SAS software: Code Development Infrastructure

Technology choices for:

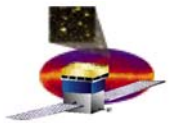
- Language
- Platforms
- Code versioning
- Execution framework
- Code documentation
- I/O

Documentation Task Force

T. Burnett, H. Kelly

Our Products: much more than code!

- Support infrastructure, must support a variety of clients:
  - developers
  - sophisticated users
  - end users
- Elements:
  - Supported platforms & compilers
  - Development environments
  - Coding and documentation standards
  - Build tools
  - Framework
  - Analysis tools



Basic principles for technology choices

- Don't invent anything unnecessarily
- Borrow from existing solutions, experience

→ High energy physics

- very similar parameters: detectors, analysis requirements, data, users
- Pioneer was here at SLAC: the Babar experiment in mid 90's,
  - Broke with Fortran-oriented past: unix, OO C++
  - Adopted industry-standard *CVS* for version management
  - Invented package-oriented build system *SRT*
  - Developed an OO *framework* for managing processing steps
  - Successfully trained physicists to deal with new environment

Technology choices: language

- Object-oriented C++
  - Basic value of encapsulation of data now well-established
  - Build on success of Babar and all other new HEP experiments: Belle, D0, CDF, ATLAS, CMS, LHCb
  - Now a standard, most compilers approach this
  - Standard Template Library provides rich menu of algorithms and object containers.
  - Required to use a C++ specific framework

Technology choices: platforms

- Windows PC
  - Our preferred development environment due to rapid development made possible by Microsoft Visual C++ MSDEV

- linux
  - The preferred choice for European developers
  - Required for SLAC batch support

- solaris
  - not supported now, but in reserve if needed for SLAC batch.

Technology choices: code versioning

- CVS!
  - Concurrent Versions System, the dominant open-source network-transparent version control system.
  - Useful for everyone from individual developers to large, distributed teams:
    - Client-server access method lets developers access the latest code from anywhere there's an Internet connection.
    - Unreserved check-out model to version control avoids artificial conflicts common with the exclusive check-out model.
    - Client tools are available on all our platforms.
  - Web-based repository browser available (cvsweb)

Technology choices: code management

Legacy of Babar's SRT: building apps from *packages*

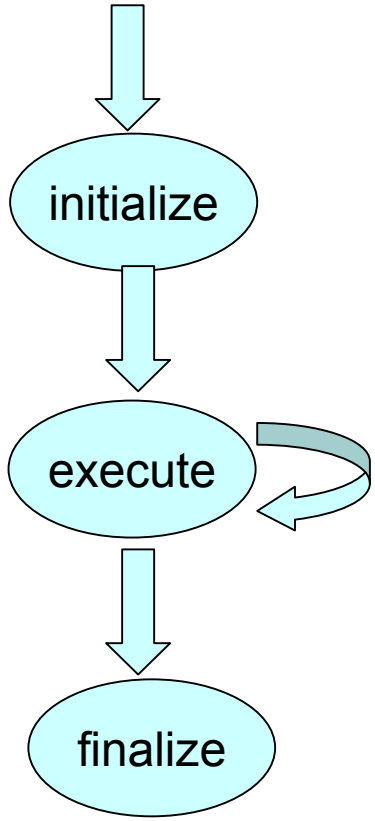
- Package: collection of source files, with public header files in a folder (usually) with the package name
- Produces a binary library and/or executable

CMT (for Code Management Tool): our choice

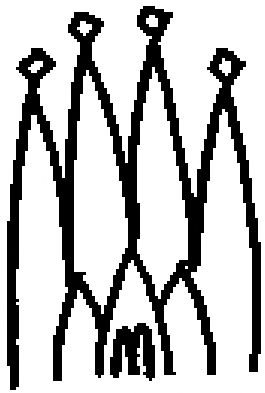
- Developed at Orsay in response to deficiencies of SRT, adopted by LHCb and ATLAS
- Supports Windows
- Clean model for package dependencies
  - Support for compile-time, link-time, and execution-time
- Configuration specified in a single file
- Includes tool to generate makefiles, or MSDEV files
- Uses CVS tags to correspond to versions

framework requirements

- Support event-oriented processing, three phases
  - initialization
  - event-loop generating or processing events
  - termination
- Define flexible way to specify processing modules to be called in the execute loop, without need to recompile/relink
- Provide *services*, especially for making n-tuples and histograms

