WhiteBoard

Mitigating the Woes of Developing and Releasing GlastRelease

- **Allow ample time for GlastRelease Releases**
  a GR tag does not imply a validated release
  - Tags for a new GR need to be submitted a week before the Release is expected to be use in production
    - Then a HEAD will be built and submitted to system tests (see below)
      - If there is a failure or odd result, the issue should be communicated up to P8 or L1 via JIRA and tracked (see below)
      - Repeat as necessary
- **Systems Tests**
  We need more people involved in running and interpreting system test results
  - A team of 3 or more will be assembled, with a rotating responsibility of running the systems tests.
    - People associated P8 and L1 will be involved in validating those results
  - Start running system tests on HEADs for P8
    Can't offer than up for L1, as we don't have a mechanism to create real HEAD builds in SCons along a branch as we did in CMT
    - First need to increase our disk allocation
- **Windows Support**
  - More Linux P8 developers plan to make use of glast-ts - begs the question - who is supporting them?
  - VC90 requires full Debug externals - while Optimized requires full Opt externals
    Certainly seems we only need Debug buildsOptimized builds should be discontinued, and all Windows externals will be provided as full debug with source and distributed via the RM
  - Issues associated with SCons/GoGui to improve its user friendliness.*** For example, the "clean" needs to really clean so we don't have to go find and delete by hand the "build" folders in each of the subpackages when we make an update to a package
    https://jira.slac.stanford.edu/browse/LSC-2
  - These and all other issues should be submitted to JIRA (see below)
- **SCons RM**
  Need a better way to deal with failures (especially windows ones)
  - Unit tests and zip files periodically fail to appear - rerunning those builds from scratch or waiting for the next build is not the best solution
    - Document the use of testBuild (run unit tests?) and createReleaseBuild (to recreate zip files on the fly?) or define some other script/exe that we can use as mere humans to populate missing bits in a build
  - Define a cron job to automatically reset the permissions on the windows zip files
    https://jira.slac.stanford.edu/browse/RM-35
- **TMine**
  - Define Unit tests
  - Reintroduce the direct use of TMine into GR, while also maintaining the TMineExt
    TMine Developers want access to TMine as a unit independent of GR
    However, we need to insure that the same version of ROOT and SConsFiles is used to build TMine in the context of GR
    GR can use TMine and its associated packages directly via the packageList.txt, and TMine is no longer an external library.
- **Clean Living**
  - JIRA is our friend
    - Random email threads are impossible to track
    - New GRs should be requested and tracks via JIRA
      - Request comes in with suggested updates for the new release (including tags if possible)
      - The request will be tracked in JIRA including updates associated with specific tag requests and system tests results
    - SCons Issues including GoGui:
      https://jira.slac.stanford.edu/plugins/servlet/project-config/LSC
    - ReleaseManager (SCons and CMT - as well as RMII web pages)
      https://jira.slac.stanford.edu/plugins/servlet/project-config/RM
    - Pass 8
      https://jira.slac.stanford.edu/browse/LPATE
    - Level 1
      https://jira.slac.stanford.edu/browse/LONE

Burning Issues

External Libraries

- We're falling behind in versions in most of our external libraries. Their sources are disappearing. They are no longer supported on newer platforms (RHEL4, Macs, etc..). Examples:
  - fftw (no longer has source available, doesn't compile on intel mac with gcc4)
  - gaudi (no longer has source available, requires minor source changes to work on rhel4)

Unit test Failures

- we have a handful of packages in GR and (I think) one in ST whose unit tests that routinely fail or enter infinite loops.
- perhaps there should be a review of what the tests are testing (if anything

"Pointer" Skims

- Eric Charles wrote a scheme to point to events in other files. This could be a good way to handle interleaved events and also take some of the pain out of opening N Root files in lock step. Could also come in handy for listing events to use for calibrations.

Dependencies and our use of CMT (Toby)

- Our dependencies are out of control: CMT has no way of evaluating whether the many includes for compilation, or explicit librarires for linking, are really needed. This needs a careful review of all our requirements files: are the use statements really needed? Can they be put in the private section? Should they have a "-no_auto_imports", which hides clients from unneeded dependencies? (This may then require "-imports" qualifiers.)
- Closely related to this is the fact that many such dependencies were generated by the needs of test programs. A very good example of this is RootIO, which exports a slew of dependencies to all its client packages, but has few dependencies to actually build itself.
- CMT provides a way to hide its complexities: patterns. One defines a few basic requirements files, and implements them with patterns. This was applied in the Science Tools, but we were always too busy to try to reorganize the Glast Release and Beam Test packages.

Environment variables (Toby)

Our use of CMT leaves a very heavy footprint for executables, which is a big strike against it. We are using it in a default mode in which it forces the definition of two variables for each package, namely <package>ROOT and <package>CONFIG. We never use the latter, but sometimes need the former. An example is the package Gleam: running the setup to define all the environment variables adds 255 variables!

This is easy to turn off, but any such assumption then needs an alternative environment variable, most often to find the root of the package. Having this explicit in the requirements file makes this public, a plus. This has been done for FluxSvc.

Test Programs (Toby)

We started a project to define test programs as sub-packages, allowing the dependencies to be factored. (A fix for much of the dependency problem mentionsed above.) This was done for FluxSvc, with a simple convention: if a package had a subpackage named "test_<package>", then that defined a test program for the package. Such a package does not show up with a CMT "show packages" command, but it is easy to check for the existence of, say, test_FluxSvc/cmt/requirements, then, when cd'ing to that folder, CMT recognizes the package. This is supported by MRvcmt, and apparently by the RM only on Windows. This ball was dropped, needs to be put back in play.

MRStudio (Joanne)

The design of MRStudio (gui layered on the core functionality) means that there can be much more commonality in the way RM does its builds and the way developers do, easing the burden of maintaining separate code to do essentially the same thing and making RM a more useful tool for debugging build problems on "the other" platform. This project deserves high priority.

Shared Libraries for Utility Packages (Joanne)

Currently facilities and detModel are built as static libraries rather than shared (it can't be a Gaudi-style component shareable of course, but it could be linked shared, similar to xmlBase and other packages). facilities is used by many (perhaps most) other packages. We are forgoing the benefits of a shareable (faster linking, less duplication of binaries, easier to propagate facilities bug fixes or improvements to all packages using it) for no reason that I am aware of. For detModel, used only by GlastSvc and the diagnostic package detCheck, the benefits are not as great, but the principle is the same. This is not a critical need, but it is an annoyance and there is some overhead involved in maintaining two branches of facilities (the main one building a static library for Gleam and ScienceTools; the other building a shareable for use in the Online environment).