IMPROVING CODE QUALITY OF THE COMPACT MUON SOLENOID ELECTROMAGNETIC CALORIMETER CONTROL SOFTWARE TO INCREASE SYSTEM MAINTAINABILITY*

O. Holme, D. Di Calafiori, G. Dissertori, L. Djambazov, W. Lustermann, ETH Zurich, Switzerland
S. Zelepoukine, ETH Zurich, Switzerland and University of Wisconsin-Madison, U.S.A.

Abstract

The Detector Control System (DCS) software of the Electromagnetic Calorimeter (ECAL) of the Compact Muon Solenoid (CMS) experiment at CERN is designed primarily to enable safe and efficient operation of the detector during Large Hadron Collider (LHC) data-taking periods. Through a manual analysis of the code and the adoption of ConQAT [1], a software quality assessment toolkit, the CMS ECAL DCS team has made significant progress in reducing complexity and improving code quality, with observable results in terms of a reduction in the effort dedicated to software maintenance. This paper explains the methodology followed, including the motivation to adopt ConQAT, the specific details of how this toolkit was used and the outcomes that have been achieved.

INTRODUCTION

The CMS ECAL DCS monitors all the relevant detector environment conditions and handles the control and monitoring of the powering systems that feed the readout electronics and provide bias voltage to the sensing elements of the detector [2]. In addition, the control and monitoring of several external applications are integrated, such as the CMS ECAL safety and cooling systems which rely on Programmable Logic Controllers (PLCs). The DCS software layer is implemented with the flexible and extensible SIMATIC WinCC Open Architecture (WinCC OA) supervisory control software [3] from ETM professional control and makes use of the JCOP Framework (JCOP FW) [4] that is developed by CERN in collaboration with the LHC experiments.

The software was developed over a period of several years with coding contributions from nine people during the development phase. In order to distribute the work amongst the developers, the software was fragmented into a series of sub-applications according to the type of data to be handled and the region of the detector for which it was designed. Examples of sub-applications include bias voltage control, low voltage control, temperature and humidity monitoring, cooling monitoring and safety system interfacing. Twelve independent components were produced and combined to form the complete DCS.

The common technology basis of WinCC OA and the JCOP FW, combined with a small number of development guidelines, ensured that the complexity of integrating the individual components was kept to a minimum. This enabled the system to be delivered on time for the start of the LHC physics program in 2009.

*Work supported by the Swiss National Science Foundation

MAINTENANCE PHASE

Following the start of detector operations, the CMS ECAL DCS project moved into the maintenance phase. In order to support the system with a reduced team of three people, a significant consolidation of the software was required. A summary of the efforts to consolidate the DCS have been reported previously [5].

A key part of the consolidation phase was a detailed analysis of the existing code base. The results of this process indicated that the openness and extensibility of WinCC OA enabled developers to implement each component using very different fundamental principles, creating difficulties for the reduced team to maintain a detailed understanding of the complete DCS software layer. Additionally, the independent development of components led to cases where similar or identical functionality was implemented multiple times.

To reduce the overall code base size, efforts were made to homogenise the components and remove redundant functionality implementations. Initially it was easy to identify areas of code to target and improve, leading towards the goal of a cleaner and smaller DCS software implementation. Although progress was made, it was not possible to quantify the improvements, nor to estimate how much more effort was required.

During this initial phase of code consolidation, developers reported seeing similar blocks of code in the files that they were working with. These duplicated code segments could easily be removed through re-factoring of the software, but it was notable that this duplication was found by chance and that other duplicates could easily have remained hidden.

In order to obtain a clear snapshot of the current code status and to track progress with time, an automatically evaluated set of metrics was required. The issue of code duplication was one of the main motivations for adopting automatic software quality assessment. The ConQAT toolkit was selected for its modern, powerful static analysis tool that can detect blocks of duplicated code known as *code clones* [6]. An additional benefit was the graphical summaries of quality metrics, which enable rapid assessment of overall results and the identification of problematic areas to target.

CONQAT CONFIGURATION

ConQAT is an extensible, open source quality assessment toolkit that is delivered with a set of static analysis tools for assessing many code quality metrics.

ISBN 978-3-95450-139-7

Workflows, saved into individual configuration files, can be designed in the Eclipse integrated development environment [7], allowing a fully customised quality assessment procedure to be defined.

A workflow for the CMS ECAL DCS software was defined that firstly selects the subset of files that contain code from a software source directory. Then, the following metrics are evaluated:

- Number of files;
- Files with functions that exceed a given length;
- Files with nested blocks that exceed a given nesting depth;
- Files that include code clones;
- Lines of code (LOC);
- Source lines of code (SLOC) excluding lines containing only white space and comments;
- Number of executable source statements (SS);
- Comment to LOC ratio;
- Number of cloned SS and cloned LOC;
- Redundancy-free source statements (RFSS). The concept of RFSS is defined as an estimate of the number of executable SS that would remain in the code if all clones were completely removed [8].