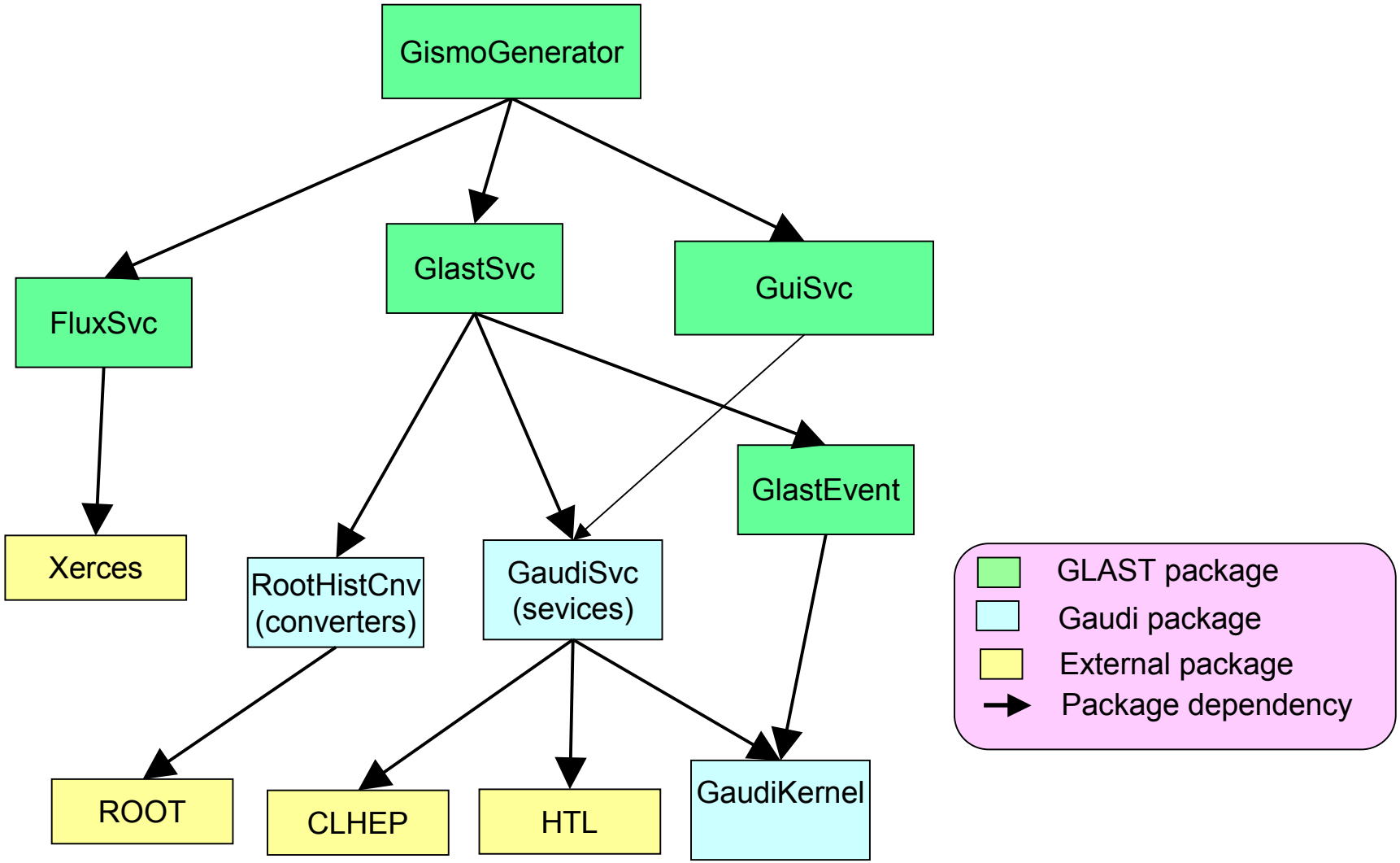


Gaudi: our framework choice



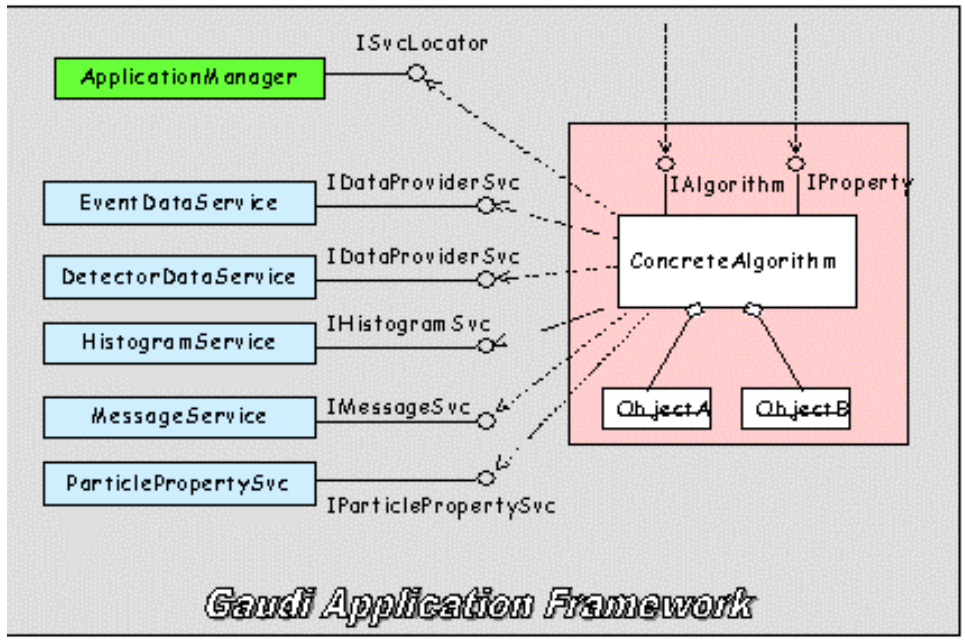
- Open source
- Stable, but active developers, in use by ATLAS, LHCb
- Very good documentation
- All code called via component interfaces:
  - Algorithm
  - Service
  - Converter
  - DataObject
- Support for shareables: all code is loaded dynamically
- Job control parameters set in job options file.

