Lesson learned: ECCE software during proposal stage

Jin Huang

Brookhaven National Lab

For ECCE Computing Team and Simulation WG

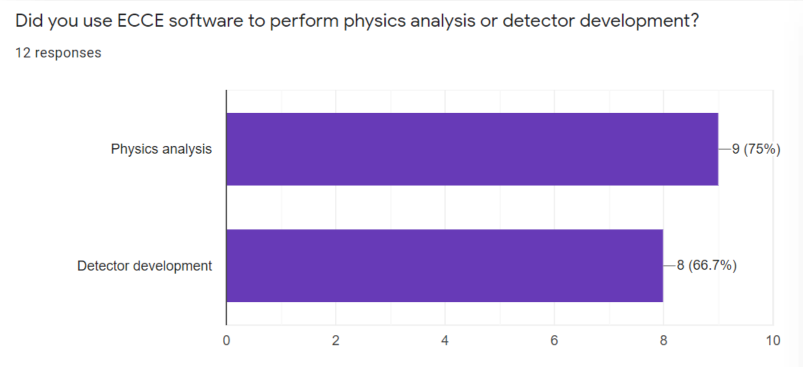
Brookhaven National Laboratory

ECCE software strategy

- Software is a tool to get physics, and it evolves with the experiment
- During the proposal preparation, ECCE determined the best option for a software tool stack, given the schedule constraints and experience of the primary developers, was Fun4all-EIC
  - Fun4All-EIC was validated and widely used since early 2010s for EIC studies
  - Gluing together community-wide sim+reco packages: Geant4, ACTS, GenFit, FastJet, etc.; comprehensive truth evaluation chain; comprehensive CI quality checks
- Post-proposal stage, ECCE will adopt and develop new tool/stack working toward the realization of the EIC detector and processing of the real data
  - Natural evolution of software for conceptual experiment. Expect ECCE software would look quite different in the next decade
- Many opportunities to explore synergies in software development

Software usages and survey

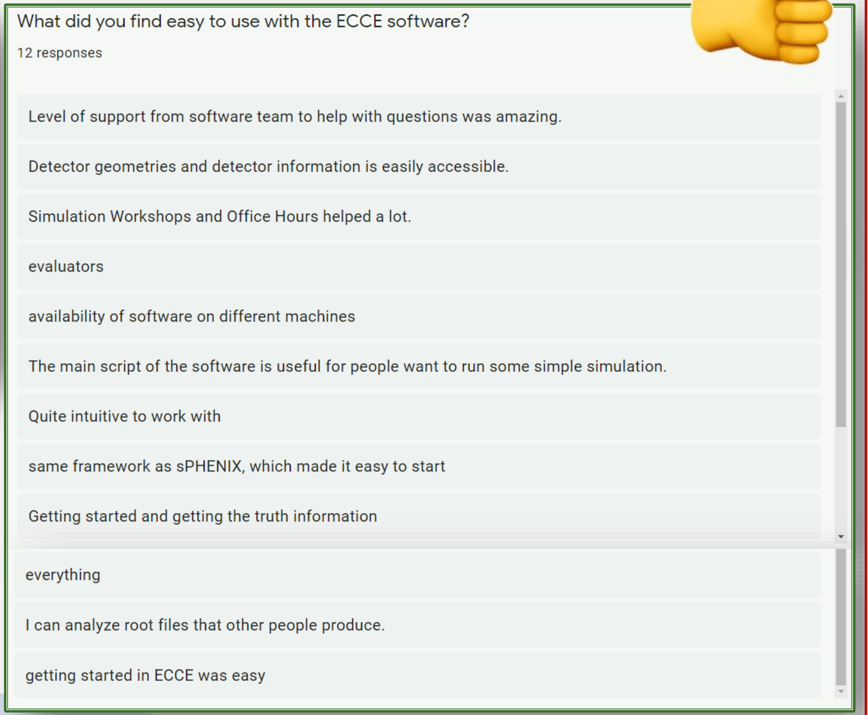
- Built for full detector simulation + reconstruction studies
- Foundation for ECCE proposal and 1000-page internal notes
- This talk: lesson learned from the past months and user survey in Jan-2022 (continue from Dave's talk)



Did you use ECCE software to perform physics analysis or detector development?

12 responses

Used in both detector development and physics benchmark, including many collaborators doing both



What did you find easy to use with the ECCE software?

12 responses

Level of support from software team to help with questions was amazing.

