LBNE LArSoft Continuous Integration Goals, Status, Needs

Tom Junk, Brett Viren

Fermilab, BNL

LArSoft Continous Integration Workshop
June 17, 2014

Development Workflow

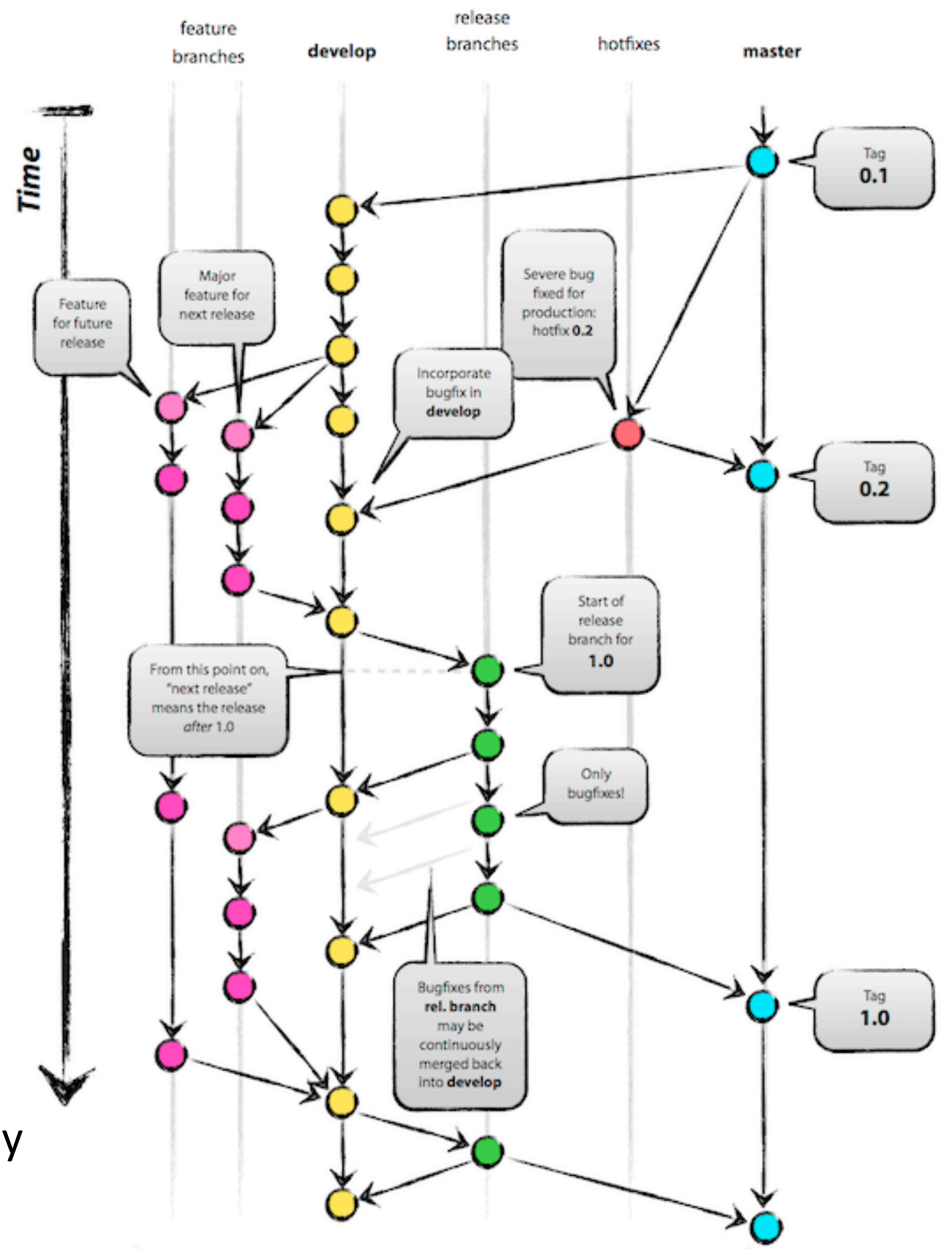
Git enables enforcing workflows including testing, review, integration.

→Better-quality software for large projects.

We like it!

This diagram is just the start – we live in a big software environment, with connections between LArSoft, lbnecode, and externals.