As a final step, the workflow defines how the results should be stored and presented. For the CMS ECAL DCS, the results are displayed in an HTML report, containing tabulated results and interactive graphics. Figure 1 shows an example display representing each source file as a rectangle whose size depends on the SLOC in that file. In this case, the colour scale of white to red reflects the percentage of lines in the file that have been detected as clones, with white shapes having no duplicated code and red shapes having the largest proportion of cloned code.

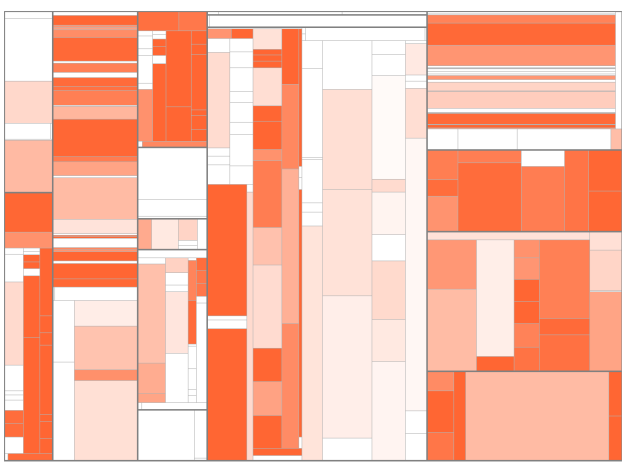

Figure 1: A ConQAT graphic showing the prevalence of code cloning in the software as it was in October 2011.

INTEGRATING CONQAT IN WINCC OA

WinCC OA is programmed primarily through a C-style scripting language called CTRL. Due to the syntactical similarity with C, ConQAT was able to analyse the CTRL code without any need for customisation. Complexity arises from the way in which WinCC OA stores code.

CTRL code can be stored in libraries of functions or standalone scripts, in which case the code is stored simply as plain text. It can also be saved in graphical user interface (GUI) definition files, with escape sequences to represent special characters such as the double quote. GUI files contain a combination of CTRL code and data that represent how graphical widgets should appear on screen. This mix of code and data can produce false positives for duplication checking software as the widget configuration data is similar in structure for many graphical objects. Furthermore, the JCOP Framework saves CTRL code into WinCC OA persistent storage objects, called *data points*. These data points can be exported from WinCC OA in a custom text file format for tasks such as transferring the objects between applications.

To isolate the code from the GUI definition files, the WinCC OA feature of converting all GUI files to an Extensible Markup Language (XML) format was used. In the XML version, all code is stored under `<script>` tags in CDATA (unparsed character data) format to avoid the need for escape sequences. With a simple XML parser, the code can then be extracted and saved to another file which only contains the CTRL code.

The extraction of CTRL from data point export files requires specific prior knowledge about the data point naming used by the JCOP FW. This information is used to search the file to identify and isolate the code from the rest of the data point information.

A New WinCC OA Component

In order to perform the CTRL code extraction and other preparatory actions, a new component was developed in WinCC OA. This tool allows a developer to manually trigger a quality assessment procedure and to choose the required quality metric thresholds. The tool takes the source code from a specified directory, for instance a checkout of a Subversion repository [9], and moves the files into a temporary workspace where WinCC OA can perform the XML conversion of the GUI files. Once the CTRL code has been extracted, the ConQAT configuration file is generated by merging a template workflow with the specified threshold values. The final action is then to trigger ConQAT to perform the analysis and report to the user when the execution is complete.

This tool has been built in accordance with the guidelines for standard JCOP FW components and, as mentioned earlier, it also searches for specific JCOP FW named objects in data point export files. These design choices do not limit its use exclusively to JCOP FW users, so any WinCC OA developer could benefit from this integration tool to analyse code from GUI panels, CTRL libraries and standalone scripts.

PRACTICAL EXPERIENCE

The initial adoption of ConQAT required not only the design of the workflow and choice of quality metrics as described earlier, but also the choice of the acceptable thresholds for certain metrics. Thresholds were needed to define the maximum tolerated function length and nested

ISBN 978-3-95450-139-7

block depth, which were chosen as 60 lines and 3 levels respectively. In addition, the minimum length of code clones had to be selected to avoid detecting similarities between very small fragments of code. This threshold was set at 15 lines. These threshold values were chosen empirically as a trade-off between the comprehensiveness of the analysis results and the impact of overwhelming developers with numerous issues of lesser importance.

ConQAT provided the DCS developers with the first quantitative assessment of the software quality, leading to three important findings:

- Code cloning affected a significant proportion of files in the CMS ECAL DCS project;
- There was a significant correlation between the files with long blocks and the files with a high degree of nested blocks;
- There was no correlation between files with long blocks and files with significant cloning.