Detector geometries and detector information is easily accessible.

Simulation Workshops and Office Hours helped a lot.

evaluators

availability of software on different machines

The main script of the software is useful for people want to run some simple simulation.

Quite intuitive to work with

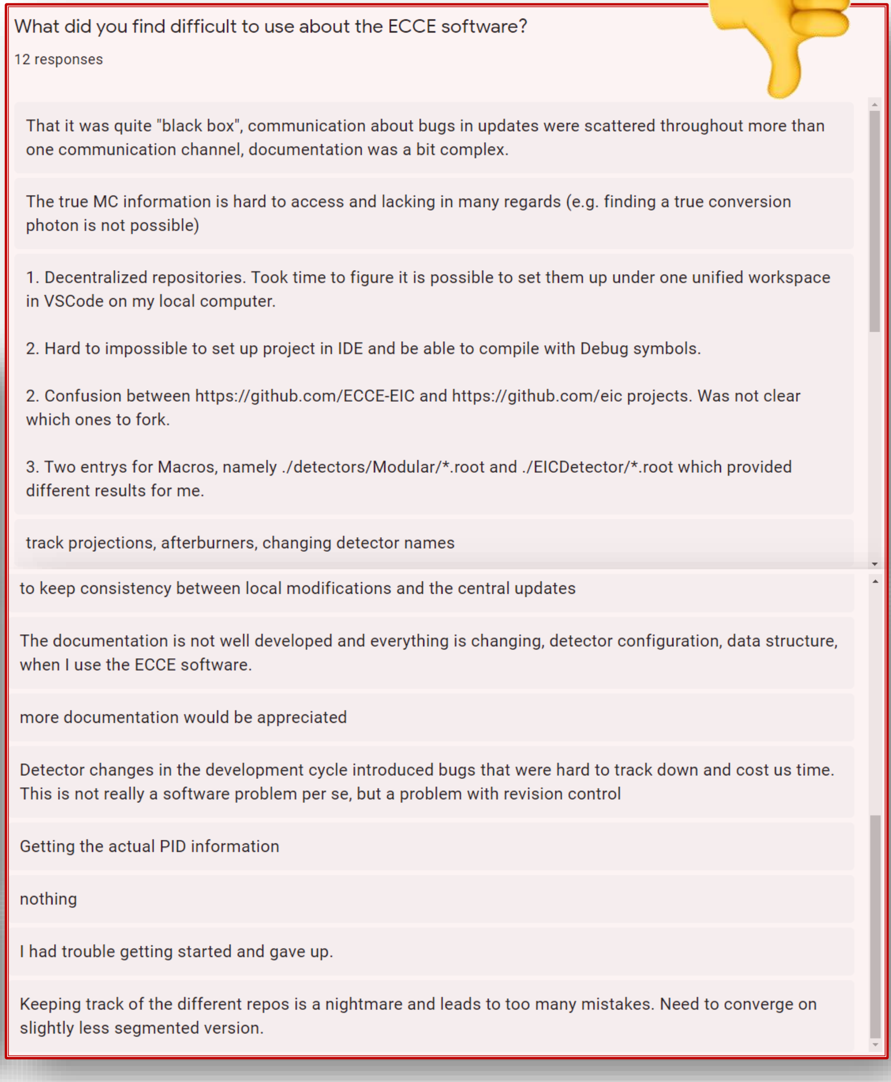
same framework as sPHENIX, which made it easy to start

Getting started and getting the truth information

everything

I can analyze root files that other people produce.

getting started in ECCE was easy



What did you find difficult to use about the ECCE software?

12 responses

That it was quite "black box", communication about bugs in updates were scattered throughout more than one communication channel, documentation was a bit complex.

The true MC information is hard to access and lacking in many regards (e.g. finding a true conversion photon is not possible)

1. Decentralized repositories. Took time to figure it is possible to set them up under one unified workspace in VSCode on my local computer.

