



EDWARD MOYSE

---

# SURVEY OF DOCUMENTATION TOOLS AND PRACTICES IN OTHER EXPERIMENTS

# INTRODUCTION

- ▶ We thought it would be instructive to try to get a feeling for what other experiments and opensource projects use
  - ▶ Including **code documentation, user guides** (e.g. git tutorial, our workbooks) and **wikis**
- ▶ In the next few pages I will show a few examples
  - ▶ (And add some subjective opinions)
- ▶ Then will summarise a few of the discussions (thanks to Graeme for the introductions!) we had with experts from other communities, in particular:
  - ▶ **Linear Collider** - Frank Gaede
  - ▶ **SuperNemo & Geant4** - Ben Morgan
  - ▶ **Belle II** - Martin Ritter

# DOCUMENTATION SURVEY

## GAUDI

- ▶ Main webpage looks to be a simple static homebuilt one
- ▶ Extensively use doxygen to document code
  - ▶ however they make a lot more use of 'related pages' than ATLAS does
  - ▶ have instructions for how to profile, code style conventions, how to build
  - ▶ i.e. they use doxygen for user guides
    - ▶ Subjective bit - I find them a bit ugly, but at least they're full of content and shipped 'for free' with the code
- ▶ They build documentation for every release

The screenshot shows the 'Class Hierarchy' page of the Gaudi Framework documentation. The page title is 'THE GAUDI FRAMEWORK v30r0 (c919700c)'. The navigation menu includes 'Main Page', 'Related Pages', 'Modules', 'Namespaces', 'Classes', and 'Files'. The 'Classes' menu is expanded to show 'Class List', 'Class Index', 'Class Hierarchy', and 'Class Members'. The 'Class Hierarchy' page content includes a heading 'Class Hierarchy', a link to 'Go to the graphical class hierarchy', and a note: 'This inheritance list is sorted roughly, but not completely, alphabetically:'. Below this is a list of classes, including `__longlong`, `NTuple::_Accessor< TYP >`, `NTuple::_Accessor< _Array< TYP > >`, `NTuple::_Accessor< _Item< bool > >`, `NTuple::_Accessor< _Item< IOpaqueAddress * > >`, `NTuple::_Accessor< _Item< TYP > >`, `NTuple::_Accessor< _Item< VALUE > >`, `NTuple::_Accessor< _Matrix< TYP > >`, `_CallbackStreamBufBase`, and `Gaudi::details::_container< CONTAINER, bool >`. A '[detail level]' link is visible at the top right of the class list.

The screenshot shows the 'Related Pages' page of the Gaudi Framework documentation. The page title is 'THE GAUDI FRAMEWORK v30r0 (c919700c)'. The navigation menu includes 'Main Page', 'Related Pages', 'Modules', 'Namespaces', 'Classes', and 'Files'. The 'Related Pages' page content includes a heading 'Related Pages', a note: 'Here is a list of all related documentation pages:', and a list of related pages: 'Platform Specific Sources', 'Gaudi Manual', 'Build System', 'How to build and use the Gaudi with CMake', 'Code Style Conventions', and 'Gaudi Plugin Service Instructions'.

The screenshot shows the 'How to build and use the Gaudi with CMake' page of the Gaudi Framework documentation. The page title is 'THE GAUDI FRAMEWORK v30r0 (c919700c)'. The navigation menu includes 'Main Page', 'Related Pages', 'Modules', 'Namespaces', 'Classes', and 'Files'. The 'How to build and use the Gaudi with CMake' page content includes a heading 'How to build and use the Gaudi with CMake', a section 'Requirements', and a note: 'The CMake configuration of Gaudi is based on the version 2.8.5 or later of CMake. On lxplus (SLC6) to call something like:'. Below this is a code block: 

```
$ export PATH=/cvmf/sft.cern.ch/lcg/contrib/CMake/3.7.0/Linux-x86_64
```

. The page also includes a section 'Quick Start' and a note: 'NOTE: If you use the LHCb environment (as of LbScripts v7r7), you do not need to prepare the environment'.

# LINEAR COLLIDER

- ▶ Moved from SVN to github just over a year ago
  - ▶ <https://github.com/iLCSoft>.
  - ▶ Moved/moving to exclusively use github for issues, feature requests and general discussion
    - ▶ “Advantage is everything is in one place”
  - ▶ Even documentation is in github (via **README.md** files), and plan is to stick to this for most smaller packages...
    - ▶ Still in progress - main [webpage](#) is simple HTML, generated by [zms\(?\)](#)
  - ▶ Still have doxygen for [some](#) packages
- ▶ One exception is [DD4HEP](#) , which has a wider audience and so it was considered worthwhile to give it a dedicated (and fancier webpage)
  - ▶ Created using [hugo](#) (faster? alternative to jekyll). Uses markdown + extensions (‘shortcodes’) for content


README.md

## Marlin (Modular Analysis and Reconstruction for the LINear Collider )

build passing coverity passed

The idea is that every computing task is implemented as a processor (module) that analyzes data in an LCEvent and creates additional output collections that are added to the event. The framework allows to define the processors (and their order) that are executed at runtime in a simple steering file. Via the steering file you can also define named parameters (string, float, int - single and arrays) for every processor as well as for the global scope. By using the framework users don't have to write any code that deals with the IO they simply write processors with defined callbacks, i.e. `init()`, `processRunHeader()`, `processEvent()`, `end()`.