With this new knowledge the code could be inspected to see the severity and impact of the problems that were highlighted with the chosen metrics. Firstly, it was noted that the code-comment ratio was not helpful for discovering real quality issues. The analysis is not able to distinguish between useful comments and situations where old, redundant code has been commented out. This reduces the effectiveness of the comment-to-LOC ratio as an indication of software maintainability.

The findings related to long blocks and nested code were always significant and highlighted complex pieces of code. Such complex code can be difficult to maintain in the long term as it can take a long time to understand its functionality and the impact of any changes.

The clone detection was also very accurate in finding similar sections of code throughout the whole project. By reducing the code cloning, the size of the CMS ECAL DCS code base could be significantly reduced, making it easier for developers to keep an overview of the complete project. By removing duplicated implementations, the risk of software faults caused by inconsistent evolution of clones [8] is also reduced. For these reasons, the issue of code cloning was chosen as the primary focus for consolidation work, in order to make the most progress in reducing the required software maintenance effort.

Initially, the ConQAT evaluation process was executed manually by developers to enable an on-demand evaluation of the software code quality metrics. Due to positive early experiences with the tool, it was decided to integrate ConQAT more closely into the regular development processes. An automated nightly analysis system was implemented in order to provide regular feedback and to continue to provide quality monitoring even if developers were not specifically focussed on removing legacy code quality issues.

CONSOLIDATION PROGRESS

By periodically running the quality analysis, the progression of the quality metrics over time can be tracked. In addition, using the history of the code in a Subversion repository enabled the retrospective evaluation of metrics before the adoption of ConQAT.

The progress in code clone elimination is shown in Figure 2. As shown, the number of cloned SS was reducing before the introduction of ConQAT. This indicates that the manual inspection method was partially effective at finding and removing code duplication.

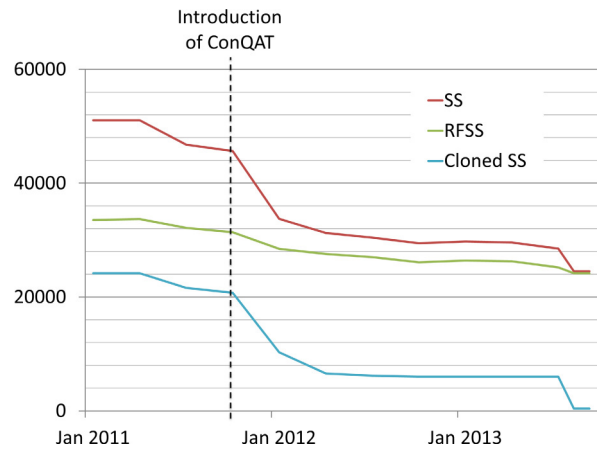

Figure 2: Progression of source statement and cloned code metrics before and after introducing ConQAT.

The time required to fix a clone is the sum of the time to analyse the problem, modify the code and then test the changes. In particular, the testing took a significant proportion of time as no automated testing was available. The consolidation was an on-going background task, but there were two periods in which developers could focus on this work, in late 2011 and summer 2013, as shown by the sharp steps in the reduction of cloned SS. At other times, tasks such as implementing new features were of the highest priority. It is significant that during such periods, late 2012 for instance, the cloned SS do not increase even though the total number of SS do increase slightly. This demonstrates that the newly written code did not include clones, indicating that continuous quality assessment was beneficial to ensure that new code met the required quality standards.

Another significant finding is that the RFSS has decreased almost continuously throughout the consolidation phase, indicating that the number of statements not affected by code duplication has also been reduced. This can be explained by the approach taken to deal with the code clones indicated in the quality analysis. The simplest way to address the issue is to focus purely on the affected lines and to make the smallest change necessary to remove the cloning. Alternatively, by looking at the implementation reasons leading to the code clones, it is possible to re-factor a larger portion of code which not only removes the duplicated lines, but can also reduce the number of redundancy-free lines needed to achieve the same functionality. By adopting this second approach, the size of the CMS ECAL DCS software was reduced by more than would have been achieved by targeting only the cloned code.

ISBN 978-3-95450-139-7

The current status of the CMS ECAL DCS code base is that the code cloning issue has been minimized. There remain around 392 cloned SS spread amongst 12 files. Each of these cases has been analysed and they typically involve very simple, repetitive pieces of code. To address these issues, the code would need to be made more complex and it has therefore been decided that there is no benefit to remove them. To visualise the progress made since the introduction of ConQAT in October 2011, the current status of code cloning in Figure 3 can be contrasted with Figure 1. The small number of findings that cannot easily be resolved demonstrates that the significant majority of clones detected by ConQAT were relevant and important to guide development efforts in consolidation of the code.

