Fermi Large Area Telescope Offline Software Maintenance Madness

Heather Kelly (SLAC)

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:

Gaudi: C++ Monte Carlo simulation and data reconstruction software utilized as part of the offline data processing pipeline

ScienceTools: all software related to scientific analysis of Fermi LAT data written in C++ with python interfaces

During our software development, we leveraged a number of external libraries which include ROOT, Gaudi, Geant4, CFITSIO, Swig, Python, and others. ROOT wasRoot Linux 3.23b and Windows V52003. With eight years ahead of us, we are in the phase of our project where we must move forward to support modern operating systems and compilers to get us through the life of the mission. This means upgrading our external libraries as well. It is crucial to our production system that we carefully orchestrate all upgrades to insure stability. This poster will focus on our experiences with two of our nineteen external libraries, maintenance of a large scale offline software project, and support of our development and user communities.

Issues And Lessons Learned

Use Externals sparingly – While 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 operating systems or compilers 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, upgrading one library, may force you 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 a particular external and get to know them. You will have questions and problems 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, if you are taking advantage of some quirk in the code of an external library, the bug will be pulled out from under you.

Features of your own free code – Our choice to adopt an event display built upon Fox and Ruby has proved to be a maintenance issue due to the loss of both developers associated with that project. We are now in a situation where we are moving to another event display which does not yet provide all the features of our old one, while the old one lacks any support. Those with Fox or Ruby experience are few and far between.

Stability versus Development
Our data processing pipeline has been utilizing a relatively stable version of Gaudi 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 one or two developers willing to tackle the job of supporting our distributed team of users and developers across the LAT collaboration.

We have weekly offline software meetings, as well as dedicated meetings for special projects.

Mailing lists and instant messaging also provide communication opportunities.

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

Why Windows? (and cygwin just won’t do)
We have a handful of proficient Windows developers attached to the Visual Studio development environment: Integrated Debugger – go from error messages to setting breakpoints in a couple of clicks.

Stability and portability – automated class member completion.

Integration Editor and Build Properties – allows programmers to set compile and link settings quickly and easily.

Supported Operating Systems
Redhat Enterprise Linux 4 – gcc 3.4.3 32 and 64 bit systems
Redhat Enterprise Linux 5 – gcc 4.2 and 64 bit systems
Windows XP (Server 2003) – Visual Studio 2003 compilers
Windows 7 (Server 2008) – Visual Studio 2008 compilers

ScienceTools Only
Mac OS X versions 10.4 (Tiger) and 10.5 (Snow Leopard)

Why Windows? (and cygwin just won’t do)
We have a handful of proficient Windows developers attached to the Visual Studio development environment: Integrated Debugger – go from error messages to setting breakpoints in a couple of clicks.

Stability and portability – automated class member completion.

Integration Editor and Build Properties – allows programmers to set compile and link settings quickly and easily.

Supported Operating Systems
Redhat Enterprise Linux 4 – gcc 3.4.3 32 and 64 bit systems
Redhat Enterprise Linux 5 – gcc 4.2 and 64 bit systems
Windows XP (Server 2003) – Visual Studio 2003 compilers
Windows 7 (Server 2008) – Visual Studio 2008 compilers

ScienceTools Only
Mac OS X versions 10.4 (Tiger) and 10.5 (Snow Leopard)

Why Windows? (and cygwin just won’t do)
We have a handful of proficient Windows developers attached to the Visual Studio development environment: Integrated Debugger – go from error messages to setting breakpoints in a couple of clicks.

Stability and portability – automated class member completion.

Integration Editor and Build Properties – allows programmers to set compile and link settings quickly and easily.

Supported Operating Systems
Redhat Enterprise Linux 4 – gcc 3.4.3 32 and 64 bit systems
Redhat Enterprise Linux 5 – gcc 4.2 and 64 bit systems
Windows XP (Server 2003) – Visual Studio 2003 compilers
Windows 7 (Server 2008) – Visual Studio 2008 compilers

ScienceTools Only
Mac OS X versions 10.4 (Tiger) and 10.5 (Snow Leopard)

Why Windows? (and cygwin just won’t do)
We have a handful of proficient Windows developers attached to the Visual Studio development environment: Integrated Debugger – go from error messages to setting breakpoints in a couple of clicks.

Stability and portability – automated class member completion.

Integration Editor and Build Properties – allows programmers to set compile and link settings quickly and easily.

Supported Operating Systems
Redhat Enterprise Linux 4 – gcc 3.4.3 32 and 64 bit systems
Redhat Enterprise Linux 5 – gcc 4.2 and 64 bit systems
Windows XP (Server 2003) – Visual Studio 2003 compilers
Windows 7 (Server 2008) – Visual Studio 2008 compilers

ScienceTools Only
Mac OS X versions 10.4 (Tiger) and 10.5 (Snow Leopard)

Why Windows? (and cygwin just won’t do)
We have a handful of proficient Windows developers attached to the Visual Studio development environment: Integrated Debugger – go from error messages to setting breakpoints in a couple of clicks.

Stability and portability – automated class member completion.

Integration Editor and Build Properties – allows programmers to set compile and link settings quickly and easily.