2. Hard to impossible to set up project in IDE and be able to compile with Debug symbols.

2. Confusion between https://github.com/ECCE-EIC and https://github.com/eic projects. Was not clear which ones to fork.

3. Two entrys for Macros, namely ./detectors/Modular/*.root and ./EICDetector/*.root which provided different results for me.

track projections, afterburners, changing detector names

to keep consistency between local modifications and the central updates

The documentation is not well developed and everything is changing, detector configuration, data structure, when I use the ECCE software.

more documentation would be appreciated

Detector changes in the development cycle introduced bugs that were hard to track down and cost us time. This is not really a software problem per se, but a problem with revision control

Getting the actual PID information

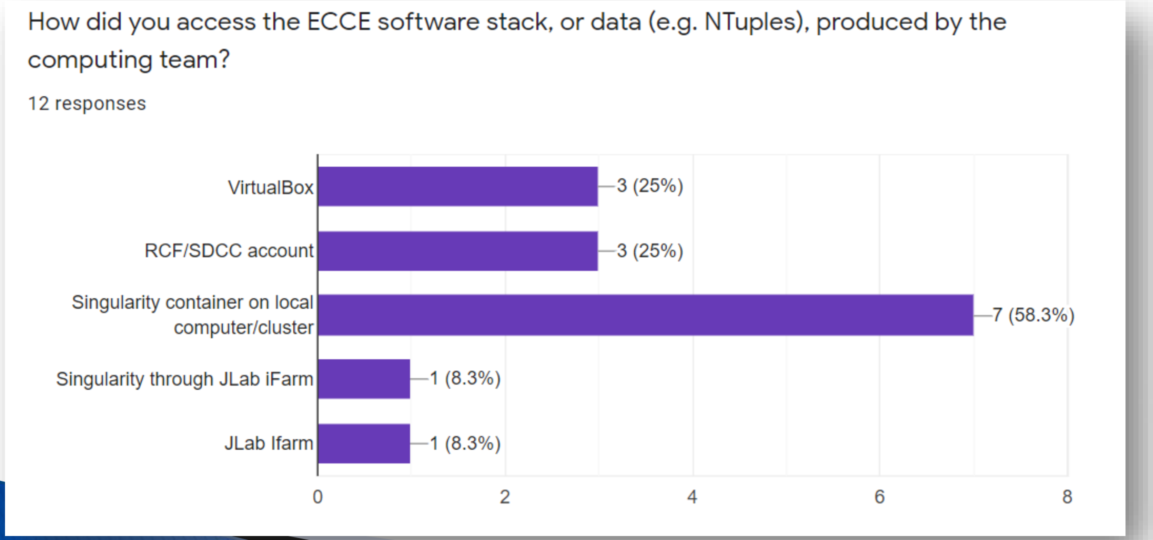
nothing

I had trouble getting started and gave up.

Keeping track of the different repos is a nightmare and leads to too many mistakes. Need to converge on slightly less segmented version.

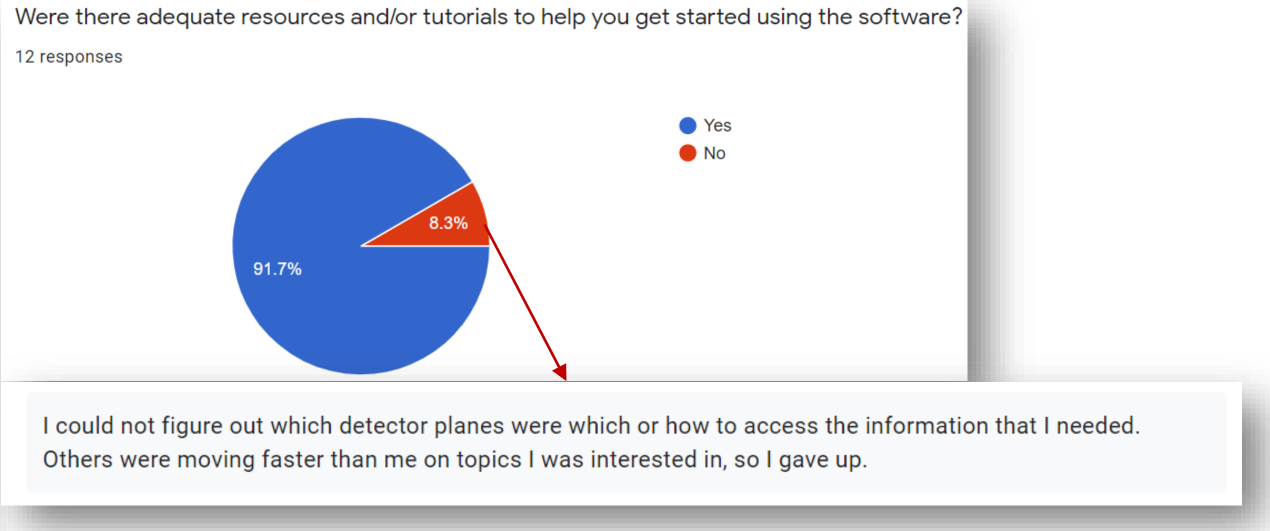
User involvement: what worked 👍

- Four simulation workshops (Apr, May, July, Sept), weekly software office hour, followed with email and chat communication tools
- Support multiple computing sites & laptop use via containerization and Continuous Delivery via CVMFS & Web



How did you access the ECCE software stack, or data (e.g. NTuples), produced by the computing team?