Glast/Gaudi Example Architecture

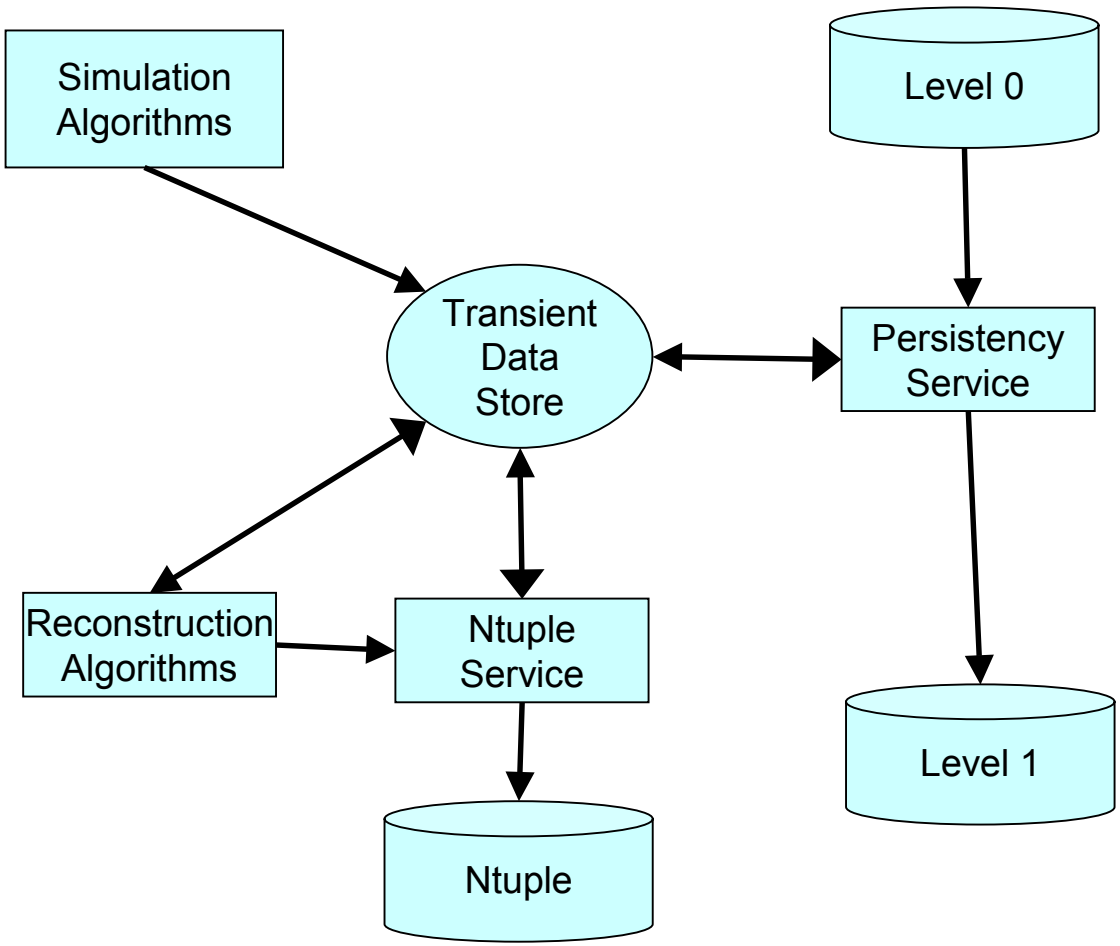


Gaudi Algorithm as a *component*

- Components are similar to those of Corba or COM: implement an abstract interface.
- Easy to substitute components: actual concrete implementation to be used is determined at run-time from a simple ascii file.
- Example diagram: A ConcreteAlgorithm:
  - Implements 2 interfaces
  - requests services from 6 services via abstract interfaces