Marlin is distributed under the [GPLv3 License](#)



### Instructions for building Marlin with CMake:

```
. path_to_ilcsoft_installation/v01-XX/init_ilcsoft.sh
```

[Return to website](#)



# DD4hep Manual

## 2 User Manual

This chapter describes how supply a physics application developed with all the information related to the detector which is necessary to process data from particle collisions and to qualify the detecting apparatus in order to interpret these event data.

The clients of the detector description are the algorithms residing in the event processing framework that need this information in order to perform their job (reconstruction, simulation, etc.). The detector description provided by *DD4hep* is a framework for developers to provide the specific detector information to software algorithms, which process data from particle collisions.

In the following sections an overview is given over the various independent elements of *DD4hep* followed by the discussion of an example which leads to the description of a detector when combining these elements. This includes a discussion of the features of the *DD4hep* detector description and of its structure.

### 2.1 Building DD4hep

The *DD4hep* source code is freely available. See the [licence conditions](#) . Please read the [Release Notes](#) before downloading or using this release.



## LINEAR COLLIDER (2)

- ▶ Use github pages for some status pages
  - ▶ <http://ilcsoft.github.io>
- ▶ Main webpage - plain HTML

Nightly	Pipeline for GCC and LLVM			
iLCSoft	pipeline <span>passed</span>			
AIDASoft	Build Status	Issues	PRs	Coverity Scan
<a href="#">DD4hep</a>	build <span>passing</span>	issues <span>4 open</span>	pull requests <span>1 open</span>	coverity <span>passed</span>
<a href="#">aidaTT</a>	build <span>passing</span>	issues <span>0 open</span>	pull requests <span>0 open</span>	coverity <span>passed</span>
iLCSoft	Build Status	Issues	PRs	Coverity Scan
<a href="#">CED</a>	build <span>passing</span>	issues <span>1 open</span>	vendor <span>unresponsive</span>	coverity <span>passed</span>
<a href="#">CEDViewer</a>	build <span>passing</span>	issues <span>0 open</span>	pull requests <span>0 open</span>	coverity <span>passed</span>
<a href="#">CLICPerformance</a>	build <span>passing</span>	issues <span>0 open</span>	vendor <span>unresponsive</span>	coverity <span>passed</span>
<a href="#">Clupatra</a>	build <span>passing</span>	vendor <span>unresponsive</span>	vendor <span>unresponsive</span>	coverity <span>passed</span>
<a href="#">CondBMySQL</a>	build <span>passing</span>	vendor <span>unresponsive</span>	vendor <span>unresponsive</span>	coverity <span>passed</span>
<a href="#">ConformalTracking</a>	build <span>passing</span>	issues <span>0 open</span>	vendor <span>unresponsive</span>	coverity <span>passed</span>
<a href="#">DDKaTest</a>	build <span>passing</span>	issues <span>0 open</span>	pull requests <span>0 open</span>	coverity <span>passed</span>
<a href="#">DDMarlinPandora</a>	build <span>passing</span>	issues <span>0 open</span>	pull requests <span>0 open</span>	coverity <span>passed</span>


international linear collider
SEARCH

LCC Home
ILC Home
CLIC Home
Physics & Detectors
Newsline
Tools

What is the ILC? ▶

Why do we need the ILC? ▶

Organisation

Press ▶

Publications ▶

Global Design Effort (archive) ▶

Quick links

Home > ILC Home > What is the ILC?

### What is the ILC?



Image: Rey Hori

# DOCUMENTATION SURVEY

## SUPERNEMO

- ▶ Use jekyll for [main page](#)
- ▶ Main code documentation is in doxygen. Also use doxygen + markdown for the [user guides](#)
- ▶ And also have an internal wiki...
  - ▶ Apparently not used much yet

Two tracker sections are coupled to one of the calorimeter walls, to make a half-detector.

### Searching for Neutrinoless Double Beta Decay

The NEMO (Neutrino Ettore Majorana Observatory) collaboration is an international physics effort including the experiment SuperNEMO and its predecessor, NEMO-3.

The [SuperNEMO demonstrator module](#) is currently being assembled at the [LSM](#) underground lab, located in the Fréjus tunnel near Modane, France. NEMO-3, also at the LSM, ran from 2003-11. Its rich repository of data is still being analysed today.

Both SuperNEMO and NEMO-3 are designed to study extremely rare double-beta decay processes, and in particular are looking for evidence of [neutrinoless double beta decay](#). This is a rare type of radioactive decay which has been predicted, but has never been observed. If this process was seen, it would prove that neutrinos were their own antiparticles, which could be a clue to the matter-antimatter asymmetry in the universe.

## Falaise 3.1.1

SuperNEMO Software Toolkit

Main Page Related Pages Namespaces Classes Files

### Welcome to the Falaise Documentation

Falaise is the software system for the SuperNEMO experiment. These pages document the usage of the core applications, together with the full C++ API of the Falaise extension system.