12 responses



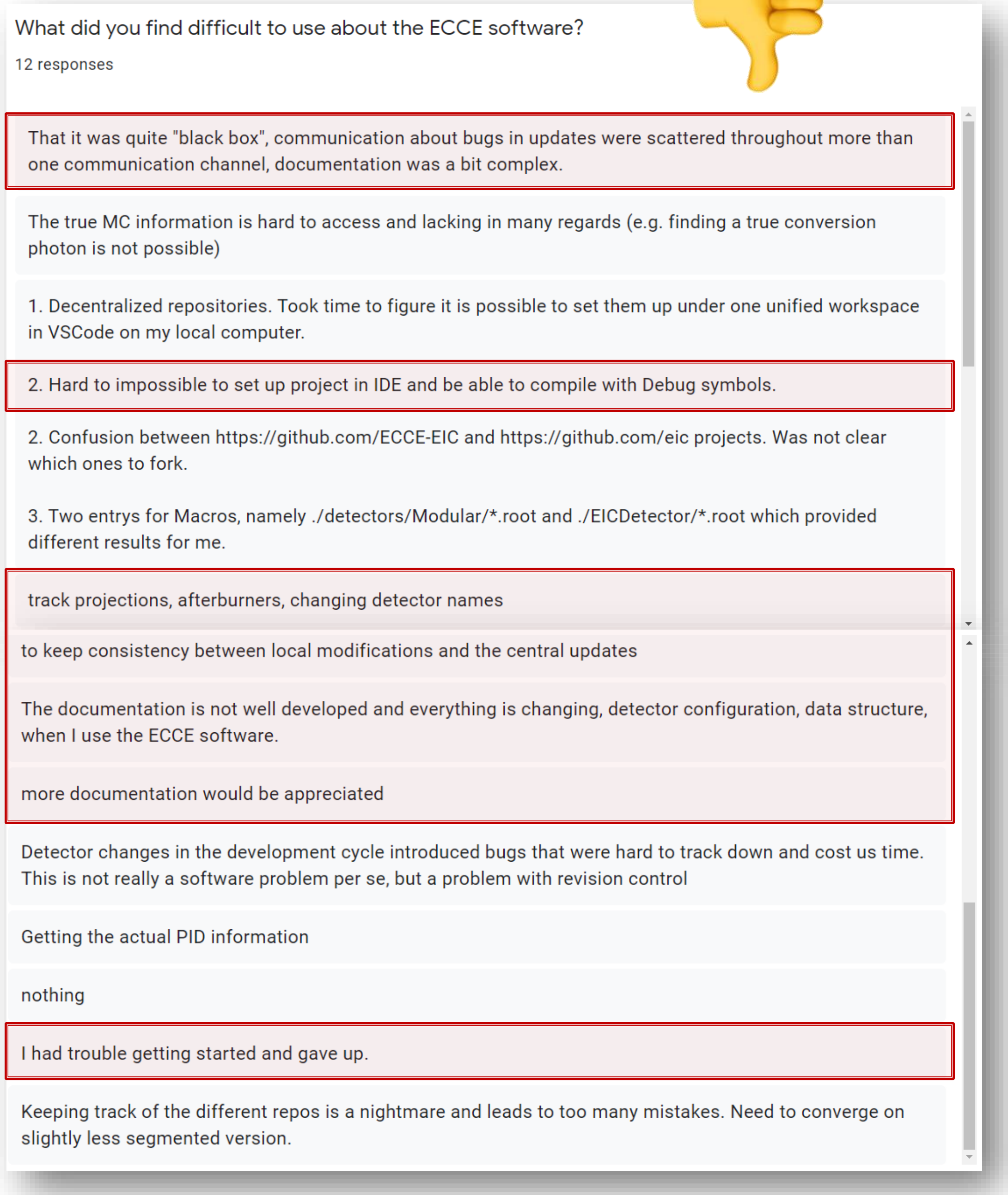
Were there adequate resources and/or tutorials to help you get started using the software?