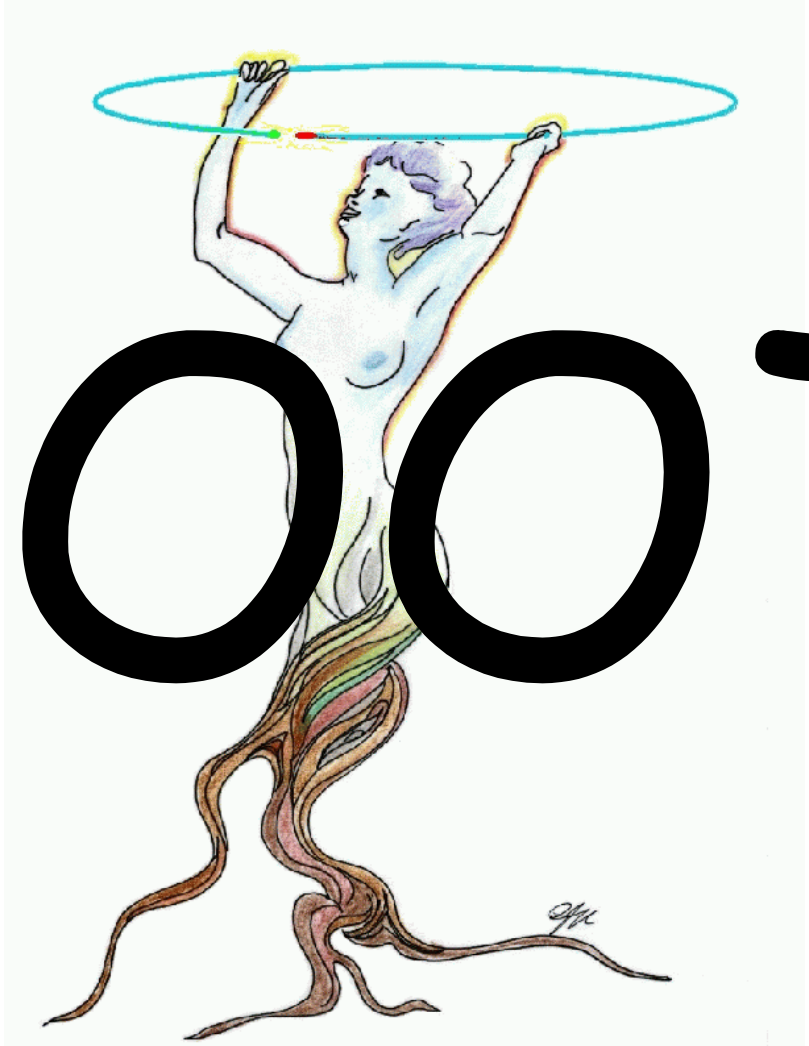
Data flow in the Gaudi framework

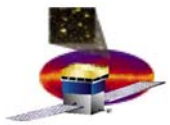


Choice: code documentation

- Doxygen!
  - http://www.doxygen.org
  - Generates on-line HTML pages
  - Generates off-line reference manuals in a variety of formats
    - Including hyperlinked PDF
  - Available on our supported platforms
- Guidelines for standard Doxygen usage under review.
  - Standard Doxygen configuration file
  - Code templates including Doxygen comments
- Documentation Task Force
  - See later

Choice: I/O format (and Event Analysis)



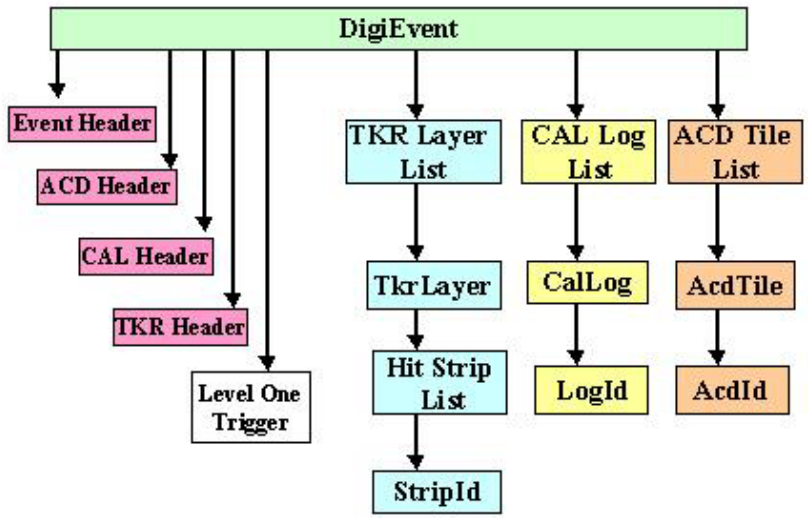


Features of ROOT I/O

- Machine independence
  - ROOT is freely available on all of our supported platforms.
- Self-describing
  - Files created today will be readable years from now.
- Support for Object I/O
  - The detailed structure of our data is preserved for analysis.
- Schema evolution
  - Changes in our internal data structures will be tracked.
- On the fly compression
  - ROOT uses an algorithm based on gzip.
- Widespread use in the HEP community.
  - CDF at FNAL; several experiments at RHIC

Object I/O

- Detailed tree structure of data is preserved.
- Described by C++ classes
- Branched I/O
  - Reduces unnecessary I/O by reading in only desired branches.
- Summary data is available in ROOT Ntuples.