Must also be aware of compatibility with uboonecode and ArgoNeuT


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Author: Vincent Driessen
Original blog post: http://nvie.com/posts/a-succesful-git-branching-model
License: Creative Commons BY-SA

Features of LBNE's FD Simulation and Reconstruction Team

- Geographically Distributed
  - International!
  - Effort dominated by university students and postdocs.
  - Organizers more at labs, but some at universities

- Many Part-time contributions

- Need dedicated lbnecode librarian(s) and Release manager

- Seasonality of available effort – More in Summer, Less in Winter

- High turnover of students and postdocs – they need physics results to advance their careers.

- Very challenging simulation and reconstruction problem
  - Even more so than previous LArTPC's: Large detector, wrapped wires. Underground location helps hugely (low cosmic backgrounds)

Features of LBNE’s FD Simulation and Reconstruction Team

- Not all supervisors are adept at using our computing tools – students and postdocs usually are the ones who spend time with the tools.

  - Summer visits to Fermilab to gain expertise
  - Requests for workshops – Europe? Brazil? Other locations?

- Externally developed packages – integration challenge
  - NEST
  - PANDORA
  - for LBNE, even MicroBooNE code is “external”

- Would like to be able to develop code within the framework and also integrate non-framework code. Needs documentation and tests

LBNE Code and LArSoft Releases

- LArSoft is under active development and has frequent releases

- LBNE releases need to follow LArSoft releases
  - sign-off procedures – are all LArSoft Stakeholders consulted before a major new feature goes in?
  - Inadvertent breakage from new features that are not supposed to cause a problem but does.
  - Solutions already there:
    - Stable frozen releases
    - Active developers: Need more!
    - Librarians and Release Manager: identifying people

- Tests can help us identify foreseeable problems early, but do not provide fixes.

Features of the LBNE Detectors

Far Detector is Very Big – P5 wants 40 kt or more

- computational challenges (memory, CPU)

Many geometries!

- 35 t !
- 10 kt, 34 kt, larger.
- Surface or underground (drift length and total APA count changes)
- Wire angles: 45 degrees and 36 degrees (more? External constraints from APA frame limits (truck, shaft) and channel counts)
- 4-APA mini-FD for computational convenience
- ICARUS mock-up (can we analyze ICARUS data in LArSoft?)

Geometry is uncertain

- Need to attract international partners
- Partners can assist in design, funding, construction

Near Detector

- Not LArSoft, but art, and should have a software environment like the FD's.

Some Example Workflows that Can be Turned into Tests

- Geometry test : testgeo.csh produces output like this (for 4-APA FD geometry)

Basic test – the test should run and produce an output file.

```
fNchannels = 11024
For all identical APA:
Number of channels per APA = 2756
Number of WireIDs in a U plane = 1254
Number of WireIDs in a V plane = 1228
Number of WireIDs in a Z Plane = 559
U channels per APA = 828
V channels per APA = 810
Z channels per APA side = 559
Pitch in U Plane = 0.49
Pitch in V Plane = 0.5
Pitch in Z Plane = 0.45
```

Check these numbers against reference versions

More Output from Geometry Test

Just a random snip of an output file:

```
 Plane 2 has 559 wires and is at (x,y,z) = (-5.566,-351.25,126);
               pitch from plane 0 is 0.952;
              Orientation 1, View 2, Wire angle 1.5708
              TPC Dimensions: 228.967 x 702.5 x 253.5
              TPC Active Dimensions: 227.539 x 700 x 252
              TPC mass: 56193
              TPC drift distance: 227.539
              drift direction is towards positive x values
        testing PositionToTPC...
done.
              TPC 1 volTPCActive has 3 planes.
```

Can compare this against pre-stored values. Actual output file has datestamps in it, so cannot just diff the file with a reference, but we could imagine writing a tool that takes off datestamps.

What is this Testing?

- gdml files
- volume sorter
- geometry service access routines
- geometry test module code

- Some changes may be intentional.

- Geometry should be versioned, so an intentional change should result in a new test.

- Automatic test-maker tool? That's kind of what testgeo.csh already is, but CI needs a wrapper around it, and it has to be updated for every geometry we add.

- We care very much that our geometry is constant. But this is rather basic. And it's an easy guinea pig to put together.