12 responses

I could not figure out which detector planes were which or how to access the information that I needed. Others were moving faster than me on topics I was interested in, so I gave up.

User involvement : what is lacking

- In general, needed more of:
  - User support
  - Communication
  - Documentation
- In general lack of manpower and time for these tasks



What did you find difficult to use about the ECCE software?

12 responses

That it was quite "black box", communication about bugs in updates were scattered throughout more than one communication channel, documentation was a bit complex.

The true MC information is hard to access and lacking in many regards (e.g. finding a true conversion photon is not possible)

1. Decentralized repositories. Took time to figure it is possible to set them up under one unified workspace in VSCode on my local computer.

2. Hard to impossible to set up project in IDE and be able to compile with Debug symbols.

2. Confusion between https://github.com/ECCE-EIC and https://github.com/eic projects. Was not clear which ones to fork.

3. Two entrys for Macros, namely ./detectors/Modular/*.root and ./EICDetector/*.root which provided different results for me.

track projections, afterburners, changing detector names

to keep consistency between local modifications and the central updates

The documentation is not well developed and everything is changing, detector configuration, data structure, when I use the ECCE software.

more documentation would be appreciated

Detector changes in the development cycle introduced bugs that were hard to track down and cost us time. This is not really a software problem per se, but a problem with revision control

Getting the actual PID information

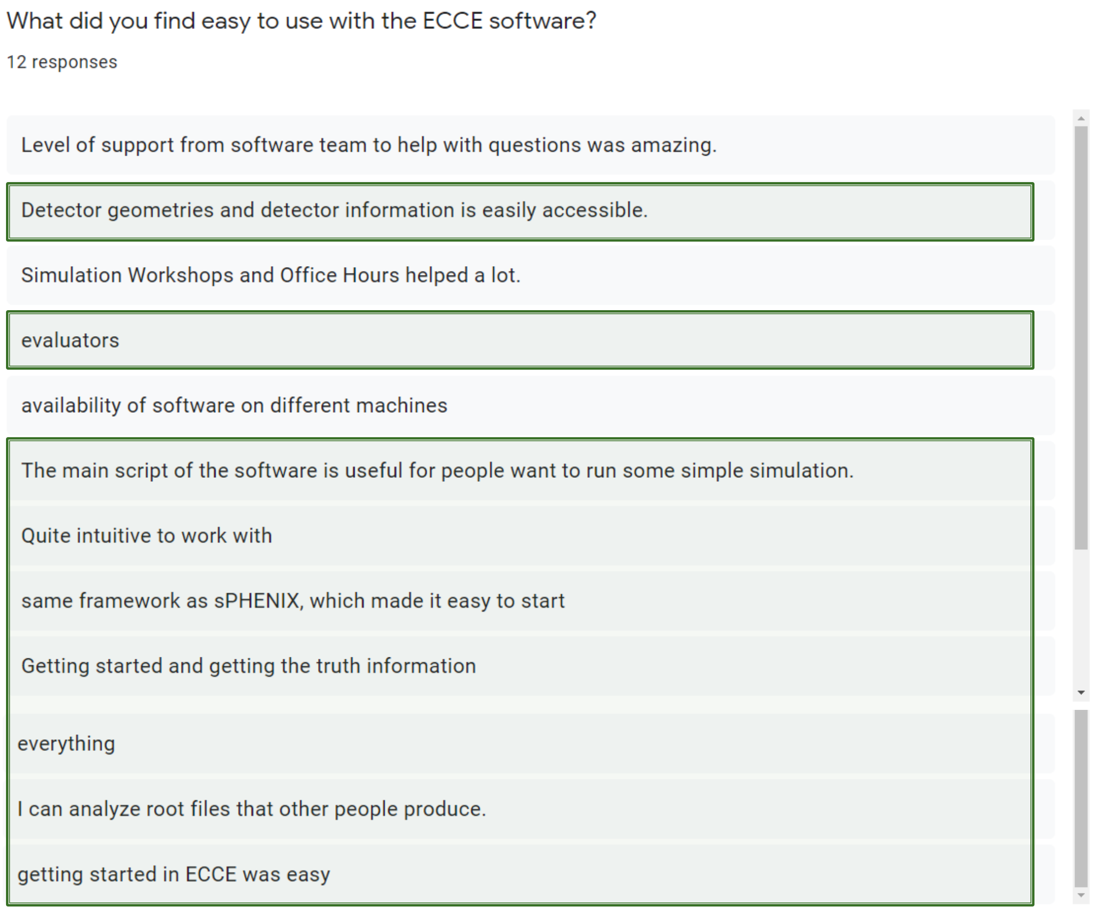
nothing

I had trouble getting started and gave up.