### Getting Started

If you're reading this online and don't yet have an install of Falaise, a [Quickstart Guide](#) is available. Note that at present Falaise and even here not all variants are guaranteed to work!

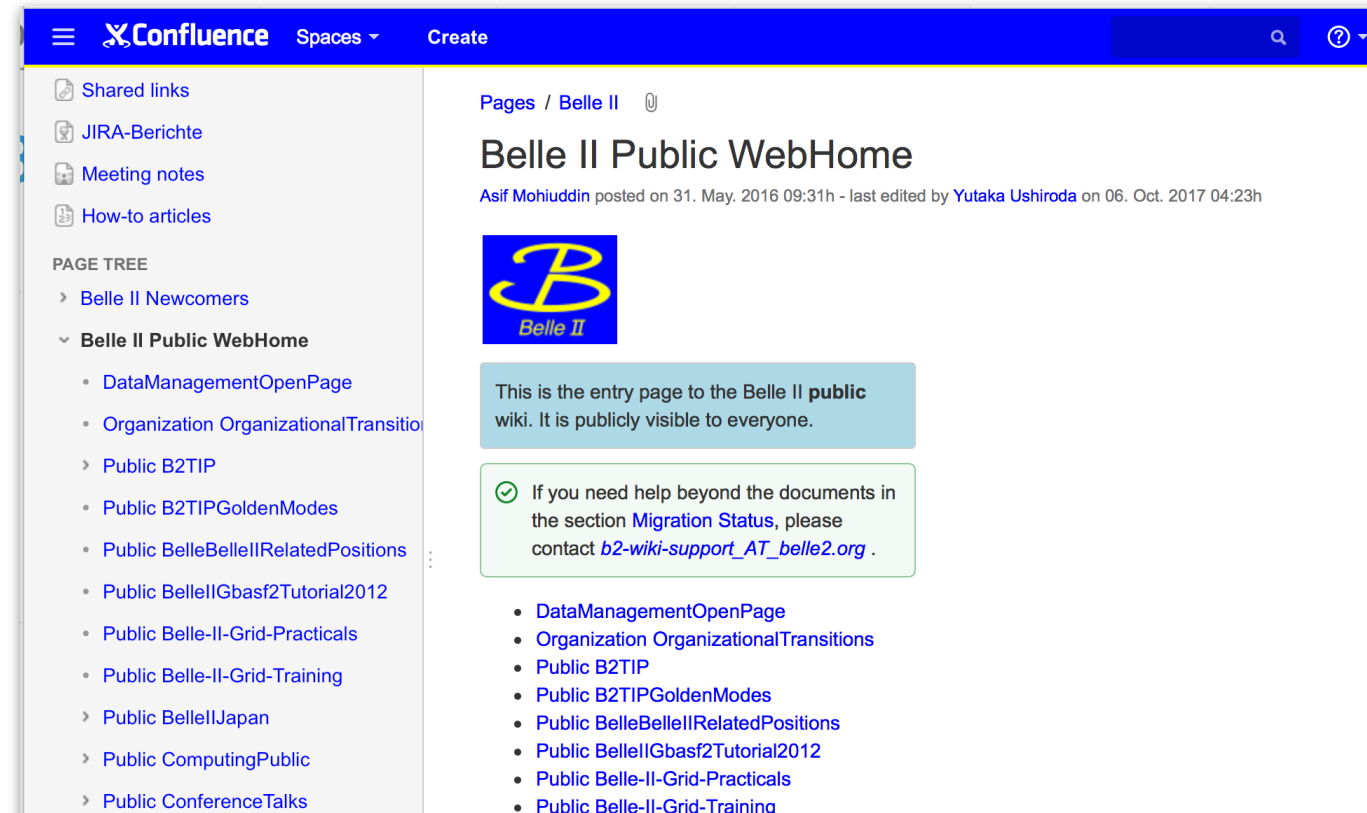
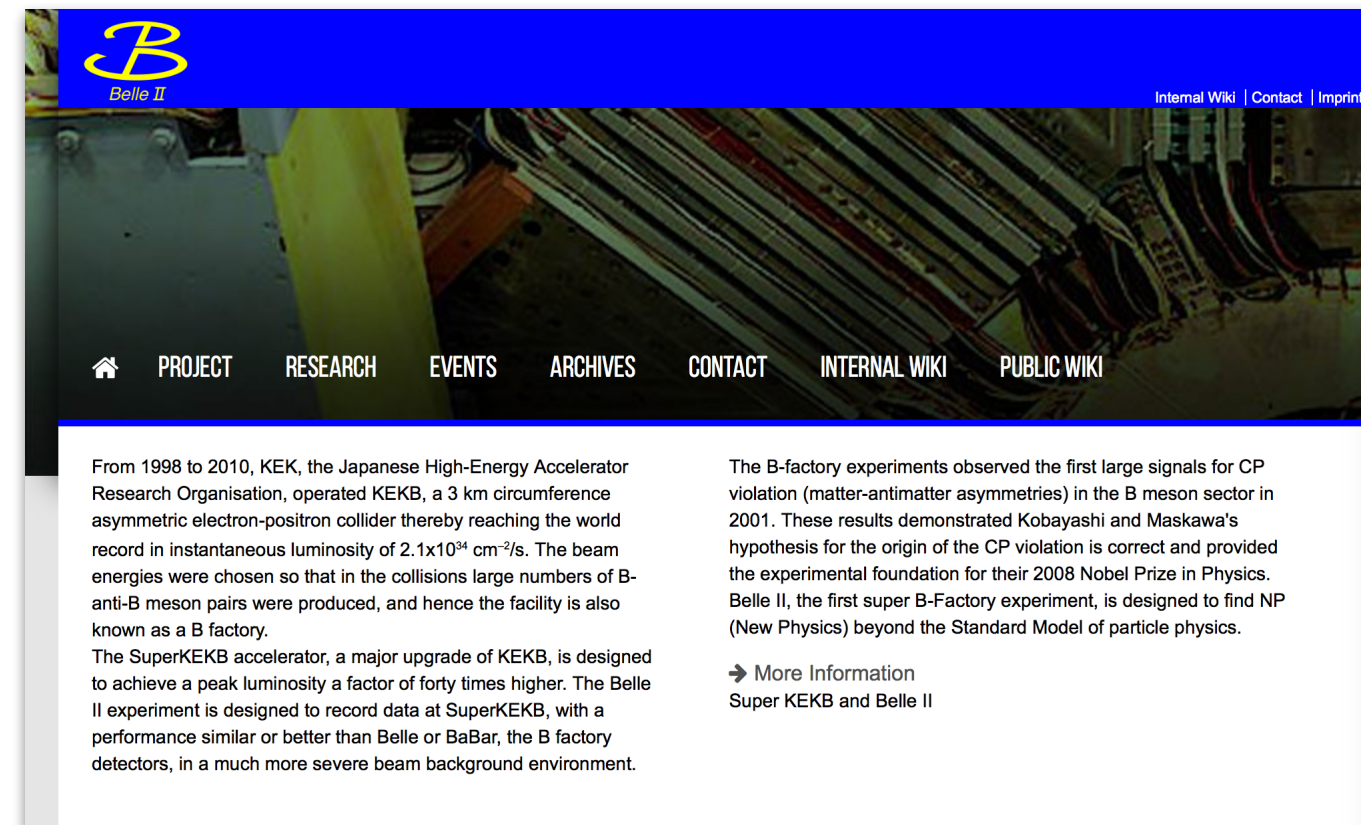
### User Guides for Core Applications and Libraries