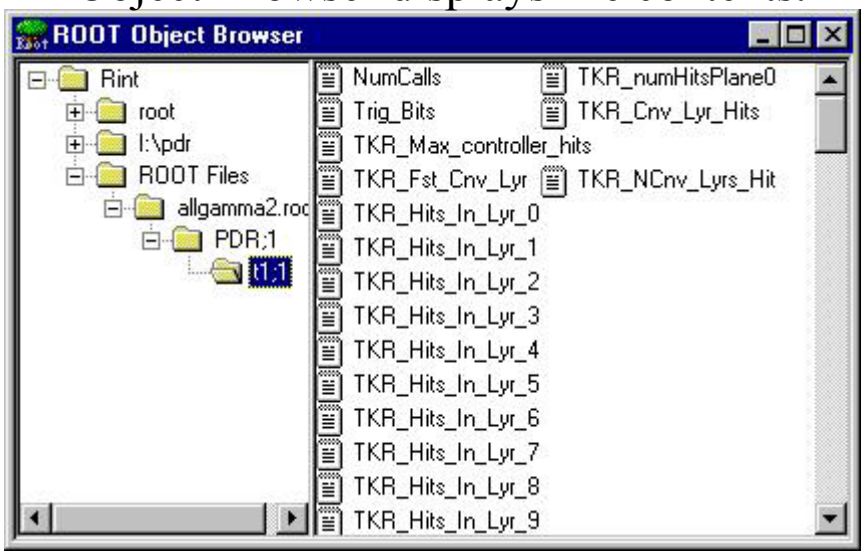
Logical structure for the raw digitization data

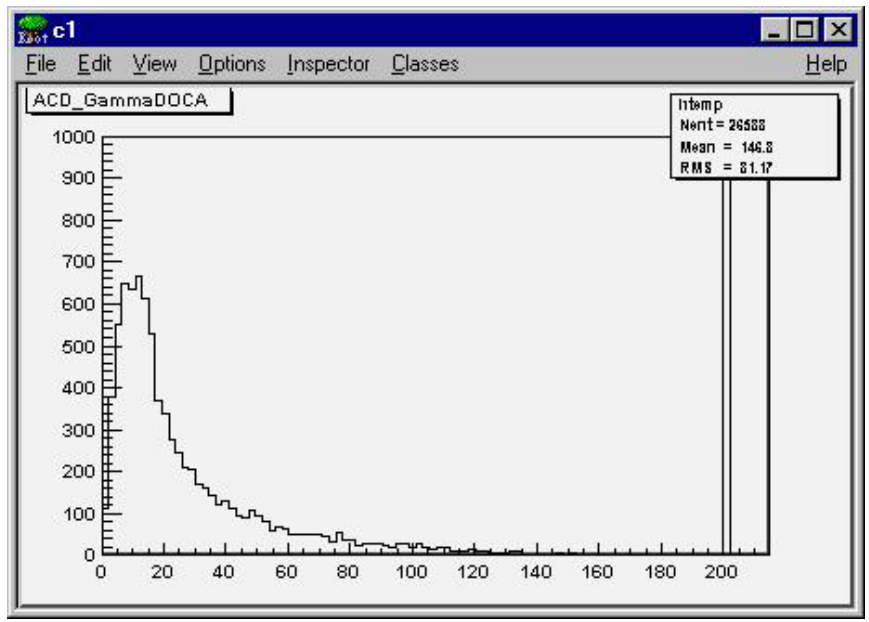
Internal structure for storage of detector data

ROOT for Event Analysis

- Supports both interactive and batch processing.
- Free and available on all supported platforms.
- Strong and growing user base.
- Histogramming, function fitting, and GUI widgets.

Object Browser displays file contents.





Histograms produced at the click of a button.

