Introduction

The Fermi Large Area Telescope (LAT) was launched as part of the Fermi Gamma-ray Space Telescope on June 11th 2008. The LAT collaboration’s offline software includes:

GlastRelease: C++ Monte Carlo simulation and data reconstruction software utilized as part of the offline data processing ScienceTools: all software related to scientific analysis of Fermi LAT data written in C++ with python interfaces

We are a relatively small collaboration with a maximum of 25 developers in our heyday. Our intent from the moment Fermi’s proposal was accepted was to provide one code system for simulation, test data analysis, and flight operations. During our software development, we leveraged a number of external libraries which include ROOT, Gaudi, Geant4, CFITSIO, Swig, Python, and Xerces. This helped to alleviate the task of man power we had available to get a system running quickly. We support a community interest in running our software on their personal laptops. This drove us to provide binary distributions and a few GUIs to aid source code compilation.

With six years of us, are we in the phase where we must move forward to support modern operating systems and components to get us through the life of the mission. This means upgrading our external libraries as well. We are a small developer force, and it is crucial to our production system that we carefully orchestrate all upgrades to insure stability.

Issues And Lessons Learned

• Using External libraries avoids re-inventing the wheel
  • C++ REAPER, Gaudi, Geant4, Wgl were all examples of code we were much better off taking and using. All supported both Windows and Linux (some also now supported Mac). We could then focus our efforts on tasks specific to Fermi.
  • Support for some OSes are better than others
  • Windows can be a bit of a problem. A majority of the externals we use now support CMake (G4, ROOT, Geant4), making this less of an issue.
  • Use External sparingly
    • External libraries can offer a treasure trove of features and free code, it does come with a cost. This is code you do not control. Your ability to later upgrade these systems or components may be impacted by the externals you choose today.
  • Pay attention to dependencies
    • Some externals depend on other libraries. You may find that there are conflicting versions required by various externals. At best, you may be forced to upgrade a number of others due to these dependencies.
  • Don’t wait too long to upgrade
    • When possible, it is much better and easier to handle incremental upgrades rather than jumping several versions at once.
  • Make Friends
    • When you do utilize an external library, find the experts associated with that external someday, and you need good resources to contact.
  • Never make use of non-standard features
    • Interfaces change, and certainly over the long haul of a mission, you are taking advantage of some quirk in the code of an external library, the bug will be pulled out from under you.
  • Be wary of your own free code
    • Our codebase is open source and available to anyone as long as you are taking advantage of some quirk in the code of an external library.

Stability versus Development

Our data processing pipeline has been utilizing a relatively stable version of GlastRelease since launch. Some external upgrades, patches and bug fixes have been allowed. We use CVS as our code repository and branching to implement required code changes to our stable releases.

Problem: There is little confidence in the use of CVS branches across our development team. Fix: We have or two developers will willing to the job of managing branch for GlastRelease. For ScienceTools, branches are avoided altogether in favor of rapidly applying patches along the main trunk and rolling out new tagged releases.

The project was on a later timeline, we likely would have moved to Subversion (SVN).

Problem: Stability is often favored over introducing “unnecessary” patches, which can result in improvements being passed over for years at a time.

Fix: The development branch should keep up with external library versions more diligently, e.g. jumping multiple versions when we do upgrade is very painful.

User Support

Our Online User Workbook, largely written by a dedicated technical writer, has been a vital component in supporting our distributed team of users and developers across the LAT collaboration.

Unfortunately, we no longer fund a technical writer, but the development team is working to keep the content up-to-date.