Order versus Chaos

Software Organisation in a Larger Project

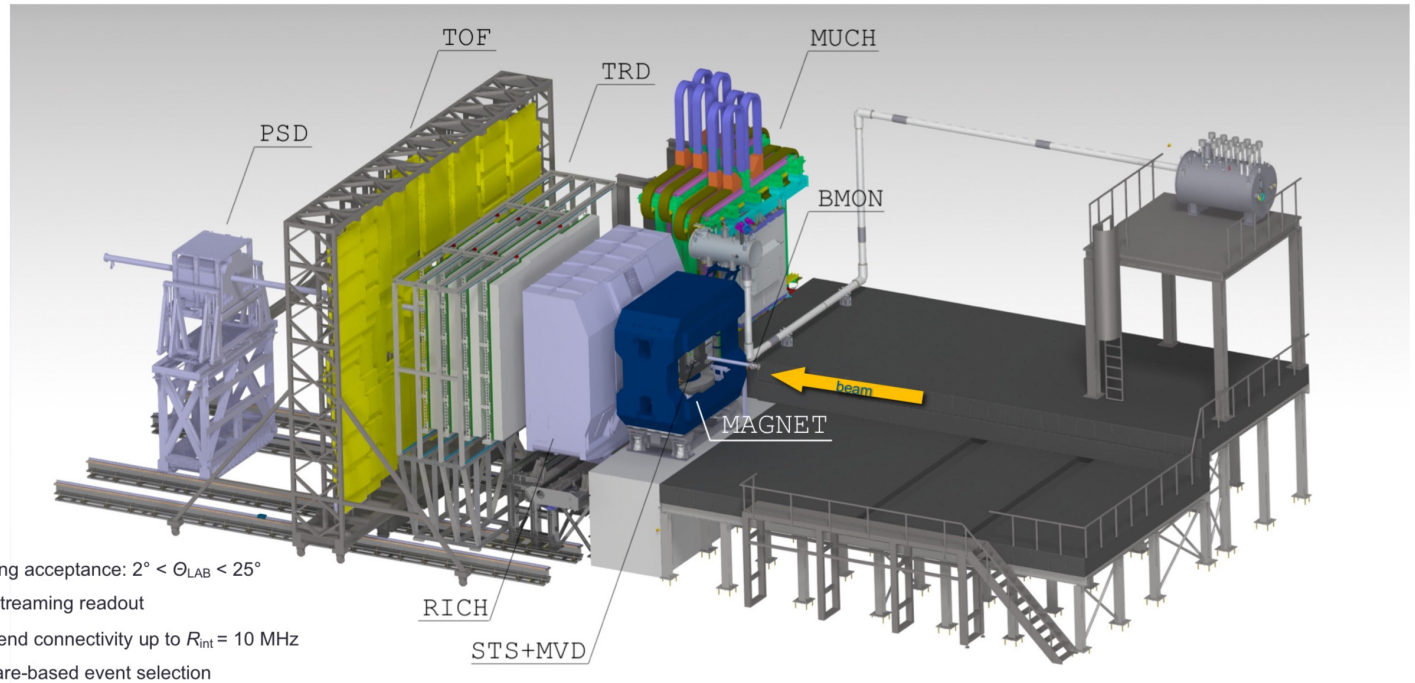
V. Friese
GSI C++ User Group
11 June 2025

Disclaimer

This is not about to tell you how to organise larger software projects.

This is to ask for advise and discuss and learn best practices how to organise larger software projects - on the example of cbmroot.