Once you have an install of Falaise, the following pages provide introductions to using and extending the core applications:

- [FLSimulate](#): guide for using the Geant4 based simulations of the SuperNEMO Demonstrator and Commissioning detector
- [FLReconstruct](#) guide for using the pipeline application for reading, reconstructing and outputting data generated by the detector
  - [Data Structures output by FLReconstruct](#)
  - [Converting FLReconstruct BRIO Format to ROOT](#)
  - [Customizing the pipeline for flreconstruct.](#)

# BELLE II

- ▶ C++ documentation is done in Doxygen, but they run a 'preexec' (INPUT\_FILTER=) to insert some documentation to group it
  - ▶ Has consequence that line numbers don't entirely match source
  - ▶ They also parse 'algorithms' and add a diagram showing the inputs and outputs (would be trivial for us to do with Read/WriteHandles I guess!)
- ▶ In process of switching python documentation from doxygen to [sphinx](#)
  - ▶ Later will shift analysis code too
  - ▶ "Personally I think it's very nice how it allows to easily mix manual and automated documentation"
- ▶ Seems like they're using [Confluence wiki](#), both for internal use, and [public](#)
  - ▶ Looks like quite desy has a contract with atlassian?
  - ▶ Has anyone checked






# LHCB

- ▶ LHCB [starterkit](#) is very nice manual style documentation
- ▶ Written in [gitbook](#)
  - ▶ Markdown based
  - ▶ Is free (subject to some [limitations](#)) for opensource projects
- ▶ Also use [twiki](#) extensively
- ▶ Not sure if they use doxygen(!)
  - ▶ Couldn't really find any code documentation at all
  - ▶ starterkit does link to the gaudi documentation though

## LHCB Starterkit



**STARTERKIT**  
LHCb  
Est. 2015

Starterkit is a group of physicists who want to improve the working lives of young researchers working on the LHCb experiment.

We create and organise workshops that cover topics ranging from basic programming skills, such as using Bash and Python, up to in-depth explorations of the LHCb software.


**? Always feeling stuck?**


The goal of Starterkit is to demystify the LHCb software. If you've ever felt like you're not sure why you're writing the code you're writing, or not sure what code you even need to write, we can help you.

**↗ From the bottom up**

We start from the very basics, each lesson building on the last. Once you understand the details and how things fit together, you have the power to learn more on your own.

