

Mu2e-doc-4507-v1

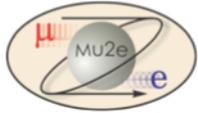


# What Am I Doing These Days

Rob Kutschke

October 27, 2011

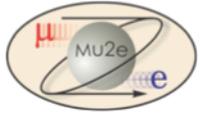
( SSE Weekly Meeting)



# Outline



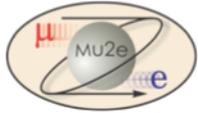
- Since this is the first talk in a series, I will talk about how we got here.



## Goals



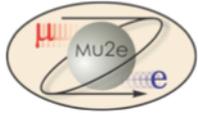
- Get Mu2e approved.



# Goals



- Get Mu2e approved.
  - My part is:
    - Acquire/develop software tools.
    - Support Mu2e people who use these tools.
    - Train Mu2e people to contribute to development.
    - Physics guidance of Mu2e people who use these tools.
    - If I am lucky, do some physics myself.
  - When it is consistent with the main goal:
    - Make choices that streamline the transition to operations.
    - Lobby other experiments to choose art.



# Software Tools

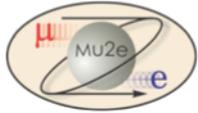


- Infrastructure Software

- Framework proper; event-data model; persistency; run-time configuration; 3<sup>rd</sup>-party software (Geant4, ROOT, CLHEP, boost, ... ), build tools, release and distribution management, file catalog, database support for geometry, conditions-data, metadata, integration with grid workflow management ...
- Should be written by software professionals and designed in consultation with the physicist end-users.

- Physics Software

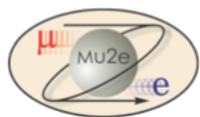
- The event-data objects and relationships among them.
- The geometry and conditions-data classes.
- Algorithms: hit creation, reconstruction, monitoring, analyses etc
- Will (mostly) be written by Mu2e people and designed in consultation with computing professionals.



# Mu2e Software



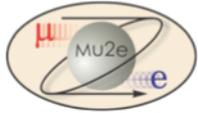
- Offline aka Mu2eSim (based on art)
  - Simulation is G4 based.
  - Root based event display.
  - Workhorse for detector design.
- G4Beamline
  - Workhorse for muon beamline design.
  - Why not Mu2eSim?
    - First out of the gate effect – 1 year lead time!
    - “C++ barrier”; people want to work with ntuples.
- MARS
  - Beat it against G4 for irradiation studies.
- **Geometries are independently maintained!**



# Why Did We Choose art?



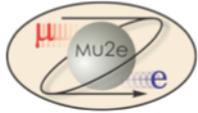
- CD agreed to support it.
- I liked many elements of the design:
  - Analysis-centric
  - Strong audit trail
  - Many utility features available on day one.
- Leverage ideas (and some code) that worked well in CMS.
- Avoid mistakes made by CMS.
- Hoped to persuade other experiments to adopt art
  - More people finding bugs and asking for features
  - Stronger case to have a strong CD team supporting it.
  - This has been realized.



# Use Cases for Simulation



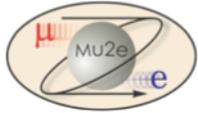
- A tool to:
  - Inform the design of the experiment
  - Once we have data: compute efficiencies, understand resolutions, backgrounds ...
  - ... and don't forget ...



# Use Cases for Simulations



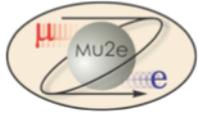
- A tool to:
  - Inform the design of the experiment
  - Once we have data: compute efficiencies, understand resolutions, backgrounds ...
  - A tool to develop, debug and characterize reconstruction algorithms.
    - Design this in from the beginning!



# CD People



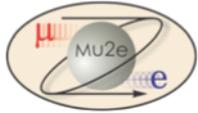
- Infrastructure software:
  - Marc, Jim, Chris, Walt, Lynn
  - Philippe in a consulting role ( low duty factor ).
  - Many people at 1 to 2 hours/month.
- Boundary between Infrastructure and Physics S/W
  - Krzysztof
  - **Would like more help here.**
- Physics S/W
  - Hans and Mark.
    - Developed some event-data objects and some algorithms that will be used in pattern recognition.



# Mu2e People



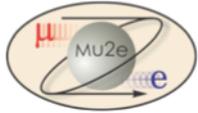
- Broad spectrum of expertise
  - Undergrads and new grad students
  - Until recently, very few post docs.
  - A few very experienced lab staff and research faculty.
    - Dave Brown, Vanya Logashenko, Andrei Gaponenko
- Documentation targeted at the middle ...
  - <http://mu2e.fnal.gov/atwork/computing/gettingstarted.shtml>
- ... but there is almost no middle
  - People have ~ my skill-set or are novices.
  - Most post docs grew up on mature experiments, so they never saw the sausage being made.