Documentation, user support

- Gaudi, CMT, CVS: user guides available
- Local guides (web-based)

Software	[Getting Started with GLAST Software (Your How-To Page)] [Web Access to CVS repository] [Using GlastSim] [Using tbsim] [Using ROOTWriter] [Using tb_recon]
Support	[Whom to Call??] [Facilities at SLAC] [UW Windows Server]
Projects	[GAUDI] [GEANT4] [PDR] [Software PDR] [Event Display]
Other Software Resources	[Italy] [UCSC TB Recon] [Goddard] [NRL Software] [Hiroshima]
Tools	[Telecon VRVS] [Using VRVS for Glast] [Instant Msg ICQ] [Using ICQ for Glast] [CVS] [Using Cvs for Glast] [CMT] [Using CMT for Glast] [Root] [Using Root for Glast] [Root at FNAL] [Creating PEGS files]

Documentation Task Force

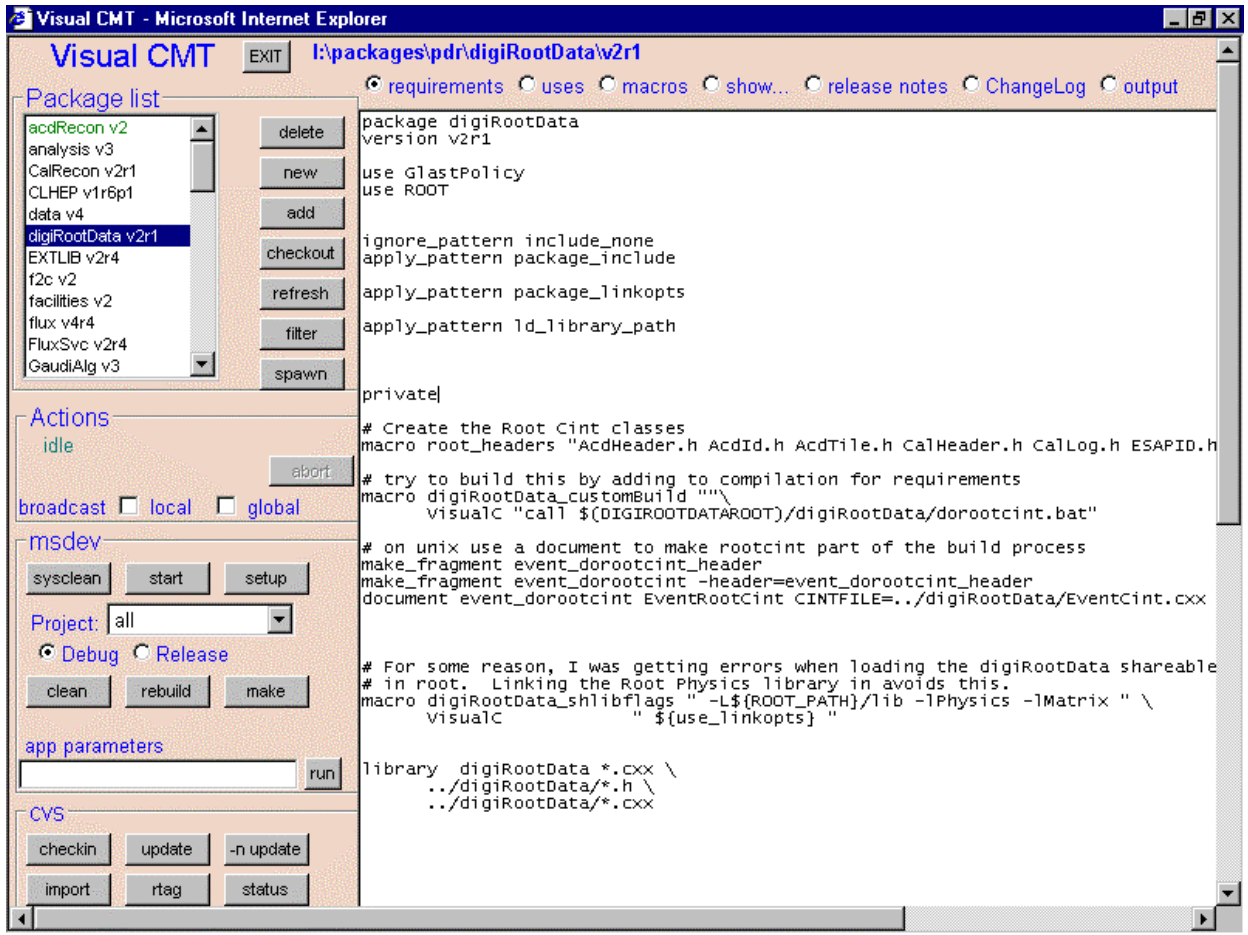
- Plan and Implement Documentation for GLAST SAS.
  - Web Site
    - http://www-glast.slac.stanford.edu/software/core/documentation/
  - Charge
    1. Ensure that all GLAST SAS policies and procedures are accurately documented.
    2. Provide and maintain standard templates for code and web pages.
    3. Assess the current documentation both for developers and users.
       a. Audit all GLAST SAS web pages to insure consistency and readability.
       b. Audit existing Doxygen generated developer documentation to highlight areas of improvement and guide creation of Doxygen guidelines.
       c. Reorganize existing documentation, in conjunction with documentation owners.
    4. Maintain Doxygen guidelines for GLAST SAS, including examples.
    5. Maintain automated generation of Doxygen pages for code packages.
    6. Develop plans for generating and maintaining a GLAST SAS Developer Guide.
    7. Develop plans for generating and maintaining a GLAST SAS User Guide.
    8. Arrange and participate in developer and user documentation walk-throughs.
    9. Develop and provide online tutorials for all GLAST SAS utilities and products.
    10. Develop a plan for regular maintenance of the documentation.