Simulating an Event

- Basic workflow: Use the particle gun to generate a high-momentum muon in the 35t geometry.

My go-to test to see if things are working at all:

```
lar –c prodsingle_lbne35t.fcl
```

- Takes 11 secs to simulate a 6 GeV muon.
- Should be a very similar event every time, but we would prefer Gianluca's random number control to ensure we get the same event every time.

Generates two files:

single35t_gen.root -- art-formatted output file, simulated up to raw ADC digits
single35t_hist.root -- Noise, electron and photon distributions per step, step size

and of course stdout and sterr

Tests Possible with the Single Particle Simulation

Check logfile to make sure an event was simulated.
Look for nan's and inf's

Look for:
TrigReport Events total = 1 passed = 1 failed = 0

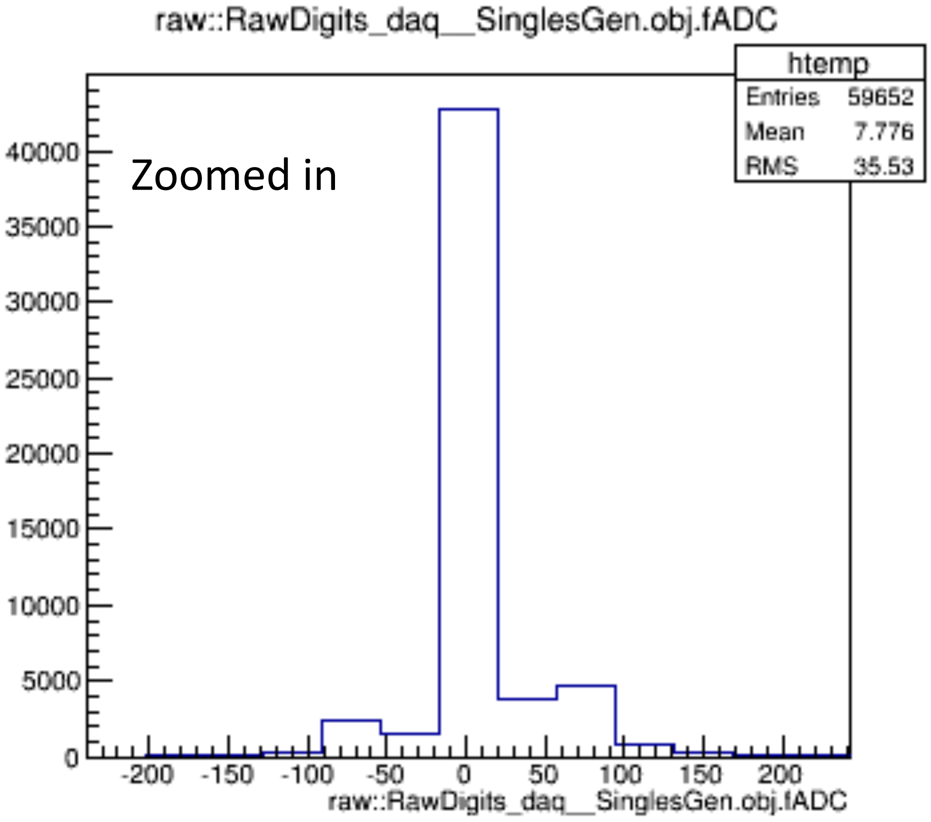
Checks with the art-formatted output rootfile:

- Check art output file to make sure exactly one event is in it

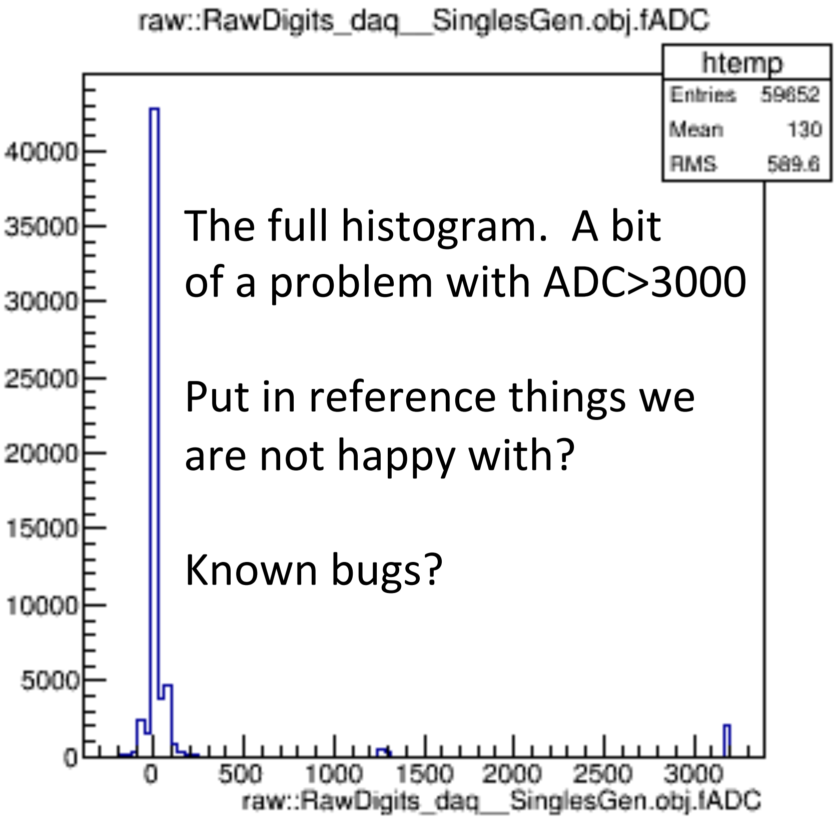
-- just look at the number of entries in the Events tree.

- Check that a muon is in the MC truth
- Check that raw digits are present
- Check distribution of raw ADC counts
- Check compression flag
- Check that metadata are as expected (will this evolve?)
- Check that photon detector data are present
- More ...

Tests Possible with the Single Particle Simulation



Zoomed in



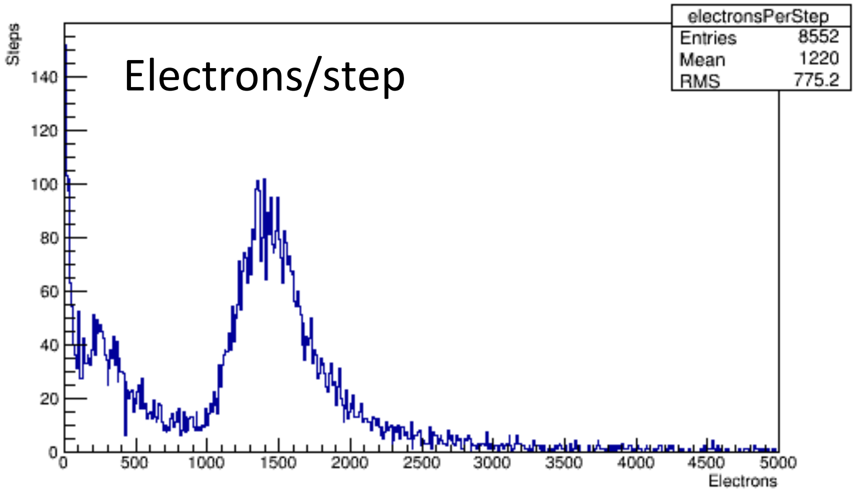
The full histogram. A bit of a problem with ADC>3000

Put in reference things we are not happy with?

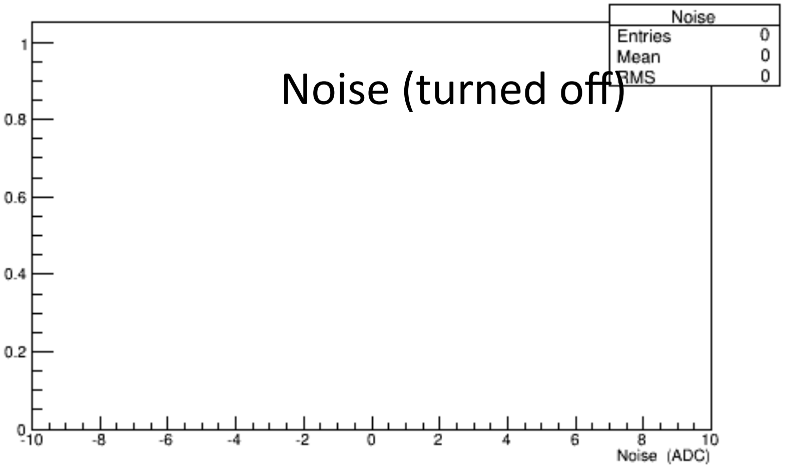
Known bugs?

