Journal of Physics: Conference Series **396** (2012) 062020 doi:10.1088/1742-6596/396/6/062020

Developing CMS software documentation system

M. Stankevicius[1], K. Lassila-Perini[2], S. Malik[3]

[1] Faculty of Mathematics and Informatics, Vilnius University, Lithuania

[2] Helsinki Institute of Physicss, PO Box 64, FI-00014 University of Helsinki, Finland

[3] University of Nebraska-Lincoln, Lincoln, NE 68588-0299, U.S.A

E-mail: mantas.stankevicius@cern.ch

Abstract. CMSSW (CMS SoftWare) is the overall collection of software and services needed by the simulation, calibration and alignment, and reconstruction modules that process data so that physicists can perform their analyses. It is a long term project, with a large amount of source code. In large scale and complex projects is important to have as up-to-date and automated software documentation as possible. The core of the documentation should be version-based and available online with the source code. CMS uses Doxygen and Twiki as the main tools to provide automated and non-automated documentation. Both of them are heavily cross-linked to prevent duplication of information. Doxygen is used to generate functional documentation and dependency graphs from the source code. Twiki is divided into two parts: WorkBook and Software Guide. WorkBook contains tutorial-type instructions on accessing computing resources and using the software to perform analysis within the CMS collaboration and Software Guide gives further details. This note describes the design principles, the basic functionality and the technical implementations of the CMSSW documentation.

1. Introduction

The CMS (Compact Muon Solenoid) experiment [1] is one of two large general-purpose particle physics detectors built on the LHC at CERN in Geneva, Switzerland. CMSSW[2] (CMS SoftWare) is the overall collection of software and services needed by the simulation, calibration and alignment, and reconstruction modules that process data so that physicists can perform their analysis. CMSSW is stored in a CVS[3] repository. The source code is written in C++ and the analysis jobs are configured in python. The goal of the CMS software documentation system is to provide software and analysis related information to all users who use it to perform physics analysis and contribute towards its development. The documentation should efficiently aid in future software development without intervening in its ongoing use.

The way CMS documentation is written strongly differs from commercial products. Commercial products have strict documentation on how a product must look and behave and might have intuitive GUI which does not require much documentation. The CMS documentation, on the contrary, especially work flows and guides, are written after completing a task or software release.

CMS uses Doxygen[4] and Twiki[5] as the main tools to provide automated and non-automated web-based documentation. Both of them are heavily cross-linked to prevent duplication of information. Doxygen is used to generate web-based code documentation (Reference Manual) from source code. Depending on a specific need, an additional set of scripts is used to extend functionality of Reference Manual. The Twiki is divided into two parts: WorkBook[6] and Software guide[7].

The CMS documentation is entirely web-based. There are many advantages of a web based documentation: it is independent of the operating system and the type of the web browser, it is cheap and easy to distribute (typically supplied over the Internet), and relatively simple to implement and it has easy navigation and search features. However, some issues exist. Page formatting might be difficult and restricted. If supplied over the Internet, large graphics take a relatively long time to download. But for a large distributed collaborations such as in CMS pros outweigh cons.

2. Tools and services

The CMS documentation tools are used to provide software documentation[8] for mainly three purposes: tutorials which are designed to teach (2.1.1. WorkBook), procedures which are designed to guide (2.1.2. Software Guide), and reference materials that are designed to support (2.2.1. Reference Manual). These tools and corresponding documentation are described below.

2.1. Twiki

CMS uses a web Twiki for its knowledge base and documentation management system. The flexibility to edit and modify a Twiki by any CMS user and instantly publish it using only a web browser (no programming required) makes that very suitable for a collaborative environment. Uniform graphical look of the pages is an advantage. The twiki page on each topic has a responsible person associated with it. Pages are categorised as follows:

2.1.1. WorkBook. WorkBook serves as the first entry point for a user on how to go about approaching physics analysis. The WorkBook is task specific and covers all aspects to help and equip a user jump-start physics analysis. One learns about how to get computer accounts and learn the CMS computing framework, common physics tools and examples on advanced analyses.

2.1.2. Software Guide. The Software Guide (SWGuide) is structured into domains which most often coincide with the organization of the Offline and Computing and Physics Projects. Each subgroup in these projects nominates a contact person for documentation and user support issues. These contact persons are responsible for its contents and they organize the documentation work within the group with the technical help from the User Support Group. They also help identify experts in the group in case of user questions on areas under the group's responsibility. For readability of the document, the main pages for each domain have a similar structure - a template page is provided. For each domain, the main page should include links to the existing documentation in the WorkBook. All relevant instructions for the use of the code should be provided, and all algorithms used in the domain should be documented, in case of an existing written note, a link to the note is sufficient.

