EsbRoot status and workplan

Budimir Kliček

(heavily discussed with Peter Christiansen)

Software stack

- C++11
- FairRoot – framework to put everything together (see talk by Konstanin)
- ROOT v6 – data model, data persistency, geometry definition
- ROOTVMC – for particle propagation (using Geant3/Geant4)
  - Included in FairRoot
- Genie – beamfiles
  - GiBUU as an option down the road – does not seem too viable anymore
- cmake – the build system
  - Included in FairRoot
- DoxyGen – documentation
- git and SVN – software repository
  - actually we use GitHub which is both SVN and git compatible

FairRoot

- C++11, ROOT v6, ROOTVMC, cmake
  - all included with FairRoot
  - see Peter’s talk

GitHub

- I created a GitHub organization “ESSnuSB”
  - https://github.com/ESSnuSB/
  - I am the only owner; there should be at least another person
- It contains:
  - a repository “EsbRoot”
  - forks of FairSoft and FairRoot
    - this is new, will be used in the future to tweak FairSoft/FairRoot installations
- Can be also used with SVN
  - but do we really want to?
- We must decide who has write permission for the repository
  - for now Guy, Peter and Budimir
  - IMPORTANT: using “pull requests” anyone can propose changes
    - those with write permissions must authorize them

Doxygen

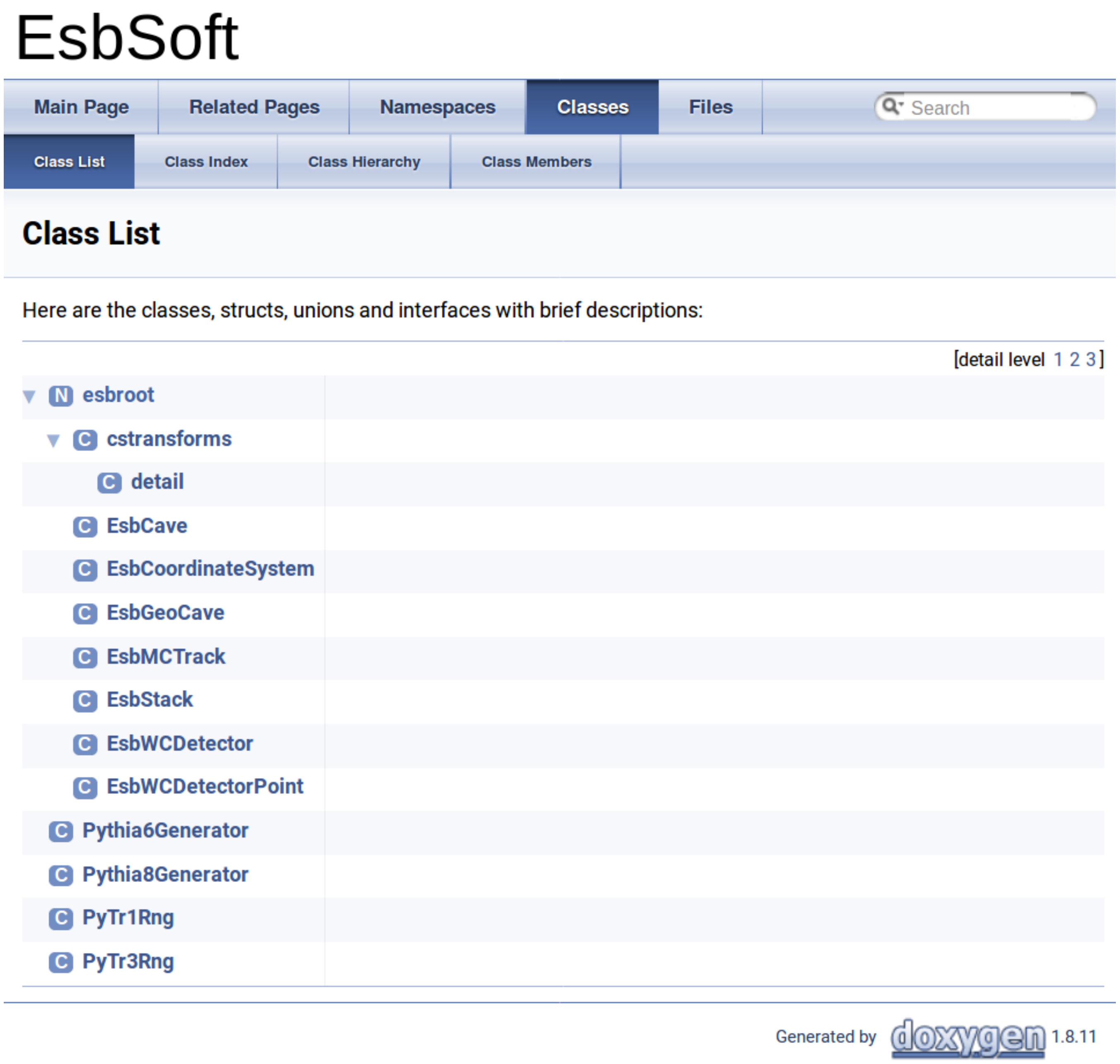
- A program to automatically generate documentation from code
  - can do HTML, Latex, …
- To do it run cmake for EsbRoot with option

