates implementing changes: enabling a **lower overall cost** of development and **increased agility**.

The first two aspects are a mid-to-long-term perspective. The implementation of the Kanban is already well advanced and can be considered finished by the end of 2015.

Kanban

The Kanban emphasizes continuous improvement, importance of human factors and bringing maximum value to the organisation.

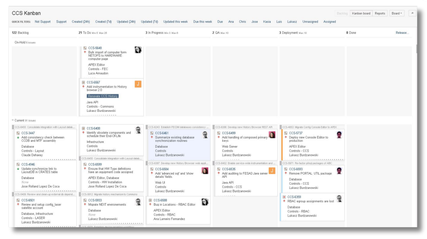
it is all about eliminating wastes

limit excesive context switching

- too much work in progress
- too many unrelated tasks started
- too many new features waiting in quality assurance queue

visualize your work

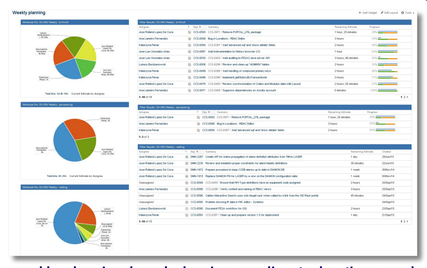
By visualising the work on a Kanban board, bottlenecks were quickly identified. Focus and effort need to control flow of work has been noticeably reduced.



CCS Kanban board showing current tasks: backlog, to-do, in progress, QA, deployed, done and more...

be pragmatic

By not relying on fixed development iterations or sprints the trust from end-users increased as their requested features and bug-fixes are not systematically subjected to prolonged wait times due to extensively planned ahead sprints.



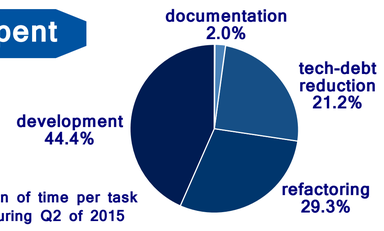
weekly planning board showing pending tasks, time needed to complete the tasks, remaining capacity

work in a team

The agility and reactivity of the team and CCS as a whole has increased. Thanks to the Kanban / Lean philosophy of just-in-time delivery every team-member can use her or his potential focusing on activities that bring the most value to end-users increasing their satisfcation.

know where time is spent

With weekly retrospectives, monthly summaries and quarterly reviews we keep track of spend time per activity, domain or issue type.



Distrubution of time per task qualifier during Q2 of 2015

in summary

By changing the way tasks are prioritized and visualized has led to a **reduction in pressure and stress** on developers.

CCS **end-users** are now much more closely **involved** in the development process and act as true stakeholders thanks to effective visualization of work in progress and clearly identified stages of the development cycle. These **human factors** are proving to be essential to the success of the on-going renovation.

Regular **retrospectives** and critical analysis of changes applied to the working process have **positively transformed** the way the CCS team works.

Architecting for the future

Renovation and supporting changes in system architecture fall into four main categories. All of them are closely related, define bounderies and shape the renovation.

context based access to the data

By attaching state information to core domain entities in the system (e.g. devices), we can now automatically notify users interested in a given “domain event”.
The high-level domain specific events, i.e.: FEC Renamed, give the users an opporutnity to subsribe and track changes to these entites which are particulary important.Workflow based transition of data in the system help and guide users while limiting potential errors.

phasing-out of proprietary GUI technologies

On average per day there are over 150 distinct user sessions (from a total of ~400 distinct registered users)

By moving to widely adapted solutions of Java based RESTfull services and HTML5/JavaScript web interfaces we adapt the technology stack which steadily gains popularity within the software engineering community and in turn facilities hiring of well-trained specialists.

system-wide tracing, monitoring, auditing

With tracking and auditing the time and/or user behind a given action is captured along contextual information like client IP address, database session and transaction IDs, name of the program unit and invoked action.

The context information is used to augment historical data tracing which gives insight to a concrete action which was invoked by the user. Stack of actions is captured as well making it possible to follow user actions in order to better understand a problem or to optimize existing worklflows.The instrumentation considerably limits the time needed to support users in investigating suspected data problems, and potentially recovering data.

lowering system complexity

During the process of suppressing accidental complexity / lowering overall complexity we have started to progressively adapt the event driven architecture.

New developments and on-going re-factoring conforms to GRASP (General Responsibility Assignment Software Patterns) patterns of Object-Oriented design, tailored to the world of relational databases.

To support these changes we have adapted **Commons4Oracle** (C4O)

Commons4Oracle

Is a set of PL/SQL libraries for Oracle database, which is actively developed in the CERN Controls group.
The library assures further standardization and foundations for future development and streamlines solutions in the CCS with other core database projects of the group thus enabling transfer of knowledge and expertise.