Setting the Stage: The CBM Experiment

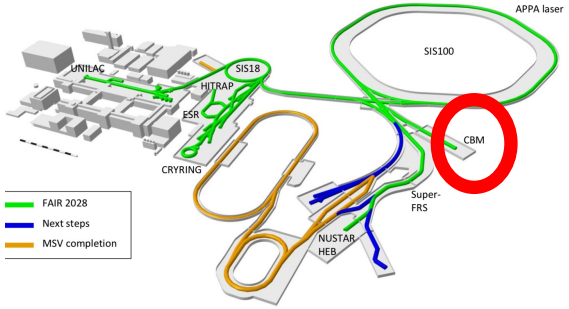


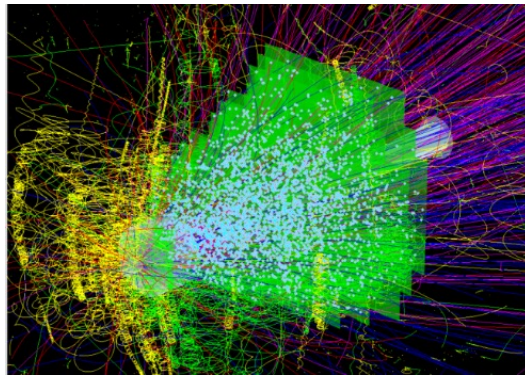
Tracking acceptance: $2° < \Theta_{LAB} < 25°$

Free streaming readout

Front-end connectivity up to R_{int} = 10 MHz

Software-based event selection

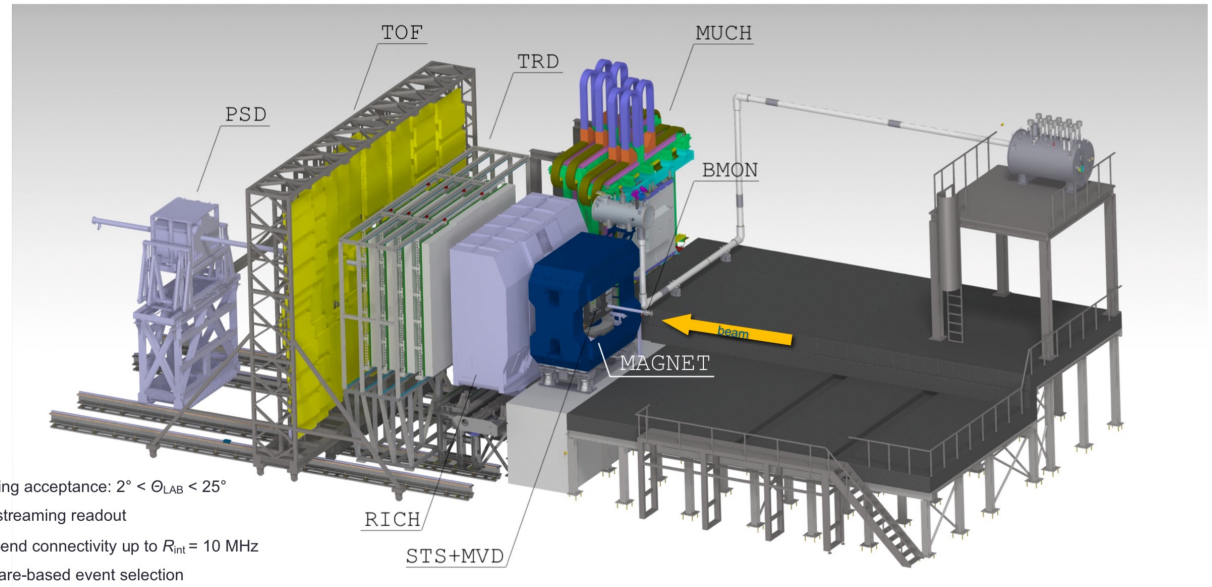




- A heavy-ion experiment at FAIR, to operate from 2028 on with beams from SIS-100
- Fixed-target; high interaction rates; no hardware trigger
- Needs online data selection in software.

Mission

Establish the software needed to run the experiment and extract physics results from its data.



Many facets:

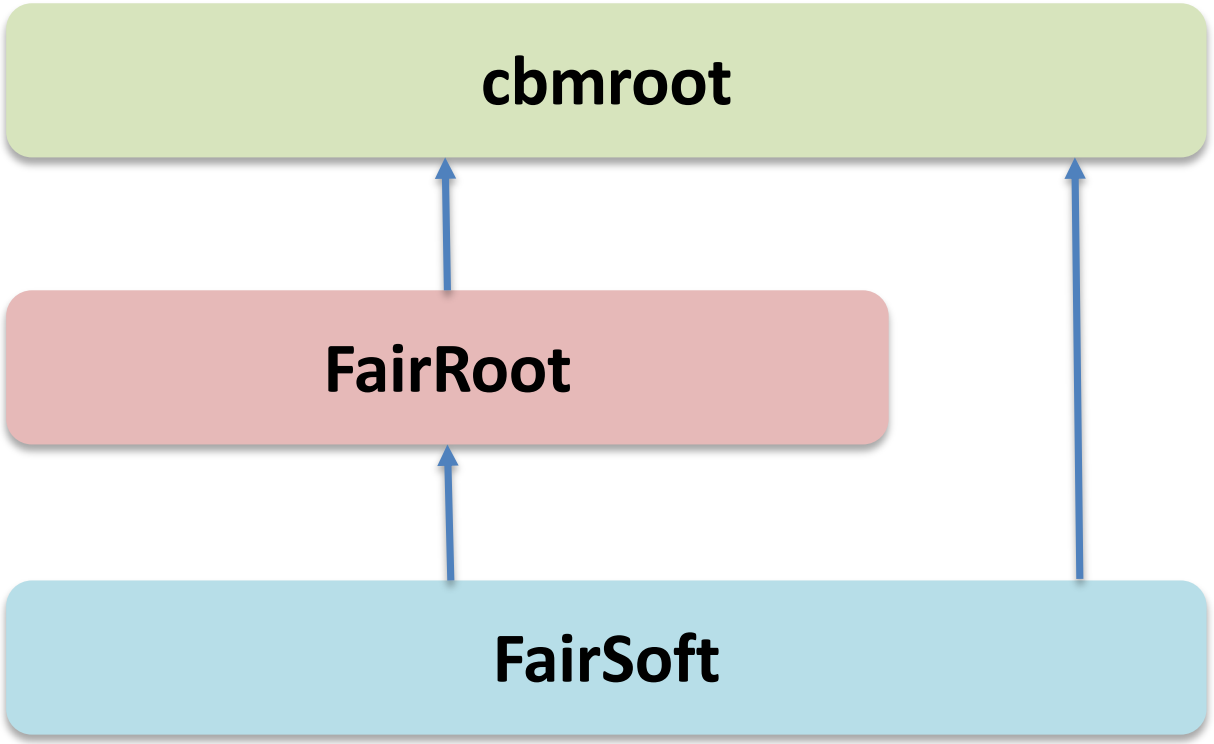
- Simulate the experiment
- Configure and control the experiment
- Digest raw data in real time to the level of physics-based data selection.
- Reconstruct events and tracks from the free-streaming data.
- Implement the offline physics analysis software.