# Recent Team Accomplishments



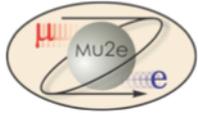
- Persisting pointers.
  - Goal: these should form a minimal, but complete, solution to the problem of “persisting pointers”.
  - Design effort included all of the CD team.
  - Implementation mostly Chris.
  - `art::Ptr<T>`, `art::Assns<A,B>`
- Event Mixing
  - Design effort involved the whole team
  - Implementation Chris (art side) RKK (mu2e side).
  - Enables study of backgrounds without generator level cuts.



# Recent Team Accomplishments



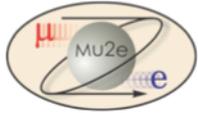
- Geometry:
  - Mostly work by Krzysztof
  - Add detailed representation of Cosmic Ray Veto system.
  - Decouple some pieces of the geometry.
  - Changed the EM Physics list attached to QGSP\_BERT
- Profiling work by Krzysztof.
- Modeling gradients in the magnetic fields
  - Mostly by Vanya Logashenko
  - Used for studies of trapped muons that can lead to electrons that can produce signal-like electrons.



# Recent Team Accomplishments



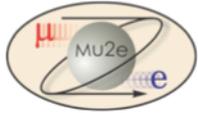
- Event generators
  - Work by Gianni Onorato
  - Have generators for all major classes of backgrounds.
  - Improved fidelity of all generators.
- Ported BaBar Kalman filter code
  - Dave Brown and RKK
  - Includes generic support for transient-only data products.
- Miscellaneous RKK
  - Added many enum-matched-to-string classes to remove magic numbers.
  - Ported Mu2e code to use `art::Ptr<T>`
  - Adapted hit making code to work with mixed events.



# Todo List



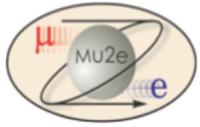
- Learn to use the grid submitting from GPCF.
  - Prep for demise of ilcsim\*
  - Use GlideIn rather than direct submission ( to get fair-share).
- Improve fidelity of tracker hit creation (Xiaobo Huang, Rice U.)
- Improve fidelity of calorimeter hit creation ( Caltech).
- First iteration of CRV hits (??)
- Speed up B-field calculation.
- Establish Base Release/Test Release model
  - Does this drive us to Cmake from scon?
- Improve unit testing and regression testing at Mu2e end.



# Todo List

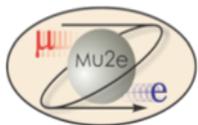


- Geometry
  - Synchronize Offline/G4Beamline/MARS.
  - Develop a tool to keep them sync'ed
  - Major refactor of the offline code to decouple elements.
  - Add extinction monitor.
- Double check G4 and thin volumes.
- Certify physics lists.
- Lots more ...



# Backup Slides





# The “C++ Barrier”



```
class ReadBack0 : public art::EDAnalyzer {
public:
    explicit ReadBack0(fhicl::ParameterSet const& pset){}
    virtual ~ReadBack0() {}

    void analyze(art::Event const& event);
};

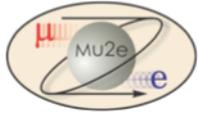
void ReadBack0::analyze(art::Event const& event) {

    art::Handle<SimParticleCollection> simsH;
    event.getByLabel("g4run",simsH);
    SimParticleCollection const& sims = *simsH;

    cout << "Number of simulated particles in event: " << event.id() << " " << sims.size() << endl;

    for ( SimParticleCollection::const_iterator j=sims.begin(), e=sims.end(); j != e; ++j){
        SimParticle const& sim(*j);
        cout << "PDG Id and momentum at creation: " << sim.pdgId() << " "
            << sim.startMomentum().vect().mag() << endl;
    }
}

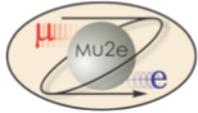
using mu2e::ReadBack0;
DEFINE_ART_MODULE(ReadBack0);
```



# “C++ Barrier”



- To discuss the example I need to introduce ideas from:
  - Basics of the C++ core language, STL, CLHEP.
  - Templates, inheritance, handles
  - Emphasize C++ references ( the & is invisible to novices)
  - What is a module? What is a framework? What is a fhicl?
  - What is a data product? What is a SimParticle?
  - How do I find out what information is inside a SimParticle?
- Many users say “C++ is too hard”:
  - Reality: you can’t even tell which manual you should look in!
  - It’s not that C++ is too hard, there is just information overload.
- Still to come: ROOT, TFileService, message logger, art::Ptr ...



# To Breach the C++ Barrier



- Need example driven, layered documentation
- Each layer should discuss the new ideas that are relevant for example at hand; then refer to the next layer for additional details.
- At the end of the chain you need the full documentation, both reference docs and conceptual ones.
- The problem is that you need to build it up from the bottom, which is tedious and boring.