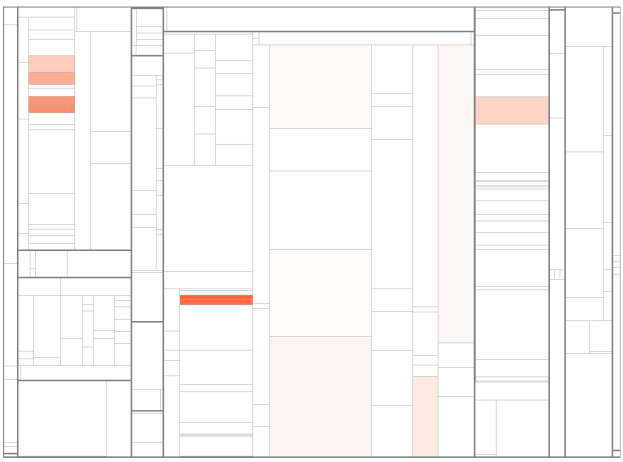

Figure 3: A ConQAT graphic showing the current status of code cloning in the software.

The quality metrics of long functions and deeply nested blocks were not the primary focus of the consolidation effort. However, developers have dedicated some time to reduce these issues, particularly in code that was re-factored due to code cloning issues. The progression of the number of files affected by the different quality metrics is shown in Figure 4. The figure also shows the total number of files in the CMS ECAL DCS software project. While the total number of files has remained almost constant since early 2012, the number affected by quality issues has been reduced.

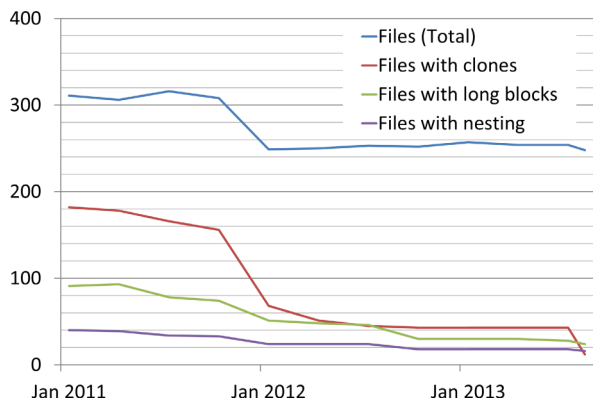

Figure 4: Progression of total number of files and files affected by different types of quality defect.

The code cloning problems have now been fully addressed. The remaining quality metrics will be the focus of the consolidation task in order to improve the readability of the code.

CONCLUSION

A manual inspection of the CMS ECAL DCS software raised awareness of quality issues within the code base. Preliminary work to resolve these problems was successful but a detailed, automatic analysis of the code was required to accurately identify the location of all quality issues.

By adopting ConQAT and building a tool to integrate it with WinCC OA, a regular analysis of the software could be performed. A set of quality metrics with suitable thresholds were defined in order to direct the developers to the problematic areas in the code.

Code cloning was targeted as the main focus to achieve a smaller code base that would be easier to maintain. The cloned SS were reduced from around 20,000 to 392 since ConQAT was introduced in October 2011. In the same period, the total SS has been reduced from around 45,000 down to 24,000 lines. These reductions are mostly due to removal of duplicated code with a contribution from re-factoring of larger code areas around cloned segments.

ConQAT has been successfully integrated into the working procedures of the CMS ECAL DCS. It has been used to direct efforts in improving the quality of existing code and, by offering continuous quality monitoring, it helps to sustain the quality of the software project when new features are being implemented.

REFERENCES

[1] ConQAT, Continuous Quality Assessment Toolkit; https://www.conqat.org/
[2] D. Di Calafiori et al., "Maintaining and improving the control and safety systems for the Electromagnetic Calorimeter of the CMS experiment," J. Phys.: Conf. Ser. 396, 012016 (2012).
[3] ETM professional control GmbH – WinCC Open Architecture – SCADA System, http://www.etm.at/
[4] O. Holme et al., "The JCOP Framework," ICALEPCS'05, Geneva, Switzerland, October 2005.
[5] O. Holme et al., "Maintaining an effective and efficient control system for the Electromagnetic Calorimeter of the Compact Muon Solenoid experiment during Long-Term Operations of CERN's Large Hadron Collider," PCaPAC'12, Kolkata, India, December 2012, FRCB01.
[6] E. Juergens et al., "CloneDetective – a workbench for clone detection research," ICSE'09, Vancouver, Canada, May 2009.
[7] Eclipse; http://www.eclipse.org/
[8] E. Juergens and F. Deissenboeck, "How Much is a Clone?," SQM'10, Madrid, Spain, March 2010.
[9] Apache Subversion; http://subversion.apache.org/

ISBN 978-3-95450-139-7