May want to split this by plane.

The Contents of single35t_hist.root

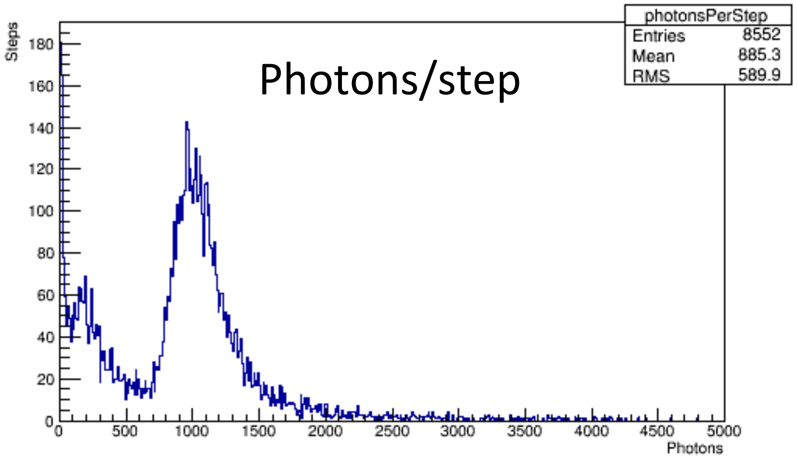


Electrons/step

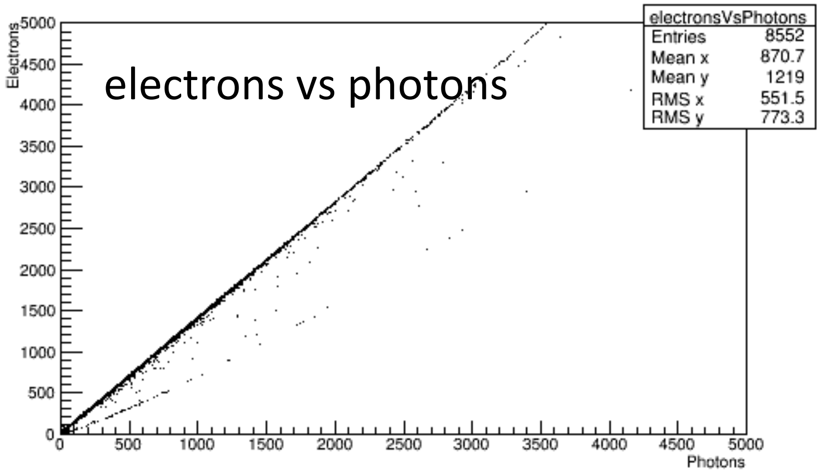


Noise (turned off)

Others: electrons per length
photons per length



Photons/step



electrons vs photons

The Next Step: Reconstruction

- Best to read in a pre-simulated test file – isolate the effects of simulation and reconstruction
- Run caldata and the hit finder
- Check logfile for error messages
- Look for hits in the output file – count recob::Hits
- Check CPU, memory usage, output file size against expectations

Unit Tests

- Geometry test isn't really a unit test – tests multiple things all at once
- A very good idea – the smaller the unit the better.
- Requires less judgment for deciding what success is – the coder can write these without convening a physics group meeting.
- Needs design thought – code may function with foreseen inputs, but fail in cases that are not tested.
- Unit tests solve (or at least identify) time-dependent problems – something that used to work fails due to a change in a dependency, can be caught with a unit test.
- Worry about orphaned code that breaks and no one wants to or even needs to fix it. Remove old code? Archive of unsupported code?

A Comment on Missing Features

LBNE has a lot of software it needs to write

Optimization of software is needed once it is written. The first attempt will not have asymptotic physics performance.

A user can write a test for someone else's code that tests

- features not yet implemented
- demands a level of performance that has not yet been achieved. (tracking efficiency. (energy resolution. PID....)

These aren't tests, so much as feature requests and performance goals. But maybe we want them!