Keeping track of the different repos is a nightmare and leads to too many mistakes. Need to converge on slightly less segmented version.

Software organization: what worked 👍

- Single example steering macro handle Event-gen, beam effects →Simulation →Digitization →Reconstruction →Analysis interface.
- Then user can custom to run part of full chain
- Analysis either via evaluation NTuple generators or analysis modules (later slides)
- Comprehensive Continuous Integration for the core-software
  - Ensure no warning from gcc, clang, scanbuild, cppcheck, and valgrind. QA plots check each stage of simulation and reconstruction
  - e.g. CI report for HCal light collection model



What did you find easy to use with the ECCE software?

12 responses

Level of support from software team to help with questions was amazing.

Detector geometries and detector information is easily accessible.

Simulation Workshops and Office Hours helped a lot.

evaluators

availability of software on different machines

The main script of the software is useful for people want to run some simple simulation.

Quite intuitive to work with

same framework as sPHENIX, which made it easy to start

Getting started and getting the truth information

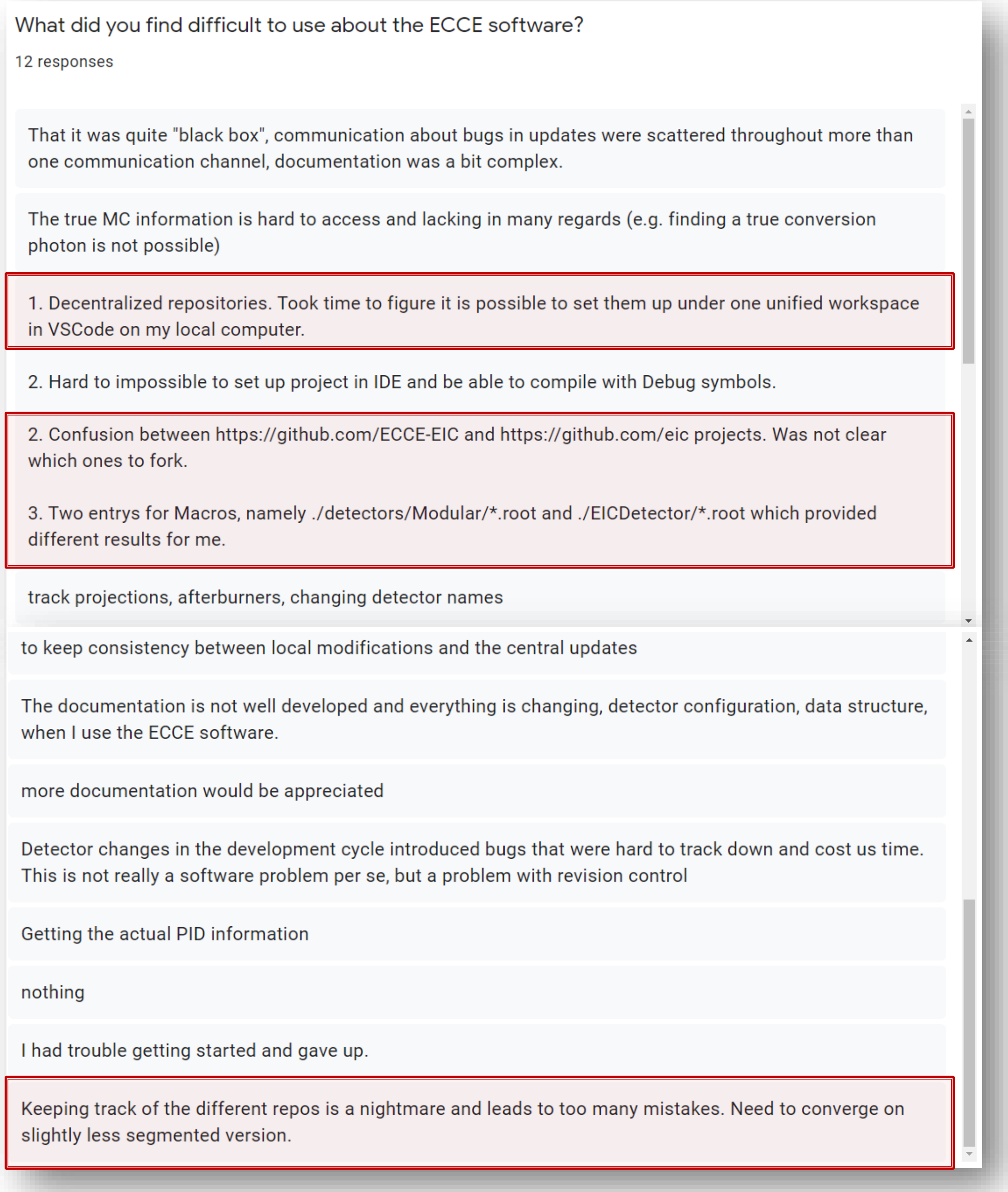
everything

I can analyze root files that other people produce.

getting started in ECCE was easy

Software organization : 👎 what is lacking

- We set out to maintain three ties ofr repositories and builds for EIC general use and for specific experiments: ECCE and sPHENIX.
  - Don’t have manpower to maintain three separate development, so some repositories are shared and mirrored