About 500 GB/s raw data from experiment

Several GB/s archival rate

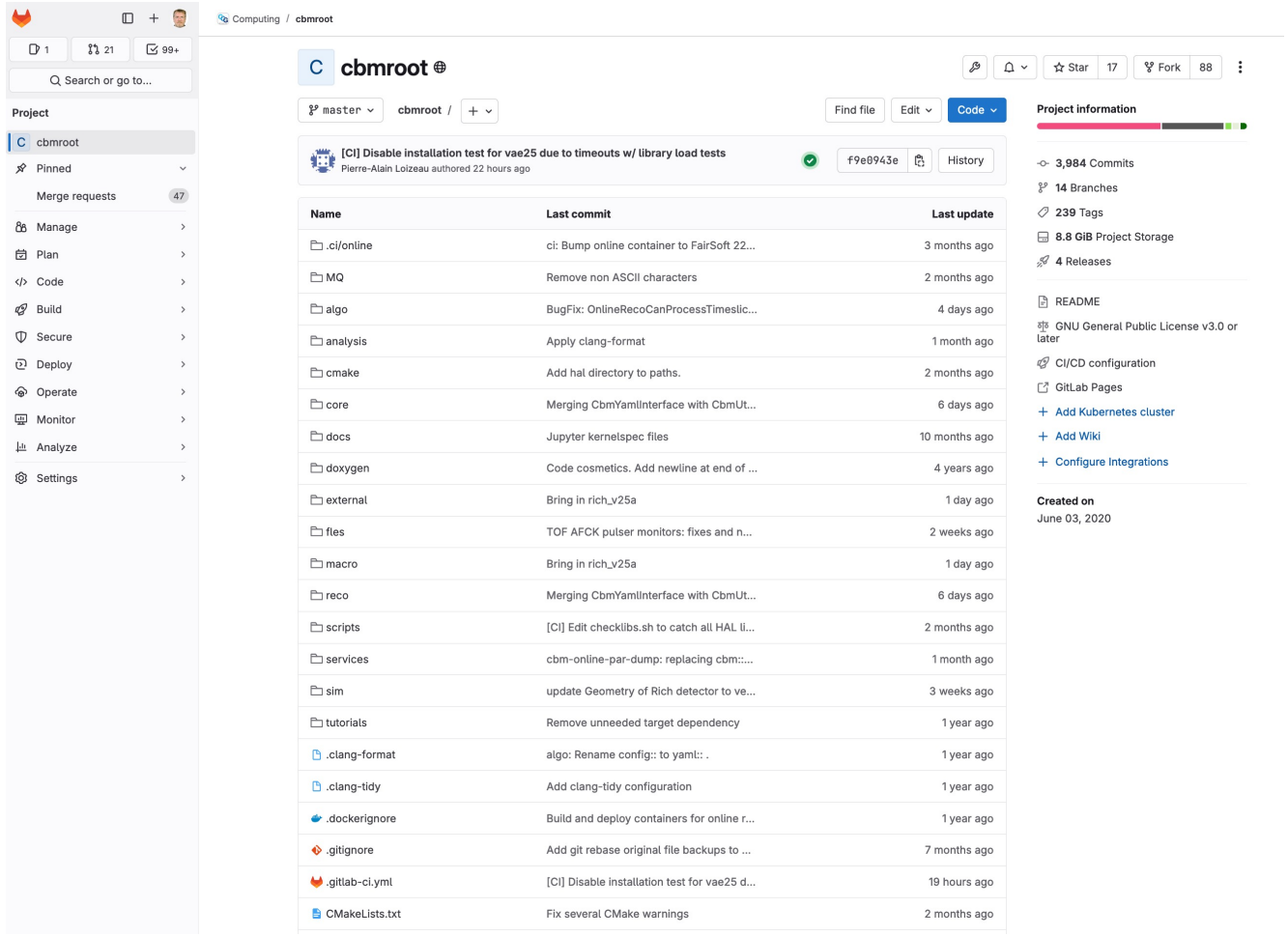
O(10) PB / year archived data volume

cbmroot



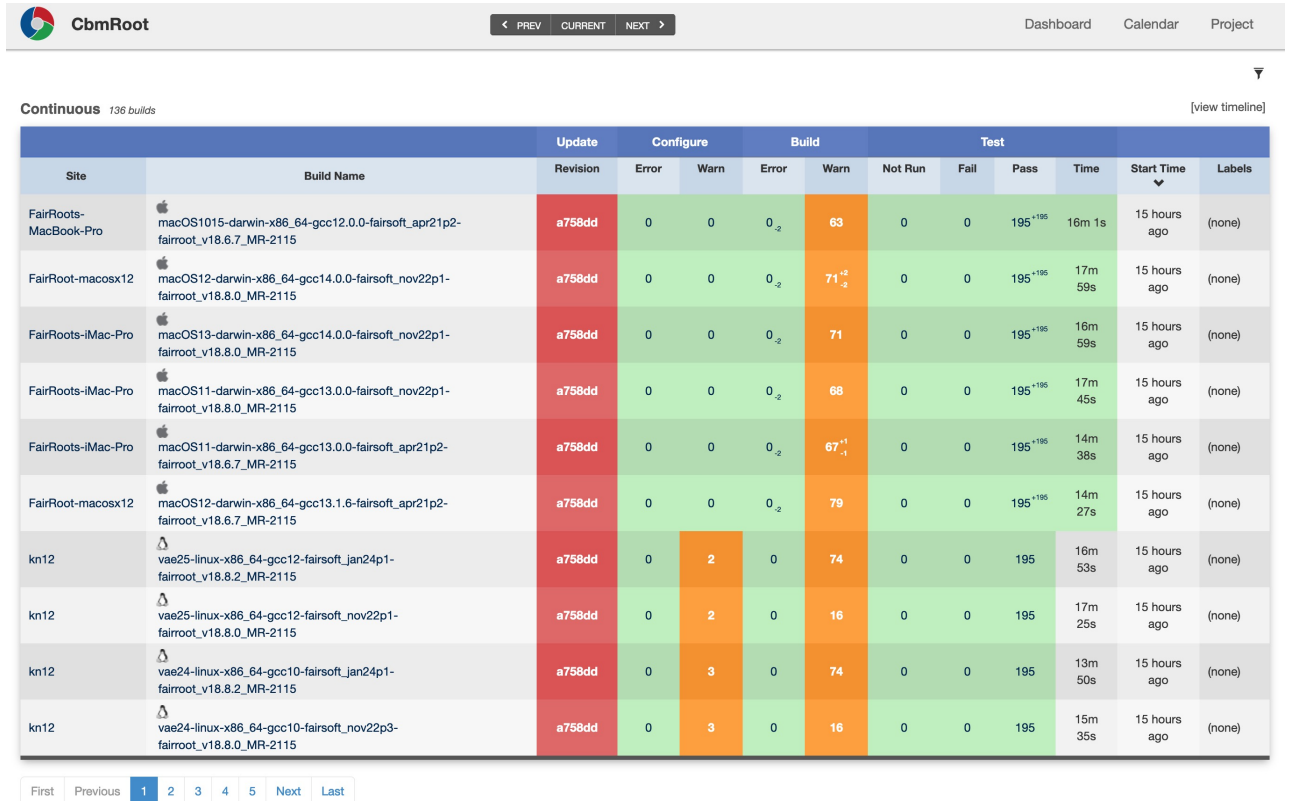
- A long history of about two decades
- Starting point was simulation and offline computing.
- Builds on top of FairRoot, using ROOT as a platform
- A number of external packages (FairSoft): ROOT, GEANT3/4, BOOST, ...
- Online data processing (HPC) in the focus since about 5 years.
- Using mCBM as test bed.

Cbmroot: Current Organisation



- One single project / repository
- Hosted in gitlab (CI/CD)
- Almost completely C++ code
- 300 k LOC, about 100 shared libraries
- Comprises a variety of functionality (online / offline)
- Long history, many contributors: strong variation in level of coding and C++ standards
- Constantly evolving: started as typical offline framework, now more and more online and performance aspects introduced.