Branch: master ▾ [starterkit-lessons](#) / [first-analysis-steps](#) / [add-tupletools.md](#) Find file Copy path

 **saschastahl** Remove some typing 668936f on Nov 2

3 contributors 

200 lines (165 sloc) | 11.4 KB Raw Blame History

### TupleTools and branches

{% objectives "Learning Objectives" %}

- Add extra TupleTools to the default DecayTreeTuple
- Configure the extra TupleTools
- Use branches
- Find useful TupleTools
- Learn how to use LoKi functors in a DecayTreeTuple {% endobjectives %}

Usually, the default information stored by `DecayTreeTuple` as shown in our [minimal DaVinci job](#) is not enough for physics analysis. Fortunately, most of the information we need can be added by adding C++ tools (known as `TupleTools`) to `dtc`; there is an extensive library of these, some of which will be briefly discussed during the lesson.

{% callout "Default DecayTreeTuple tools" %} The default tools added in `DecayTreeTuple` are:

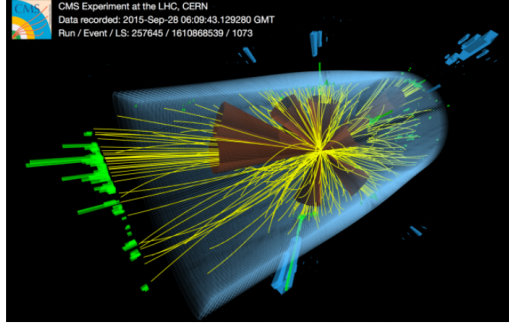
# DOCUMENTATION SURVEY

## CMS

- ▶ Code is on github
- ▶ Use [github pages](#) for user guides
- ▶ Doxygen to document C++
  - ▶ They have a hand-build 'splash page' for all releases
  - ▶ (implies they build doxygen for many releases)

CMSSW on GitHub Project page Feedback

## Welcome to CMS and CMSSW



The LHC smashes groups of protons together at close to the speed of light: 40 million times per second and with seven times the energy of the most powerful accelerators built up to now. Many of these will just be glancing blows but some will be head on collisions and very energetic. When this happens some of the energy of the collision is turned into mass and previously unobserved, short-lived particles – which could give clues about how Nature behaves at a fundamental level - fly out and into the detector. Our work includes the experimental discovery of the [Higgs boson](#), which lead to the award of a Nobel prize for the underlying theory that predicted the Higgs boson as an important piece of the standard model theory of particle physics.

CMS is a particle detector that is designed to see a wide range of particles and phenomena produced in high-energy collisions in the LHC. Like a cylindrical onion, different layers of detectors measure the different particles, and use this key data to build up a picture of events at the heart of the collision.

Many of CMS Software components (CMSSW) are [hosted on Github](#). You can find here a

### Quick start

- [Main page](#)
- [FAQ: CMSSW on Github](#)
- [Proposing changes to CMSSW](#)
- [CMSSW pull request approval workflow](#)
- [Collaborating with peers](#)
- [Working with CMSSW and UserCode](#)
- [Resolving conflicts & porting features](#)
- [How to use git through a proxy](#)

### Release Management

- [Building a CMSSW release](#)
- [Integration Builds results](#)
- [Historical plots](#)
- [Starting a new release cycle](#)
- [Forward ports](#)
- [Managing users and categories](#)
- [List categories and packages](#)
- [Latest available IBs](#)
- [List of pending PRs](#)
- [CMS-bot documentation](#)

<a href="#">Package Documentation Guide</a>	<a href="#">CMSSW Release List</a>	<a href="#">ECAL API Documentation</a>	
<b>Night builds</b>			
<b>CMSSW_10_0_*</b>			
	<a href="#">CMSSW_10_0_0_pre1</a>	<a href="#">CMSSW_10_0_0_pre2</a>	
<b>CMSSW_9_4_*</b>			
	<a href="#">CMSSW_9_4_1</a>		
	<a href="#">CMSSW_9_4_0</a>	<a href="#">CMSSW_9_4_0_patch1</a>	<a href="#">CMSSW_9_4_0_pre1</a>
			<a href="#">CMSSW_9_4_0_pre2</a>
			<a href="#">CMSSW_9_4_0_pre3</a>

# ASIDE: THIS IS WHAT ATLAS HAS

## **ATHENA DOCUMENTATION**

### **Release:**

[atlas\\_22.0.X-DOX](#)