- This leads to very complex repository structure cross three GitHub organizations.
  - Sources of many confusions
  - Preventing extending CI beyond Core-software
- This is confusing and error-prone, we should avoid this post proposal stage



What did you find difficult to use about the ECCE software?

12 responses

That it was quite "black box", communication about bugs in updates were scattered throughout more than one communication channel, documentation was a bit complex.

The true MC information is hard to access and lacking in many regards (e.g. finding a true conversion photon is not possible)

1. Decentralized repositories. Took time to figure it is possible to set them up under one unified workspace in VSCode on my local computer.

2. Hard to impossible to set up project in IDE and be able to compile with Debug symbols.

2. Confusion between https://github.com/ECCE-EIC and https://github.com/eic projects. Was not clear which ones to fork.

3. Two entrys for Macros, namely ./detectors/Modular/*.root and ./EICDetector/*.root which provided different results for me.

track projections, afterburners, changing detector names

to keep consistency between local modifications and the central updates

The documentation is not well developed and everything is changing, detector configuration, data structure, when I use the ECCE software.

more documentation would be appreciated

Detector changes in the development cycle introduced bugs that were hard to track down and cost us time. This is not really a software problem per se, but a problem with revision control

Getting the actual PID information

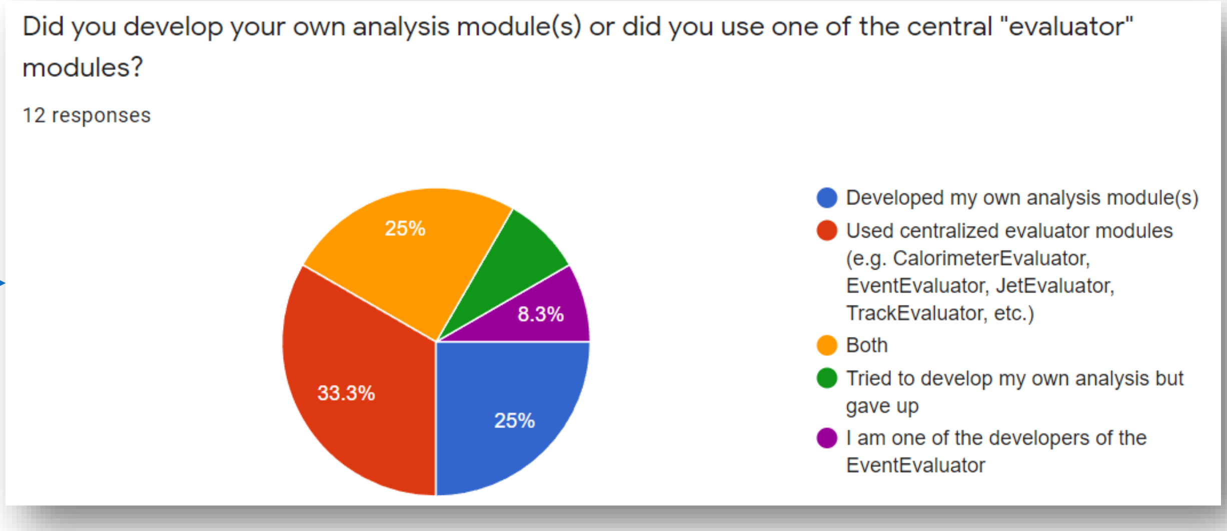
nothing

I had trouble getting started and gave up.

