CERN Open Data Portal - Improving usability and user experience of CMS Open Data research tools.

Harri Hirvonsalo

Abstract
This report summarizes the work I have done during my assignment as participant of CERN Summer Students 2015 programme. Main goal of my Summer Student project was to lower the bar for people to start utilizing open data that CMS experiment has released in November 2014 to CERN Open Data Portal (http://opendata.cern.ch).

Project included various working packages and tasks, such as:
- Determine the obstacles that potential users of CMS research oriented open data who don’t have previous knowledge about internal workflow of analysis tasks at CMS experiment would run into.
- Produce more introductory material and tutorials for conducting basic physics analyses with CMSSW to CERN Open Data Portal.
- Study the feasibility of podio-framework (https://github.com/hegner/podio) for CMS Open Data users.

The project work was done under the supervision of Kati Lassila-Perini whom I thank greatly for her help, patience and support.
Introduction

CERN Open Data Portal ([http://opendata.cern.ch](http://opendata.cern.ch)) was launched in November 2014. The portal provides open access to research data produced by experiments at the Large Hadron Collider (LHC). The available material is classified under two categories, Education and Research. The audience of Education-related materials are high-school level educators and students as well as to general public interested in High-Energy Physics (HEP). Research-section is geared towards, but not limited to, scientific re-use of actual research data. CMS experiment provides almost 40 terabytes of data, tools to analyze this data and brief tutorials on how to use them. Currently the amount of this introductory material is somewhat limited and it takes quite an effort to utilize the data without prior knowledge of the internal workflows of CMS experiment. My summer project aimed to lower the bar for people to start utilizing open research data released by CMS experiment.

Project background

The original description of the summer student project I got selected to was following: “The student will design and develop online teaching applications based on the CMS experiment open data. The main audience of these applications is at the high-school level, and the student is expected to give his/her view on how to best target this young public.”

After I arrived at CERN this plan changed a bit. I started to focus more on Research-section of the CMS pages in CERN Open Data Portal. The reasoning behind this decision was that in order to be able to say what the potential users of CMS Open Data need, someone without prior knowledge about inner workflow of CMS experiment needs to go through the materials and identify the weak spots of current material; what is missing and what can be considered hard to understand. In addition to the more exploratory goals of the project one of the secondary goals was to evaluate the feasibility of a new data model library for CERN-ROOT based data-analysis.

From retrospective point of view I would say that this was a very good decision. Not only I got to learn CMS Software framework (CMSSW) and how to use it, but I also got valuable,
first hand information about data-analysis in the area of high-energy physics. My main contribution for CMS collaboration is undoubtedly the insight I can provide after going to through the same learning process that people without prior knowledge about CMS experiment's internal workflow would presumably undergo.

Main work packages / tasks
In this section I briefly describe the tasks / work packages I worked on during the project.

Introduction material and tutorials
In my opinion the examples about data-analyses found in CERN Open Data portal do a very good in what they are supposed to do; give a glimpse on what an analysis looks like and what are the activities needed to run the examples. What they don’t provide is the knowledge on how to write your own data-analysis code. For this matter Open Data portal suggest to turn to more detailed data sources, the TWiki-based CMS WorkBook [https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBook] and the CMS Offline SW Guide [https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuide]. The amount of information available from these sources is overwhelming. TWiki-pages are structured quite good, but the level of detail varies a lot and at least I felt that I was a bit lost navigating the documentation. As I digged deeper, I noticed that there are pages providing good information that one would not find by following the table of contents of CMS WorkBook or SW Guide. Documentation clearly was (and still is) one part where there is room for improvement, and I wanted to work on that.

As so much information is already written and available in these public CMS TWiki-pages, there was no point in writing yet another introduction or tutorial. On the other hand, I felt that examples found in TWiki are a bit oversimplified and lack a clear, physics related "background story". What I wanted to do, was to use some existing “real” analysis code as an example and comment (almost) every line in CMSSW' Python-configuration file and in the analysis module C++ -file at least from software point-of-view; what does this line do, why it is needed and a link to relevant CMS TWiki-page for more information. Comments from more physics oriented point of view would also be good, but I certainly wasn’t experienced enough to provide this.