  cmake -DBUILD_DOC=ON
  - there is also option BUILD_ONLY_DOC
  - you need to have doxygen installed

    sudo apt-get install doxygen
- HTML documentation will be created in directory
  - [build_dir]/doxygen
- This is only local for now, but:
  - I asked for and received a server at

    essnusb.irb.hr
  - I will set it up to generate and publish doxygen documentation after every push to our GitHub repository
  - will be linked from essnusb.eu and our GitHub page
- Of course, we must annotate our code for this to work
  - see here: http://www.doxygen.nl/manual/docblocks.html

Doxygen



EsbSoft

Main Page | Related Pages | Namespaces | Classes | Files | Search

Class List | Class Index | Class Hierarchy | Class Members

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[detail level 1 2 3]

- ▼ N esbroot
  - ▼ C cstransforms
    - C detail
  - C EsbCave
  - C EsbCoordinateSystem
  - C EsbGeoCave
  - C EsbMCTrack
  - C EsbStack
  - C EsbWCDetector
  - C EsbWCDetectorPoint
- C Pythia6Generator
- C Pythia8Generator
- C PyTr1Rng
- C PyTr3Rng

Generated by doxygen 1.8.11

Goals (TBD)

Need to have	Nice to have
1) full simulation of neutrino interactions: neutrino interaction rate per spill (or p.o.t.), vertex chosen according to cross sections on different materials, proper connection between ND and FD event rate, … Our input to WP5 depends on all of this. 2) partial geometry of water Cherenkov detectors without geometry of PMT's, the PMT digitization will be done by dividing the outer water wall to sectors and using a reasonable model 3) integration of fine-grained near-detector in the framework 4) comparison of our results with WCSim and MEMPHYS	1) complete geometry of all detectors: including PMT's, wls fibers, magnets, caverns, supports, etc.. We should include as much information as we can get in the design study phase (we definitely won't have technical drawings, though) 2) full digitization, i.e. simulation of detector responses (PMT's or any other that we have) 3) automated integration with neutrino beam simulation (i.e. our software asks WP4 software to provide list of neutrinos exiting the target station on the fly) 4) full automated reconstruction of tracks in all detectors, reconstruction of energy and other physical parameters 5) automated production of migration matrices 6) it must be as easy to use as possible (push the button system) 7) something else?

Semi-independent tasks

a) ND and FD particle guns

** need-to-have

1) A trivial gun - choose a list of primary particles manualy (muons, protons, pions, ...) - this is what we have now

2) A neutrino gun from file - read list of pregenerated neutrino interactions from file, choose the vertex coordinate

3) A simple neutrino gun - choose neutrino flavour, NC/CC, momentum and vertex position, receive primary particles

4) A full neutrino gun - neutrino momentum direction, flavour, NC/CC, energy, vertex position are automatically chosen according to beam simulation and interaction probabilities (fully automatic could be nice-to-have, at least semiautomatic is need-to-have)

b) Near and far WC detectors - use ND and FD particle guns

** need-to-have

1) Simple detector geometry, no PMT's, no track reconstruction, record only hits on the walls of water wolume, simple visualisation - we have more or less this now

2) Rough digitization of hits - PMT's are simulated as areas on water volume walls. "PMT" response is simulated from number of cherenkov photons, their timing, and their momenta when reaching the area. First track reconstrction is possible here.

** nice-to-have

3) Full digitization using models of currently available commercial PMT's (from Peter: PMT models change with time)

c) Fine grained detector - uses ND particle gun.

**need-to-have

1) Should be integrated in EsbRoot.

d) Integration of beam simulation to ESSnuSB-soft

** need-to-have **

1) Completely decoupled - provide a list of neutrinos in a file, in TSCS (coordinate system). Need to define the file structure. Provide neutrino energy spectra in form of ROOT histograms.

2) Start of integration - Discuss the possibilities with WP4

** nice-to-have (depending on how difficult it turns out) **

3) Fully integrated - We get a "neutrino gun", a function which provides "next" neutrino on demand. Must be somehow normalized to p.o.t.