Keeping track of the different repos is a nightmare and leads to too many mistakes. Need to converge on slightly less segmented version.

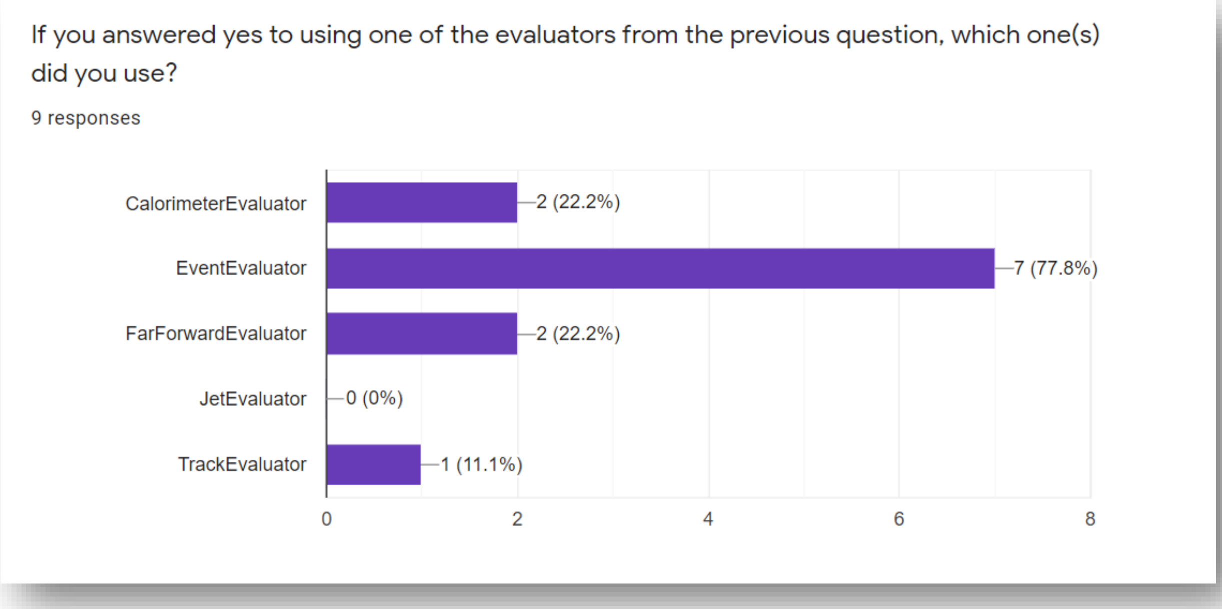
Analyzer interface

- Analyzers could choose to use flat TTree (evaluators) or DST for full access using user-defined analysis modules, or both
- Flat TTree
  - Easy for user to pick up, flexible and fast in post process
  - Hard to use a single tree to adapt for all analysis. External code to QC
- User analysis modules
  - Allow access for all information available in sim+reco. Guaranteed reproducibility. Quick to make a task specific small TTree, e.g. ECCE Centauro jet, AI lepton ID
  - Some learning curve, and require use of one of the ECCE computing sites to process DSTs



Did you develop your own analysis module(s) or did you use one of the central "evaluator" modules?

12 responses



If you answered yes to using one of the evaluators from the previous question, which one(s) did you use?

9 responses

Beyond this talk: software topics for future discussions

- Validating simulation + digitization chain
  - Simulating hadron response in EIC calorimeters is non-trivial
  - Pooling community data for validation, e.g. arXiv:1704.01461v1
- Tracking reconstruction
  - First large-scale application of ACTS outside ATLAS is in Fun4All [arXiv:2103.06703]
  - Adaptation to reliably use in EIC, e.g. curved tracker, pattern recognition tunes, alignment
- Background and radiation field studies
  - EIC data is likely background/noise dominated, challenging to determine e.g. SynRad [CDR Chapter 8]
  - On-going engineering + detector studies
- AI detector optimization, reconstruction, analysis
  - Hot topic these days, driven factor in the ECCE detector design
- Analysis interface: flat tree and full accesses
  - What variable in flat tree would be sufficient for most analysis?
  - How to make the structure flexible for distinct requirement of different analysis, e.g. HF?

Continue to Bill's talk on User experience

Remix credit: Dave Morrison