Luckily enough, one of the research group utilizing CMS open data (Achim Geiser and Irene Dutta from DESY) provided me with preliminary code used in their analysis and they also offered to help me with the physics oriented documentation of the source code. Their analysis studies the feasibility of CMS open data by reproducing plots from famous, highly cited papers released by CMS collaboration and as such is a good candidate for the base of the example source code. The original publications are freely available (CMS-EWK-10-002, CMS-MUO-10-004, CMS-BPH-10-002, CMS-QCD-10-006, CMS-FWD-11-001) for people to read and this way the source code is presented in a meaningful context. Of course the already existing examples have their place too.
Currently this work package is not finished. I have studied the source code, commented roughly half of it from software oriented point of view and delivered it to Achim Geiser for proof-reading and physics oriented commenting. I have also started dividing the source code into modules where each one reproduces plots from one study only. In the original version the code produced plots from all the papers on one run and that is the way to go if you’re producing plots. If you're learning material you want to have the code in small enough pieces for it to be comprehensive, even if it isn’t performance-wise from data-analysis point of view.

I’m hoping to get this work package finished before my contract at CERN ends, but if I don’t, I’m going to continue working on it as an collaborative user of CERN Open Data portal.

Integrated development environment suitable for CMSSW

“An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. IDEs are designed to maximize programmer productivity by providing tight-knit components with similar user interfaces. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs have intelligent code completion. IDEs present a single program in which all development is done”


The idea for providing this kind of tools for CMS open data users hit me after I started to actually analyze the material for the new example source codes. I realized that the job of learning the nuances of CMSSW (functions, methods, classes, code structures etc.) and analysing code that you haven’t written yourself might be more faster with an IDE. As with all the things that are matters of taste, there are differentiating opinions, but based on my experience gained through four years of software engineering (design and development) studies I would recommend using one; learning a new software framework and using a language that is not that familiar, has been, at least for me, much more easier by using a proper IDE.

The following list of requirements was used for guidance for selecting an IDE to work with, but failing to fulfill a single or even two to three requirements didn’t necessarily exclude an IDE from comparison:

- Strong support for C++ and Python, as these are the main language used when working with CMSSW. XML support was seen as bonus.
- Language and library-level (e.g. CERN-ROOT and CMSSW specific libraries) support for code autocompletion, syntax checking, refactoring and hyperlinking of source and documentation files.
- As much operating system independent as possible.
- Preferably Open Source Software (OSS), because CMS encourages and all the material available in CERN Open Data portal promote the use of OSS and openness in general.
- Integrated visual debugger.
- Support virtual workspaces / projects / environments to ease working on multiple analyses simultaneously.
- Integrated visual performance profiler for benchmarking resource usage and performance of analysis code.
- Integrated version control and collaboration tools (e.g. Git-integration), preferably with a GUI.

As this development and / or customization of IDE should be done as quickly as possible, already existing material (scripts, documentation, tutorials) on the matter were seen as a big plus. In addition, the IDE should provide means for remote development using a virtual machine as the preferred way of working with CMS open data is to use a CMS specific virtual machine. The aim of this requirement was to increase performance of the IDE by running it in the host operating system (the operating system used to emulate the virtual machine) and executing the actual analysis in the virtual machine. The selected IDE provides support for remote development, but the feature is not currently used.

The following IDEs or advanced source code editors were seen as the strongest candidates. Existing material for customising the IDE in question for CMSSW and / or CERN-ROOT is provided in parenthesis.

- CodeLite (http://codelite.org/)
- Atom - https://atom.io/
- Liclipse - http://www.liclipse.com/

The selected IDE was Eclipse. Another one I evaluated quite much was CodeLite. Liclipse seemed very interesting but due to its commercial licence I didn’t try it out. SublimeText was rejected for much of the same reasons.

