



Mu2e-doc-5227

Analysis Model

Zhengyun You

University of California Irvine

Mu2e Computing Review

March 5-6, 2015



Outline



- The current status of Mu2e analysis model
 - Offline package release and build
 - Writing analysis modules
 - Documentation and tagging analysis jobs
- Analysis model development plan for the future
 - Base + Satellite release
 - Development and maintenance of analysis modules
 - Analysis Trees
 - Data persistency
- Summary



The Current Status



Offline Release and Build (1)



❑ Everyone builds their own Offline

- Check out the whole Offline package (source code, configurations, ...) from git / cvs repository
- Build the whole Offline package entirely using scons
 - Incremental build knows “only to build what is new or has changed”
 - scons scans the full code base to check that (~2 minutes)
 - There are short cuts available to scons experts

❑ No supports for partial build

- Build all packages or nothing
- Much less vulnerable to the problem that someone changes a header but does not recompile everything that must be recompiled
- More robust for beginners and non-software-experts



Offline Release and Build (2)



□ Why did we do it this way?

- Until recently the code base was small enough that this scaled well enough
- Time for one full build
 - On one of the mu2egpvm machines, an 8-way build takes about 30 minutes of real time
 - 120 minutes of CPU time (usr+sys)
- Disk space for one build
 - ~2.5 GB
- Until now, incremental builds are fast enough



Write Analysis Modules

□ How to write an analysis module

- Example modules provided for people to start from
- No concepts of "Production ntuples" or officially support "Analysis Trees"
- No systematic code review of analysis code

□ Where are the analysis modules

- Most analyzers create their own modules within the Mu2e Offline base code
- A few analyzers have separate git / cvs repositories that they check out on top of a build of Mu2e Offline
 - Literally in the same directory tree
 - This code is not part of Mu2e Offline but is still official Mu2e code
 - As above, incremental build of the full Mu2e Offline + their analysis code



Run Analysis Jobs



□ Documentation for analysis jobs

- The main production jobs are well documented
 - The tag/version of the code used in production/analysis
 - Configurations used in analysis
 - Statistics
- For other studies, trying to improve documentation via peer pressure

□ Run analysis jobs with a tag

- Run jobs with a local build is not a good habit
- Encourage people to commit and push, pull and rebuild code, tag it before doing the actual production run
 - Clearly know which version of code you are using for production jobs
 - Easier for others to understand and reproduce
 - When showing “important analysis plots”, software tag is required



The Future Plan



Analysis Module Release (1)



□ Base + Satellite release

– Base release

- A complete build of the Mu2e Offline that is published for all to use
- Publish these to CVMFS and perhaps to other places.

– Satellite release

- Any code that can be checked out separately from a base release and built against that base release
- May contain just a few analysis modules and their helper functions
- May contain a large analysis suite
- This differs from the current practice of checking out additional code on top of a private build of Mu2e Offline.

□ This combination will have much faster turn around

- Save build time
- Save disk space
- Easier for many satellite releases for a base release



Analysis Module Release (2)



- ❑ **A prototype system to support analysis with satellite release**
 - Still with the "write your own module" style
 - Simplify and speed up the build/release cycle.
 - Existing example analysis modules will continue to work
 - The prototype system to be released this summer.
- ❑ **Need to plan a repository structure**
 - Will analyses all be in one big git repository?
 - One or more git repositories per analysis group?
 - Continuously refactor helpers and utilities into a place to be used by all projects (base release / elsewhere), and make this systematic?
 - The repository structure for examples and introductory materials?
 - Where does test code belong?
 - Need to involve the collaboration in this discussion



Analysis Module Release (3)



Once the above is settled

- Many of the existing analysis modules can be moved out of the base release.
- Keep the ones that are part of formal testing and production
- Develop onboarding materials using the satellite release mechanism



Analysis Module Development



- **Engaging non-computing-experts for analyses**
 - We have some introductory examples to help non-computing-experts to start with
 - Read back, Track analysis, Calorimeter, Pattern Recognition
 - We will refactor the introductory examples into many small modules, not one giant one
 - Remove references to detector choices that we rejected
 - Write text to go with them
- **Development and Maintenance of the analysis modules**
 - The introductory examples will be running as part of the nightly build and verify that they work



Analysis Trees



□ Art based / Root based Analysis

- Art based
 - Have access for more detailed information
 - Require basic knowledge of art and Mu2e Offline framework
- Root based
 - Write out the event information you need into Root Trees for analysis
 - More simple and easier for analyzers

□ Analysis Trees

- The collaboration has asked that we support this
- Discuss with the collaboration about the right way to support this
- Some of the issues to discuss
 - Who has the responsibility to develop and maintain the modules?
 - Develop prototypes and train people in analysis group
 - Who has the responsibility of running the ntuples production jobs?
 - What kind of reviews of analysis codes are necessary?



Data persistency



- ❑ Read & Write data from/to dCache, SAM
- ❑ Analysis output
 - At present we do not persist tracks
 - To use tracks in analysis, need to rerun the tracking
 - Why no track persistency?
 - The track fit is done using code acquired from BaBar/Super-B
 - This code uses a very different persistency mechanism
 - Writing the code to persist state and restore a fully functional track is a lot of work
 - We have a rough design but it is not yet even a prototype
 - Do not want to create persistent tracks that become widely used but inconsistent with what the future persistent tracks will look like
 - At present, it is still a small problem but we need to fix it before it becomes a big one



Summary



- The current analysis model has served us well
 - We turned our simulation and reconstruction outputs into detailed sensitivity estimates with well understood systematic errors
 - We studied tradeoffs among design options using high level metrics, such as reconstructed tracks
 - But work was largely done by computing experts
- In the next phase of the experiment we need to enable contributions from physicists who are not computing experts.
 - Will work with the collaboration to choose a direction
- There are many details listed on the previous pages
 - Our goal is to complete the core of this program in about one year



Backup Slides



Backup