Workplan (TBD)

- Before and during the workshop, I think we should start with these tasks:

  1. Add support for Doxygen documentation to GitHub (I'm already doing it)
  2. Coordinate systems implemented as classes with simple interface (in progress)
  3. Bookkeeping and reporting on our activities
  4. Particle guns using Genie

     - after workshop we should have at least point 2. (A neutrino gun from file), start working on point 3. (A simple neutrino gun)
  5. Near and far WC detectors

     - after workshop we should have point 1. (Simple detector geometry, no PMT's, no track reconstruction, record only hits on the walls of water wolume, simple visualisation), start working on point 2. (Rough digitization of hits)
  6. Fine grained detector

     - start looking how to integrate it to EsbSoft
  7. Integration of beam simulation to EsbSoft

     - after workshop we should have point 1. (Completely decoupled - provide a list of neutrinos in a file), discuss possibilities in point 2. (discussion)

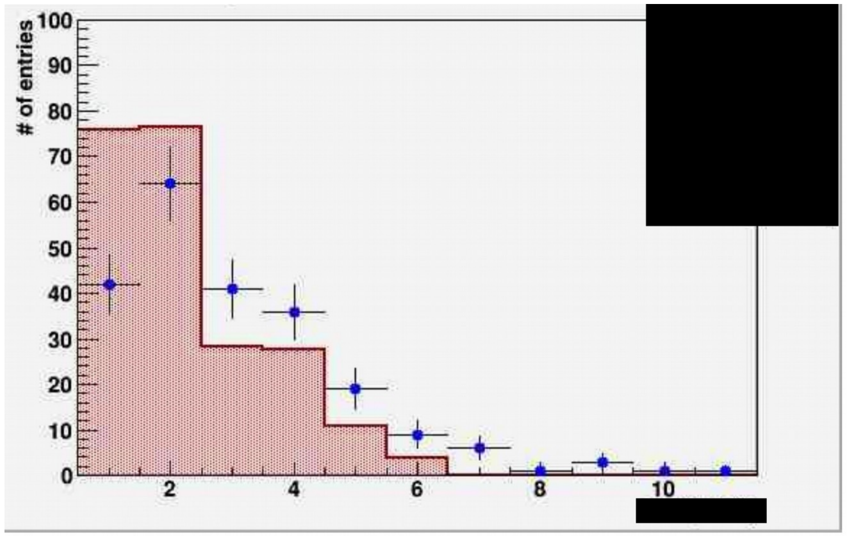
- We must decide who is doing what!

Code quality

- Code quality = physics quality
  - prototyping is inevitable, but in the end we must have well documented and clean code
- We need:
  - good naming conventions
  - defined coding practices
  - use tools that are available to us – GitHub, doxygen, FairRoot
  - regular meetings (once a month?)
  - limited code review?

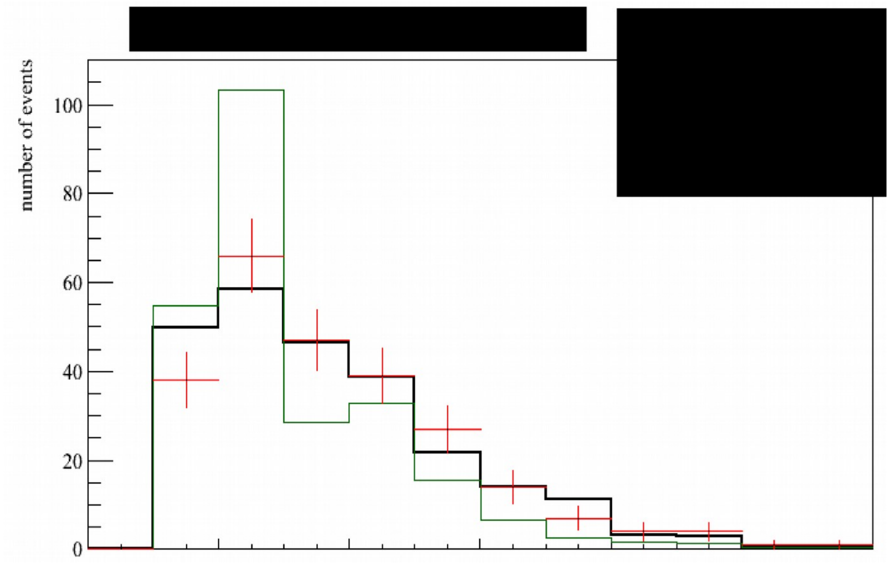
Why we must be careful

Example from another experiment, data/MC comparison ~ 6 years ago



[redacted]
[redacted]

BEFORE



[redacted]
[redacted]

AFTER

Crosses – data, Black lines – MC prediction

Difference:
- code cleaning / fixing technical bugs
- single tweak to geometry of the order ~1 um (was wrongly hard-coded in some parts of the code)

No change to the physics model, reconstruction algorithm, ...!

Workshop workplan

- Monday – discuss division of works after the talks
- Tuesday
  - we go together trough pieces of code, fix few bugs as an example
  - we do some simple documentation using Doxygen
- Wednesday
  - work
- Thursday
  - work
- Friday
  - Write reports and conclude