2.2. Doxygen

Doxygen can generate a web-based documentation (in HTML) from a set of documented

source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code. One of the most time-saving features in Doxygen is its autolinking support. As it analyzes comments, it looks for specific patterns: camelCasing, parenthesis(), name::spaces and attempts to determine whether those strings are actually references to the existing classes, functions, namespaces, variables, etc. If it determines that they are, it automatically generates a link to the appropriate generated documentation.

2.2.1. Reference Manual. The Reference Manual is a web-based source of documentation that is composed of different tools. The code documentation is generated using Doxygen. Additional set of scripts extend functionality.

2.2.2. Package documentation. The software package documentation is a part of the Reference Manual. The goal of the package documentation is to give an overview of the package functionality at the level of being useful to people who take over the responsibilty of a domain, such as the new convenors or developers. Some of the packages are already documented in different places such as Twiki. These documents can be linked to avoid duplication of information.

2.3. CMSSW utilities

In addition to the documentation suite described above, CMSSW includes several utility functions to inspect the data contents or the software structure. For example, *EdmDumpEventContent* is an inspection tool that shows the content of a data file. It specifies, for each product, which are the kind of objects and their module and instance labels. It is foreseen to expand the use of such tools to provide instantaneous documentation.

3. Issues and solutions.

A documentation system that is maintained, shared and updated by a large number of CMS users comes with its own share of issues. We have identified several of them. These issues and their solutions are described below.

- **Issue*:*** *Keeping documentation up-to-date.* CMS develops and makes analysis in large scale which makes it difficult to keep non-automated (Twiki) documentation up-to-date.
  Solution*:* *Document Maintenance.* Maintenance activities include correcting errors, updating propagated changes and overall improving the document's content as well as archiving (or removing) older content.

- **Issue*:*** *Format and style of documentation.* There is no single defined format and style for how to write documentation because documentation types vary widely. Much more effort is required to format / prepare software documentation than the effort of providing the actual content for the documents.

  Solution*:* *Template preparation* is the process of providing the same guidelines for a particular documentation task to avoid the complications of documentation formats, styles and technologies. This also highly reduces the effort of providing actual content for the documents. Practice shows that most of users prefer short and plain instructions, but unfortunately, short messages do not contain enough information to clarify a topic. However, if instructions are long and explicit, people

do not always read them.

- **Issue:** *Responsibility and motivation of content providers.* Due to the quick turnover of content providers it is not always clearly defined who has the responsibility for certain types of documentation tasks.

  The success of well written documentation highly depends on the good will of content providers. Sometimes team members are not motivated to document their code as they see little immediate benefit in maintaining supporting documents.

  Solution*: Communication with content providers***.** Communication is a very important process to understand the needs, provide an outlook from a different point of view and to create an overall picture of the documentation process. An example, documentation of the CMSSW data formats. Data formats were maintained in tabular in a Twiki page. Many content providers found it difficult to keep up the dependence of the documentation on CMSSW release schedule and hence, saw little benefit in maintaining pages this way. A decision was made, together with content providers, to migrate existing data format documentation from the Twiki to the CVS and to keep updating it there. In this way maintaining the dependence of the data format documentation on the CMSSW release became more automated and easy.

- **Issue:** Missing *Reference Manual functionality.* There is some functionality that the CMS documentation needs but Doxygen cannot provide: extracting and linking configuration files, splitting lists of classes, namespaces and configuration files, grouping package documentation by package and domain. The large amount of CMS software documentation makes its browsing slow. Users need the ability to switch between two different software documentation releases.

  Solution: *Additional scripts. A w*ide variety of scripts is used to provide additional functionality. Several CMS services are integrated to provide up-to-date information about a particular CMSSW release. To increase the browsing speed and readability, large amounts of CMS code documentation are split into smaller pieces.

- **Issue:** *Duplication of information* due to miscommunication between teams.

  Solution: *Reminding to keep documentation DRY (Don't repeat yourself)* A principle stating that every piece of knowledge must have a single unambiguous, authoritative representation within a system.

- **Issue:** *Maintenance of Twiki pages.* The challenge of such an enormous amount of information is to track the information that may be stale or outdated.

  Solution: *Twiki reminder service.* To deal with this and improve the performance of the Twiki usage, a reminder service of Twiki pages has been tested and deployed. As a result, each person responsible for a Twiki page gets a reminder about the pages that he or she maintains and is encouraged to remove the outdated information. To reduce the effort on the part of the responsible, the reminder service in addition offers a possibility of a delete request. The Twiki user only

needs to select the topics to be deleted and they will be deleted centrally.

4. Recommendations

CMS has different recommendations and rules for writing documentation. While some are general, others are specific to the documentation tool used.

4.1. General recommendations

The usability of documentation has been evaluated[9] by these principles[10]:

- *Match between documentation and the real world*. The documentation should speak the users' language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

- *Match between documentation and the product*. The forms, screens, manuals, and online helps system should match so that the same terminology is used in all of them. This may contradict with "Match between the documentation and real world" if the interface uses strange terminology.