Cbmroot: Current Organisation



CbmRoot PREV | CURRENT | NEXT Dashboard Calendar Project

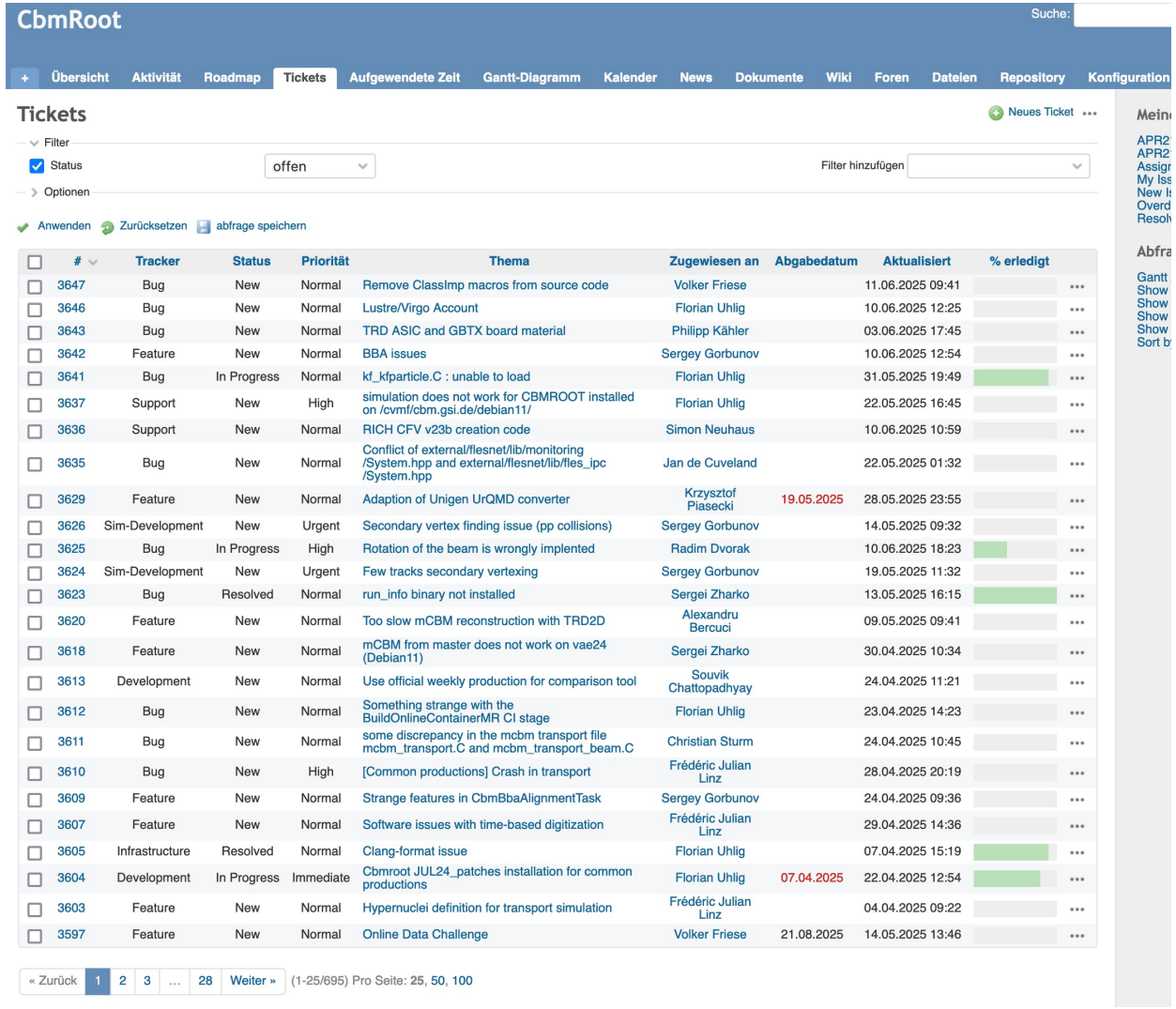
Continuous 136 builds [view timeline]

Site	Build Name	Update Revision	Configure Error	Configure Warn	Build Error	Build Warn	Test Not Run	Test Fail	Test Pass	Test Time	Start Time	Labels
FairRoots-MacBook-Pro	macOS1015-darwin-x86_64-gcc12.0.0-fairsoft_apr21p2-fairroot_v18.6.7_MR-2115	a758dd	0	0	0_{-2}	63	0	0	195^{+195}	16m 1s	15 hours ago	(none)
FairRoot-macosx12	macOS12-darwin-x86_64-gcc14.0.0-fairsoft_nov22p1-fairroot_v18.8.0_MR-2115	a758dd	0	0	0_{-2}	71^{+2}_{-2}	0	0	195^{+195}	17m 59s	15 hours ago	(none)
FairRoots-iMac-Pro	macOS13-darwin-x86_64-gcc14.0.0-fairsoft_nov22p1-fairroot_v18.8.0_MR-2115	a758dd	0	0	0_{-2}	71	0	0	195^{+195}	16m 59s	15 hours ago	(none)
FairRoots-iMac-Pro	macOS11-darwin-x86_64-gcc13.0.0-fairsoft_nov22p1-fairroot_v18.8.0_MR-2115	a758dd	0	0	0_{-2}	68	0	0	195^{+195}	17m 45s	15 hours ago	(none)
FairRoots-iMac-Pro	macOS11-darwin-x86_64-gcc13.0.0-fairsoft_apr21p2-fairroot_v18.6.7_MR-2115	a758dd	0	0	0_{-2}	67^{+1}_{-1}	0	0	195^{+195}	14m 38s	15 hours ago	(none)
FairRoot-macosx12	macOS12-darwin-x86_64-gcc13.1.6-fairsoft_apr21p2-fairroot_v18.6.7_MR-2115	a758dd	0	0	0_{-2}	79	0	0	195^{+195}	14m 27s	15 hours ago	(none)
kn12	vae25-linux-x86_64-gcc12-fairsoft_jan24p1-fairroot_v18.8.2_MR-2115	a758dd	0	2	0	74	0	0	195	16m 53s	15 hours ago	(none)
kn12	vae25-linux-x86_64-gcc12-fairsoft_nov22p1-fairroot_v18.8.0_MR-2115	a758dd	0	2	0	16	0	0	195	17m 25s	15 hours ago	(none)
kn12	vae24-linux-x86_64-gcc10-fairsoft_jan24p1-fairroot_v18.8.2_MR-2115	a758dd	0	3	0	74	0	0	195	13m 50s	15 hours ago	(none)
kn12	vae24-linux-x86_64-gcc10-fairsoft_nov22p3-fairroot_v18.8.0_MR-2115	a758dd	0	3	0	16	0	0	195	15m 35s	15 hours ago	(none)