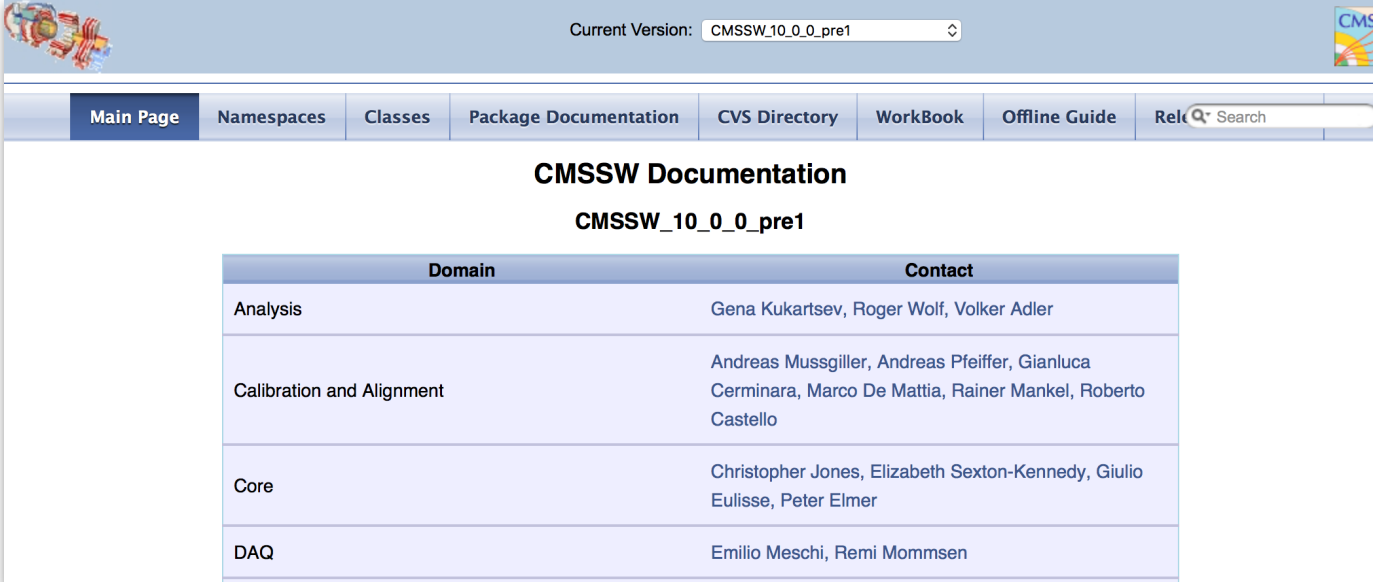
<https://atlas-sw-doxygen.web.cern.ch/atlas-sw-doxygen/index.php>

- ▶ I think we should make this look a bit nicer.
  - ▶ Adam did a great job with the functionality! But can we add some CSS?
- ▶ I think we should be building Doxygen for multiple releases.

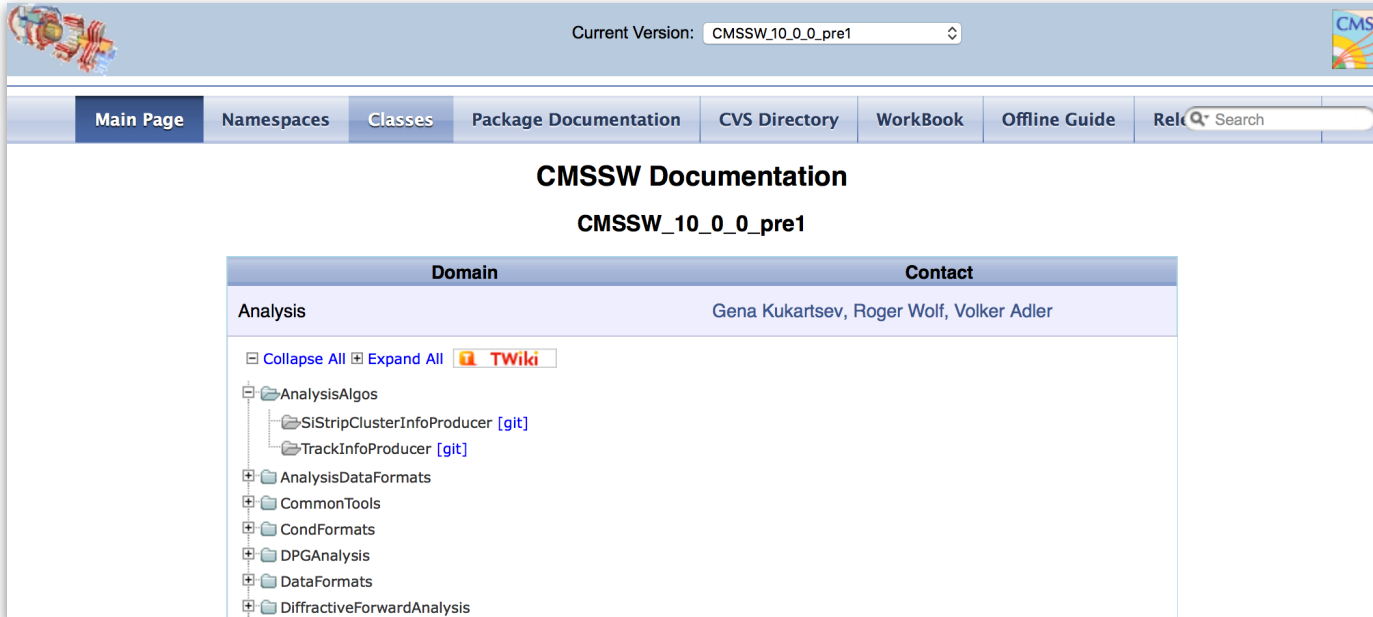
# CMS

- ▶ Doxygen cont'd
  - ▶ They divide up into domains with contacts
  - ▶ Can expand and shows content from twiki (?) and links to git
  - ▶ Code to generate this [here](#)
  - ▶ Not perfect - some links are broken, but I do think this linking between git and doxygen is very nice, and I also think that dividing the software up into Sections, makes it more comprehensible.
- ▶ Also:
  - ▶ [LXR](#) to search code
  - ▶ Make extensive use of Twiki, much as we do