- *Support for different users*. The documentation should support users with different levels of knowledge on the domain as well as those assigned different tasks in the domain. Any unnecessary information for a specific user must be hidden from other users or be easily overlooked. Quick reference information for expert users should be available.

- *Effective information design*. Information must be presented in a way that is easily found and understood by the users. Short lines and paragraphs are easier to read. Graphics, tables, and lists are easy to scan and read, and appropriately used to support the information need the user has. Unnecessary graphics only slows the reading and the download time of web-based documentation. Write instructions in imperative form and address the user directly using active sentences.

4.2. Doxygen (code documentation) rules.

Every class and class method should have a brief one-line description, and a detailed description of the algorithm. Methods also require descriptions of all inputs and outputs.
There are two ways of writing comments in source code for Doxygen:

- Regular comments.
- A special comment block is a C or C++ style comment block with some additional markings, so Doxygen knows that it is a piece of a structured text that needs to end up in the generated documentation. For Python and other programming languages there are different commenting conventions.

The Doxygen website explains ways on how to structure the contents of a comment block in the code so that Doxygen incorporates them in the documentation and the output looks good.

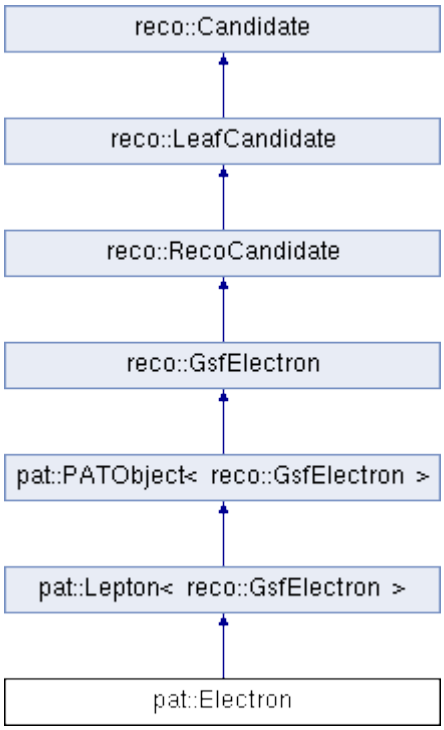

Figure 1. Class dependencies

In most cases a picture is worth a thousand words, see for example the inheritance diagram in Figure 1. A simple flow diagram of a process can make that process easier to understand by showing the relationships between the various tasks, verification steps and deliverables and by showing who is responsible for each task or verification step.

4.3. Recommendations for Twiki pages

Twiki pages should:

- **be clear.** I.e. be plain, direct, unambiguous, and specific.
- **be as concise as possible.** Verbosity is not a reliable solution against misinterpretation. Omit needless words. A direct and a concise writing may be clearer than random examples. Footnotes and links to the other pages may be used for further clarification.
- **maintain scope and avoid redundancy.** Clearly identify the purpose and scope early in the page. Content should be within the scope. In case the scope of one page overlaps with the scope of another, minimize redundancy. When one page refers to another, it should do so briefly, clearly and explicitly.

4.4. Audience

The CMS audience differs by the task they perform within the experiment and the level of computing skills and knowledge (novice, experienced, or expert) that they may have. For the author, defining the target helps writing a document that is applicable to the needs and level of knowledge of the user. Each piece of software may have more than one target audience, and it is important to differentiate between the types of users so that the needs of each of them is met as much as possible.

Once the audience is defined, the next step is to analyze the tasks of that audience. The tasks are the steps the user takes to complete their job. These tasks can be determined by observing the users at their job and having one-on-one and group conversations with the user group.

5. Conclusions

The objective of this paper is to describe current practices followed in the preparation of CMSSW documentation. It describes common issues rising from need to manage large amount of information and to deal with a large number of CMS users. Tools and methods solve most of issues and satisfy CMS needs, but there is room for improvement. In the future it is planned to better exploit current tools to improve overall documentation system and to involve more people in order to cover more fields.

6. References

[1] CMS Collaboration; The CMS experiment at the CERN LHC, JINST 3 (2008) S08004.
[2] https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookCMSSWFramework
[3] http://cvs.web.cern.ch/cvs
[4] http://www.doxygen.org
[5] http://twiki.org
[6] https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBook
[7] https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuide
[8] Sudhir Malik, 2011 J. Phys.: Conf. Ser. 331 082006
[9] Kati Lassila-Perini 2010 J. Phys.: Conf. Ser. 219 082011
[10] Purho V 2000 Heuristic inspections for documentation – 10 recommended documentation heuristics Usability Interface 6. Available at http://www.stcsig.org/usability/newsletter/0004-docsheuristics.html