First | Previous | 1 | 2 | 3 | 4 | 5 | Next | Last

- Code quality management in CDash
- About 200 tests (continuous / nightly / weekly)
- Mostly checking code integrity (flawless compilation and execution)
- Automated check of results started, but is in infancy yet.
- Test coverage about 25 %.

Cbmroot: Current Organisation



- Using Redmine as communication platform for developers and users
- Tickets / Planning / Documentation (Wiki) / Fora

Experiences / Issues / Problems

- Hosting and maintain such large a system becomes increasingly difficult and expensive with size.
- The necessity to support legacy code impedes development with modern coding standards.
- Large variety of coding skills among the contributors.
- Difficult to introduce / enforce coding standards and conventions.
- Library dependences become hard to manage (e.g, avoid circular dependences)
- Dependence from external packages introduces maintenance efforts (e.g., when updating).
- Available workforce is < 50% of what you would like to have.

Disclaimer (2)

So much to what we have.

Now come the questions: how shall we proceed? Recommendations, best practices, etc...

Goal is to have a sustainable code base comprising the needs for operating the experiment in some three years from now.

Q1: A Single Repository or Multiple Ones?

Pro single:

- Synergy in maintain the development infrastructure (gitlab, CDash, ...)
- Easier system integration and control of coherence.
- Easier to see interfaces in the IDE.
- No need to define dependencies (interfaces, versions) between projects.

Pro multiple:

- Better separation and organisation of development.
- Easier to introduce different levels of coding standards / conventions.
- Granular decision which external packages to link to.
- Developments in active areas are not decelerated by those in less active areas.
- Better overview and management.

Q2: How to Define Libraries

- When does it make sense to introduce a new, separate library?
- Limiting cases:
  - One single library for everything
  - One library for each compilation unit

Q3: How to Organise Code

- One directory per library?
- Should the directory of a library have subdirectories?
- Should the installation be flat (single include directory) or have the same hierarchy structure as the code base?
- Connected to that: Is it advisable to have non-unique file names? (i.e., the path is needed to qualify the class)

- Example:
  - Non-unqiue: cbmroot/algo/detectors/sts/Hitfinder.cxx
  - Unique: Cbmroot/reco/detector/sts/CbmStsAlgoFindHits.cxx

Q4: Namespaces

- When does it make sense to introduce a (sub-)namespace?
- Should a namespace be connected to a library? Or a directory?

Q4: Namespaces

- When does it make sense to introduce a (sub-)namespace?
- Should a namespace be connected to a library? Or a directory?

Q5: Coding Conventions

- How much sense do coding conventions make?
- Don’t they solely rely on taste? E.g., CbmStsAlgoAnaCluster vs cbm_sts_algo_ana_cluster
- If I choose to have some: How do I enforce them?
- Too strict regulations decelerate the development process.
- Taste changes with time.... But adopting to new standards is a huge enterprise...

Q6: Anything Else...

Do you have more aspects / recommendation which I did not touch here?