Eclipse IDE
Eclipse CDT (C/C++ Development Tooling) extended with PyDev-plugin is the main configuration used. CDT-package delivers C/C++ language and compiler support and PyDev provides support and tools for Python-development. Eclipse CDT comes with a plugin for git-integration, EGit.

Eclipse IDE was chosen mainly because it is Open Source, I had quite a lot previous experience with it and I found various sources which described how to configure and customize it for CERN-ROOT and CMSSW related work. The instructions weren’t however as comprehensive I had hoped and there was room for example improving user experience,
by getting the visual debugger to work and having the possibility to open ROOT-shell or TBrowser-window straight from the user interface of Eclipse.

Eclipse provides support for performance and memory usage profiling, but because of limited amount of time I only minimally tested them. Gprof that come with the GCC-compiler didn’t work as expected as cmsRun didn’t generate profiling information file (gmon.out) at the end of the program execution. I did try IgProf which is a performance profiling tool originating from CERN (http://igprof.org/). It worked very well and was easy to setup, but I couldn’t come up with to integrate IgProf into Eclipse’s user interface. Eclipse offers support for Valgrind-tool (and for the other supportive tools Valgrind uses), so it might be something worth checking out.

I wanted have the IDE ready to use for Open Data portal users as painless as possible and therefore it needed to be run in the CMS Open Data virtual machine. Because of restrictions imposed by the virtual machine I had to use Eclipse 4.2.2 (Eclipse Juno SR2), but by updating virtual machine’s OpenJDK-package to version 1.7.0, Eclipse 4.4.2 (Eclipse Luna SR2) should run without problems. Upgrading the OpenJDK-package also enables to use more recent version of PyDev-plugin, which in turn provides new features to Python-development.

Latest version of Eclipse at the time of writing was Eclipse 4.5 (Eclipse Mars), but unfortunately it cannot be installed on the virtual machine due to Scientific Linux 5 lack of support for GTK+-packages after version 2.10.4 and Eclipse Mars requires at least GTK+ 2.18.0.

Future improvements
While working on this task, I came up with couple of interesting ideas for alternative ways and alternative development tools for CMS Open Data.

Oomph installer
With Eclipse 4.5 / Mars, Eclipse Foundation has released a very user friendly installer that aims to provide a way to setup a development environment around Eclipse automatically. The installer is part of Oomph-project (https://wiki.eclipse.org/Eclipse_Oomph_Authoring) which has been developed with the idea that people wanting to contribute to some OSS project, don’t want to spend a day setting up the development environment. Instead they want to focus on programming, testing and delivering their contributions to upstream repositories with ease. Oomph-project aims to provide a development environment that has been configured to some specific project straight out-of-the-box.

I spent a day working on Oomph but the main problem was that I couldn’t get Oomph-installer to work on CMS virtual machine as it seems that it uses libraries from Eclipse Mars and therefore it doesn’t run on an operating system without at least GTK+ 2.18.0. The Oomph tool itself should work for Eclipse Luna installations and I filed a feature request to support the installer on system with an older version of GTK+. If the feature is implemented this could be a good way proceed with further easing the Eclipse based development environment for CMS Open Data and even if it doesn’t future releases of CMS
Open Data might come with a Scientific Linux 6 based virtual machine where the installer definitely works.

**Eclim**

Although I decide to focus on Eclipse, I understand not everyone likes to use it for editing source code. As a matter of fact, I myself only use it for bigger projects where Java or C/C++ is used. During the IDE evaluation I ran into Eclim-project (http://eclim.org/) that aims to bring the strongest parts of Eclipse (code completion, syntax checking, source file indexing, etc.) to vim text editor. Basically Eclim provides an interface that other text editors can use to make calls to the Eclipse’s engine. Although Eclim is developed for vim, the interface it offers is not tied to vim and there are more or less active project that provide Eclim compatible plugins for text editors such as TextMate, Sublime text and Emacs [http://eclim.org/relatedprojects.html]

I didn’t have time to test them but if the Eclipse installer based customization for CMSSW ever gets going Eclim could be used to provide support for other text editors as well.