[http://cmsdoxygen.web.cern.ch/cmsdoxygen/CMSSW\\_10\\_0\\_0\\_pre1/doc/html/index.html](http://cmsdoxygen.web.cern.ch/cmsdoxygen/CMSSW_10_0_0_pre1/doc/html/index.html)



Domain	Contact
Analysis	Gena Kukartsev, Roger Wolf, Volker Adler
Calibration and Alignment	Andreas Mussgiller, Andreas Pfeiffer, Gianluca Cerminara, Marco De Mattia, Rainer Mankel, Roberto Castello
Core	Christopher Jones, Elizabeth Sexton-Kennedy, Giulio Eulisse, Peter Elmer
DAQ	Emilio Meschi, Remi Mommsen

Domain	Contact
Analysis	Gena Kukartsev, Roger Wolf, Volker Adler

- [-] Collapse All [+]  
 [+]  
 [-] AnalysisAlgos
  - [-] SISStripClusterInfoProducer [git]
  - [-] TrackInfoProducer [git]
- [-] AnalysisDataFormats
- [-] CommonTools
- [-] CondFormats
- [-] DPGAnalysis
- [-] DataFormats
- [-] DiffractiveForwardAnalysis



# ALICE

- ▶ User guides
  - ▶ Use github pages for [top level](#) software webpages & user guides
  - ▶ They also use doxygen for this (see right)
  - ▶ And of course, use doxygen for documentation of [code](#)

ALICE SW

CI

AliPhysics

GIT

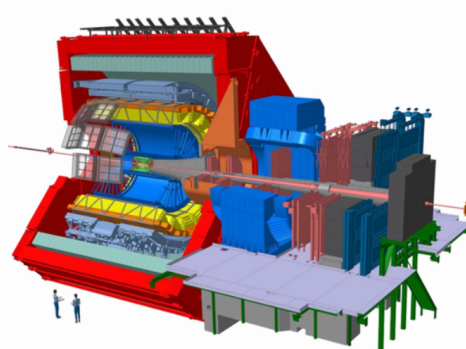
Basic Tutorial


Advanced Tutorial

Build Infrastructure

## ALICE Software on Github

The ALICE Collaboration has built a dedicated detector to exploit the unique physics potential of nucleus-nucleus collisions at LHC energies. Our aim is to study the physics of strongly interacting matter at the highest energy densities reached so far in the laboratory. In such condition, an extreme phase of matter - called the quark-gluon plasma - is formed. Our universe is thought to have been in such a primordial state for the first few millionths of a second after the Big Bang. The properties of such a phase are key issues for Quantum Chromo Dynamics, the understanding of confinement-deconfinement and chiral phase transitions. For this purpose we are carrying out a comprehensive study of the hadrons, electrons, muons and




**AliRoot Core** f5d51d7 (f5d51d7)

Main Page
Related Pages
Modules
Namespaces
Classes
Files
Search

- ▼ AliRoot Core
  - ▼ MUON code documentation
    - How to check that your aliroot is w
    - How to check that your aliroot is w
    - ▶ MUON Simulation
    - ▶ **MUON Reconstruction**
    - ▶ MUON Data definition and access
    - ▶ MUON Tracker visualisation progr
    - ▶ MUON Event display
    - ▶ MUON Evaluation
    - ▶ MUON The Software testing on Cos
    - ▶ MUON Fast simulation
    - ▶ MUON Raw data
    - ▶ MUON Mapping
    - ▶ MUON Tracking DA
    - ▶ MUON Trigger DA
    - ▶ MUON Calibration
    - ▶ MUON Offline Calibration and Align
    - ▶ MUON Geometry
    - ▶ MUON Trigger
    - ▶ ALICE FMD Off-line code

### MUON Reconstruction

The reconstruction is a multistage process, driven by the **AliMUONTracker** and **AliMUONReconstructor** classes via the **AliReconstruction** class, which is divided into three parts:

- the digitization of the electronic response,
- the clustering of the digits to locate the crossing point of the muon with the chamber,
- the tracking to reconstruct the trajectory of the muon in the spectrometer from which we can extract the kinematics.

All the adjustable options and parameters used to tune the different part of the reconstruction are handled by the class **AliMUONRecoParam**.