Documentation Task Force contd.

- Schedule
  - Item (2) Standard templates will be completed the end of December 2001. ✓
  - Item (3a) The audit of existing web pages should be completed by mid January 2002. (3c) The web pages will be reorganized by mid February 2002.
  - Item (3b) The audit of existing Doxygen pages will be completed by mid-January 2002. Suggestions will be provided to package owners by mid-February 2002.
  - Item (4) Guidelines and documentation for Doxygen usage will be completed by early January 2002.
  - Item (5) The automated generation of Doxygen pages is ongoing, and the initial scripts are in place.
  - Item (6) The outline and design report of the first GLAST SAS Developer's Guide will be created by March 2002. A first draft of the Developer's Guide will be available for review by July 2002.
  - Item (7) The outline and design report of the first GLAST SAS User's Guide will be created by August 2002. A first draft of the User's Guide will be available for review by October 2002.
  - Item (8) The methods and procedures for conducting documentation walk-throughs will be drawn up by the end of February 2002. The first of these walk-throughs will be scheduled for late March 2002, as the new version of the reconstruction algorithms become available.
  - Item (9) A design report and list of required tutorials will be generated by March 2002. The first online tutorial will be made available July 2002.

Help in the form of a GUI

- GUI interface to:
  - CMT: manage packages
  - CVS: check out, commit
  - MSDEV: build, or start its GUI
  - executable: run with command-line parameters
  - doxygen: run, examine results
- Only Windows. Hope to extend to unix.



The Coding Process

- Inline documentation: doxygen
  - Each package must have a *mainpage.h* to introduce the purpose, provide direct links to top-level classes (Doc task force on top of this.)
- Coding rules
  - Avoid potentially bad constructions
  - Maintain some uniformity
  - Standard templates for appearance
- Testing
  - Each package defines test programs
- Reviews
  - Periodic reviews of code for design, adherence to reviews

package flux
v4r5

This package contains all code to generate particles for GLAST simulation. The primary interface is via a FluxMgr object.
A list of possible sources, with details on implementation, is in the file xml/source_libarary.xml
All calculation of spectra is done in Spectrum objects,

Managed setups for developers

University of Washington Terminal Server

- Uses Windows 2000 Terminal Server: free clients available for any Windows operating system
- Complete environment available for users, including VCMT, cvs, ssh, msdev, doxygen

SLAC unix

- Standard group .cshrc
- releases automatically available

Both: plan to implement automatic build facilities for

- overnight builds of HEAD versions
- on demand builds of specified packages