**Web-browser based IDE**

This was kind of a wild idea. What if CMS Open Data virtual machine would host a web-server that in turn offers some open source IDE that runs in a web-browser? User would basically download the virtual machine, run it without GUI and interact with virtual machine and CMSSW through web-browser.

There are several commercial products available (e.g. Cloud9, Codeanywhere) that provide an IDE, a file browser and access to terminal, all within any modern web-browser. Cloud9 seems to offer some parts of their product as open source software in their Github-repository (https://github.com/c9/core), but it might be difficult to set up on SLC5.

I would call this work package finished as soon as I write installation and setup instructions and possible automation scripts for them. Tutorials, instructions and how-to’s on the actual usage of Eclipse would be good too, but there already exist quite a lot of material for this. Of course CMSSW specific tutorials might be useful. This customization is not perfect and there is always room for improvement. Also the remote development features of Eclipse were not utilized at all, but based on the short tests I conducted, simple ssh-shell and remote compiling didn’t provide good user experience.

**Plain Old Data Input Output (podio) library**

The goal of this task was to find out if podio-library [https://github.com/hegner/podio] could be used to help the re-use of CMS Open Data.

Describing what podio-library is, is hard. In general, it is a set of C++ classes and a class-hierarchy generator for implementing custom data models. It aims to provide memory safety by providing an abstraction layer that handles operations typically associated to error prone programming behind the scenes.
It’s intended usage is in high energy physics data-analysis, but due to its general nature, there is no real restrictions for using it in some other context. In the end it is a data model generator not a data model generator for high energy physics data-analysis. For a more detailed explanation and usage instructions I would suggest reading these https://github.com/hegner/podio/blob/master/documentation/doc.md, https://github.com/hegner/podio/blob/master/lcio/datalayout.yaml and not just the front page of podio github-repository. More documentation might be available also from this project as it and podio clearly have some common features: https://github.com/HEP-FCC/albers-core

This was actually the first one I worked on during my stay at CERN. My instructor, me and the author of this library (Benedict Hegner) had couple of meetings and I worked with this for few weeks but have to admit that I didn’t proceed far; maybe it was the lack of understanding about CERN-specific data-analysis or I weren’t experienced enough with programming. Nevertheless, I realized that I first need learn how to write analysis code for CMSSW and then try to get back on this one.

I’m including this task in the report for the sake of completeness and for the fact that podio is clearly something worth looking into. Used in the right way it could provide more error free and a simpler way to write data-analysis code. Regarding to CMS Open Data, podio might provide a way for people not interested writing analyses that utilize CMSSW to extract and convert available open data into a format more convenient for them.

Conclusions

Working on this CMS Open Data project has been very pleasant. I got to work on the tasks I felt confident enough work on and the level of support and encouragement was amazing. As for the results, I am happy that I got something done, but on the other hand I’m a bit disappointed as I believed that this timeframe of three months would have been more than enough to finish all the tasks presented in this report and maybe even more. Nevertheless working on this particular project has sparked something, as I currently feel that I will continue working with CMS Open Data and hopefully contribute to its development. Oh and I did do a poster about CERN Open Data Portal and for summer student poster session. It is more or less an advertisement for the portal but it briefly introduces two tasks from this summer project. For the time being it can be found here: https://docs.google.com/presentation/d/176GA5mbYVlWjNFe-vTpk5j09q8QbXZ7-Ik3aEqWIQCTM/edit

In a more general context the CERN summer student program has been a great opportunity to gain experience on working in a large, international research institute. I also got some understanding what it means to be working in the area of high energy physics with the best. There are so many people with various kind of expertise and doing scientific research on such a large scale wouldn’t be possible without a single one of them.

I had a wonderful time at CERN. It truly was an unique and fabulous experience. Thanks for everyone I had the pleasure of working with.