### Digitization

- We read the RAW data, convert them (convert them back for simulated data) to digit (object inheriting from **AliMUONVDigit** stored into containers inheriting from **AliMUONVDigitStore**). This conversion is performed by the class **AliMUONDigitMaker**.
- We calibrate the digits, via **AliMUONDigitCalibrator**, by subtracting pedestals and multiplying by a constant gain. All the calibration parameters (pedestals, HV) are read from the OCDB and stored into **AliMUONCalibrationData** objects.
- We create the status of the digit (e.g. pedestal higher than maximum or HV switched off), using

# KDE

- ▶ Thought it might be interesting to look at a big opensource project
- ▶ They seem to use a lot of homegrown tools for their documentation
- ▶ For example, I think their [wiki](#) is custom
- ▶ They do, however, use [doxygen](#) (and lxr) for code documentation

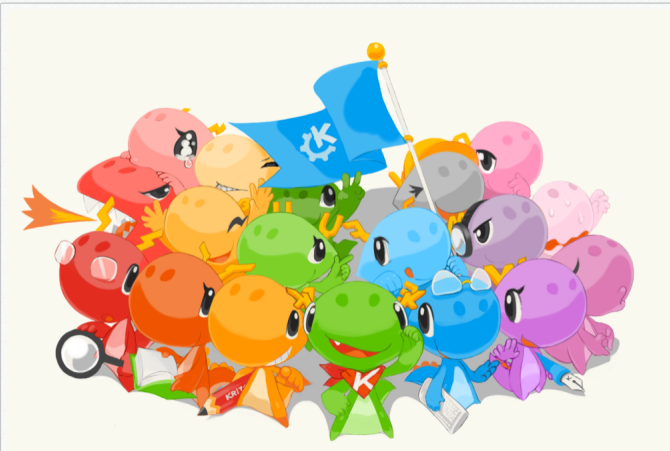
KDE Community Wiki

Search

Main page Discussion View View source History

## Main Page

This page contains changes which are not marked for translation.



Konqi and the KDE community

Share your Knowledge

NAVIGATION

- Main page
- Community portal
- Current events
- Recent changes

CONTRIBUTOR HELP PAGES

- Tasks and Tools
- Modify a page
- Add new content
- Page elements
- Typographical guidelines
- More markup help

TOOLS

- What links here
- Related changes
- Special pages
- Printable version
- Permanent link
- Page information

## KApidox

Search for...

### Introduction

This framework contains scripts and data for building API documentation (dox) in a standard format and style.

The Doxygen tool is used to do the actual documentation extraction and formatting, but this framework provides a wrapper script to make generating the documentation more convenient (including reading settings from the target framework or other module) and a standard template for the generated documentation.

### Dependencies

#### Required

Python 2 or 3 is required to run the scripts. Whichever version of python you use needs to have the jinja2 and yaml (or pyyaml) modules.

The following command should install them for the current user:

```
pip install --user PyYAML jinja2
```

Of course, you need Doxygen!

#### Optional

Doxyqml and doxypypy might be needed to let doxygen document respectevly qml and python sources.

#### Quick Links

- Main Page
- Namespace List
- Namespace Members
- Alphabetical List
- Class List
- File List
- Dependencies
- Related Pages

#### Class Picker

-- Choose --

#### About

Scripts and data for building API documentation (dox) in a standard format and style

**Maintainer**  
Olivier Churlaud

**Supported platforms**  
FreeBSD, Linux, MacOSX, Windows

**Community**  
IRC: #kde-devel on

# SUMMARY OF TOOL USE

This is a bit of a guess in many cases...

Treat as illustrative rather than authoritative!

Project	KDE	Gaudi	Linear collider	Supernemo	Belle II	LHCb	CMS	Alice
Webpages	html	html	github pages, html	Jekyll		twiki, html	twiki, html	github pages
Code	Doxygen + LXR	Doxygen	Doxygen	Doxygen	(filtered) Doxygen Sphinx	?	Doxygen	Doxygen
User guides	Userbase <a href="#">wiki</a>	Doxygen	github, hugo	Doxygen	confluence wiki	gitbook	github pages, twiki	github pages, Doxygen

## HIGHLIGHTS FROM DISCUSSIONS

- ▶ No one had an alternative to using doxygen for C++ code, but no one I spoke to was completely happy with it
- ▶ We could add Breathe to generate Sphinx from doxygen xml, but this would slow down our documentation generation, and it isn't obvious its output is better
- ▶ One generate complaint of doxygen is that it doesn't make it easy to structure the code - it basically dumps everything it is given into one massive list, which doesn't really work for Athena
  - ▶ We get around this to some extent by using packagedocs (our replacement for mainpages). Not ideal, because there is no automated link from the class to the @page (packagedoc)
- ▶ Sphinx requirements more configuration but "that seems to result in documentation which is actually readable (as opposed to an alphabetic index of classes)"
- ▶ "The drawback is that users need to learn yet another markup language (reStructuredText)."

## SUMMARY

- ▶ We seem to be using similar tools to everyone else, (and aren't doing too badly with our documentation - relatively speaking - from what I can see)
- ▶ Could try Hugo instead of Jekyll for static websites, (but Jekyll doesn't really seem slow to me)
- ▶ Almost everyone uses Doxygen for code documentation, and yet no one I spoke to is completely happy with it
  - ▶ Main complaints relate to problems navigating/understanding large codebase
  - ▶ Occasional Doxygen bugs (maybe Doxygen-clang will help?)
  - ▶ I think we could learn a bit from CMS about integrating it with git, and making it less monolithic
- ▶ Gitbook looks nice (and LHCb startkit is extremely good IMO), but I don't think nice enough for us to tie ourselves into a commercial product
- ▶ Wasn't able to look too much at other wikis (protected behind firewalls)