Addressing technical-debt

In most cases, end users are not directly aware of the technical debt but as software engineers we should perceive it as negative value. It is adverse to system architecture and design, which are planned, deliberate and visionary.

targeted re-factoring

is predefined as a concrete group of tasks based on the following criteria

1. **Identify boundarie** - to clearly know when the activity should finish.
2. **Identify clear gains** - to justify the effort. The gains should be tangible, based on facts.
3. **Identify risks** - to know the impact both within and outside the service.
4. **Define rollback / fall-back strategy** - to limit any potential negative impact, mainly in critical areas.
5. **Estimate and prioritize** - to realistically plan the effort alongside regular activities.

the value to be gained from the re-factoring can be classified into distinct areas

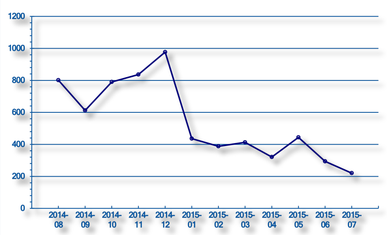
1. **Consistency** - i.e. limiting the likelihood of data corruption and/or of non-deterministic states.
2. **Performance** - improving the response times for data reporting and querying for clients.
3. **Maintenance** - lowering: the total cost of development, likelihood of introducing new errors, and the usage cost paid by clients (e.g. by obscurity APIs or lack of documentation).
4. **Agility** - ensuring the extendibility of the architecture and limiting the cost / time of delivering new features to clients.

Static Code Analysis with Commons4Oracle

The static code analysis (SCA) is the analysis of computer software source code on the contrary to dynamic analysis, which is based on code execution.

Commons4Oracle provide a custom SCA framework which includes a pre-defined set of analysis rules, which can be customized and extended

The analysis results, fluctuations and evolution of the metrics are the inputs to qualitative assessments and serve as a basis for future planning.generates reports summarizing the number of rule violations, severity and links to the source. The reports are used to identify areas for in-depth analysis and planning of the re-factoring.



With SCA in place we are able to evaluate our efforts over time, relying on factual data rather than assumptions.

Example chart showing count of invalid objects in development database, aggregated per months.
Resul of database schema analysis over one year.

Abstract

The **Controls Configuration Service** (CCS) is a **key component** in CERN’s data driven **accelerator Control System**. Based around a central database, the service also provides a range of client APIs and user interfaces - enabling configuration of controls for CERN’s accelerator complex. The service has **existed for 35 years** (29 based on Oracle DBMS).

To cater for changing requirements and technology advances there has been **substantial evolution** of the CCS over time. Inevitably this has led to increases in CCS complexity and an accumulation of technical debt. These two aspects combined have a negative impact on the flexibility and maintainability of the CCS, leading to a potential bottleneck for Control System evolution.

This paper describes on-going renovation efforts (started mid-2014) to tackle the aforementioned issues, whilst ensuring overall system stability. In particular, this paper covers **architectural changes**, the **agile development process** in place - bringing users close to the development cycle, and the **deterministic approach** used to treat technical debt. Collectively these efforts are leading towards a **successful renovation** of a core element of the Control System.

Conclusions

The renovation of a mission critical service with many years of history is a challenge. Alongside changing requirements, growing expectations and needs to **consolidate** various sub-systems of the Control System, the CCS started to play an even more important role during recent years. The necessity to adapt to these changes and satisfy new requirements is the driver for the on-going CCS renovation.

Progressively **reducing technical debt** increases overall **agility**, but more importantly it also helps to design a better system for the **future**. CCS users now have a much better understanding than previously of the value of these changes and together with their increased satisfaction - renovation and technical debt reduction is perceived as **added value**.

The **Kanban** way noticeably **improved** the CCS team efficiency and contributed to increased **end-user satisfaction**. New architecture solutions lay foundations for an advanced, cohesive and agile system that embraces the context and workflows of how CCS users work. The renovation started over a year ago marked the beginning of a new and **exciting era** in the long history of the Controls Configuration Service of the CERN Controls system.

References

[1] J. Cuperus et al., ICALEPCS1997 - ID085, [2] R. Gorbonosov, The Control Systems of the Large Hadron Collider, CERN Academic Training Lecture Regular Program, http://cds.cern.ch/, [3] J. Cuperus et al., ICALEPCS2003 - WE114, [4] M. Arruat et al., ICALEPCS2007 - WOPA04,
[5] G. Kruk et al., ICALEPCS2013 - MOCOBAB05[6] MM. Lehman, Laws of Software Evolution Revisited, EWSPT '96., [7] C. Larman, Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd ed.), ISBN 0-13-148906-2, Prentice Hall, (2005) [2004]
[8] T. Ohno, Toyota Production System: Beyond Large-Scale Production, ISBN 978-0-915299-14-0, Productivity Press, (1998), [9] H. Kniberg, Lean from the Trenches: Managing Large-Scale Projects with Kanban (1st ed.), ISBN 978-1934356852, Pragmatic Bookshelf, (2011)