GLAST LAT TECHNICAL REPORT	Document # **LAT-TD-00322-04**	Date Effective November 13, 2001
	Author(s) R. Dubois	Supersedes None
	Subsystem/Office Science Analysis Software	
Document Title **LAT Science Analysis Software Subsystem Preliminary Design Report**		

DRAFT 4

Gamma-ray Large Area Space Telescope (GLAST)
Large Area Telescope (LAT)
Science Analysis Software (SAS) Subsystem Preliminary Design Report

CHANGE HISTORY LOG

Revision	Effective Date	Description of Changes
2	11 Aug 2001	Many updates from team members' comments
4	13 Nov 2001	Update milestones and timeline to match PMCS

Table of Contents

List of Figures

List of Tables

1 Purpose

This document presents the status of the GLAST LAT Science Analysis Software subsystem design and planning in support of the August 17, 2001 Peer Design Review.

2 Acronyms and Definitions

ACD	The LAT Anti-Coincidence Detector Subsystem
AGN	Active Galactic Nuclei
AO	GLAST Announcement of Oppportunity
BFEM	Balloon Flight Engineering Model
BTEM	Beam Test Engineering Model, 1999-2000 beamtest run
CAL	The LAT Calorimeter Subsystem
CMT	Code Management Tool
COTS	Commercial, off-the-shelf
CVS	Concurrent Versioning System, Code Versioning System
DPF	Data Processing Facility
Gaudi	a C++ code framework
GEANT4 (G4)	a C++ simulation and particle transport package
GISMO	C++ simulation and particle transport package
GLAST	Gamma-ray Large Area Space Telescope
GRB	Gamma Ray Burst
GUI	Graphical User Interface, as in Windows or X/Motif.
HEP	High Energy Physics
ICQ	a free instant messenger tool (“I seek you”), owned by AOL.
IOC	Instrument Operations Center
KF	Kalman Filter, an algorithm for fitting trajectories in the TKR
LAT	Large Area Telescope
MC	Monte Carlo
MIP	minimum ionizing particle
MOC	Mission Operations Center
MySQL	free SQL-based relational database
ntuple	A tabular data structure, sometimes called an n-tuple.
PDS	Gaudi Persistent Data Store
PI	Principal Investigator
PSF	Point Spread Function, a measure of the angular resolution

ROOT	a C++ analysis framework, developed at CERN.
SAS	Science Analysis Software
SDP	Science Data Processing center
SQL	Structured Query Language, used to access databases
SSC	Science Support Center
TBD	To Be Determined
TBR	To Be Resolved
TDS	Gaudi Transient Data Store
TKR	The LAT Tracker subsystem
TOT	Time over threshold, a measure of the TKR charge
VRVS	Virtual Room video-conferencing system, web-tool
WBS	work breakdown structure

3 Applicable Documents

3.1 Software WWW Home Page

- http://www-glast.slac.stanford.edu/software/

3.2 Requirements

- Level 3 specification - LAT-SS-00020-00
- Level 4 specification - http://www-glast.slac.stanford.edu/software/Requirements/

3.3 Data Format Definitions

- Proposed MC Hits concept - http://www-glast.slac.stanford.edu/Software/mc_hits_proposal_rev1.htm
- Proposed Digitizations definitions
    - ACD
        - reqs - http://www-glast.slac.stanford.edu/software/PDR/SAS/acd_digis_req.htm
        - def'n - http://www-glast.slac.stanford.edu/software/PDR/SAS/acd_digitization.htm
    - TKR - http://www-glast.slac.stanford.edu/software/PDR/SAS/TkrDigis.htm
    - CAL - http://gamma.nrl.navy.mil/glast/Calorimeter_software_reqs/CalorimeterDigiRequirements.htm
- Prototype Reconstruction definitions
    - text - http://www-glast.slac.stanford.edu/software/balloon/root/Root-Class-Definitions.htm#RECON
    - diagram - http://www-glast.slac.stanford.edu/software/balloon/root/ReconRootClasses.pdf

3.4 Flight Geometry Specification Documents

- http://www-glast.slac.stanford.edu/software/detector_description/

3.5 Subsystem Reviews

- CAL - http://www-glast.slac.stanford.edu/software/Reviews/CAL/
- TKR - http://www-glast.slac.stanford.edu/software/Reviews/TKR/
- Core - http://www-glast.slac.stanford.edu/software/reviews/core/

3.6 External Packages

- Gaudi - http://proj-gaudi.web.cern.ch/proj-gaudi/
- ROOT - http://root.cern.ch

- CVS - http://www.cvshome.org
- CMT - http://www.lal.in2p3.fr/SI/CMT/CMT.htm
- Xerces - http://xml.apache.org/xerces-c/index.html
- xml - http://www.xml.org
- Doxygen - http://www.doxygen.org/
- MySql - http://www.mysql.com/information/

3.7 User Documentation

- Software introduction: http://www-glast.slac.stanford.edu/software/Overview/
- Software "How-To" Guide: http://www-glast.slac.stanford.edu/software/CodeHowTo/

3.8 Workshops

- General - Jan 2000 - http://www-glast.slac.stanford.edu/software/Workshops/dec99workshop/info/
- General - May 2000 - http://www-glast.slac.stanford.edu/software/Workshops/May00Workshop/Planning/
- Core - Aug 2000 - http://www-glast.slac.stanford.edu/software/Workshops/GaudiG4Aug00/
- General - Sept 2000 - http://www-glast.slac.stanford.edu/software/Workshops/September00Workshop/
- Core - May 2001 - http://www-glast.slac.stanford.edu/software/Workshops/CoreWS042001/

3.9 General GLAST References

- Response to AO 99-OSS-03. “GLAST Large Area Telescope, Flight Investigation: An Astro-Particle Physics Partnership Exploring the High-Energy Universe.” Volume 1: Scientific andTechnical Plan. Foldouts: A, B, C, D.
- GSFC 433-SRD-0001, “GLAST Science Requirements Document”, P.Michelson and N.Gehrels, eds., July 9, 1999
- LAT-SS-00010, “LAT Instrument Performance Specification.”
- GSFC 433-SPEC-001, “GLAST Project Mission System Specification,” April 24, 2001
- GSFC 433-IRD-0001, “GLAST Science Instrument – Spacecraft Interface Requirements Document”, Draft July 14, 2000
- GSFC 433-MAR-0001, “Mission Assurance Requirements (MAR) for Gamma-Ray Large Area Telescope (GLAST) Large Area Telescope (LAT)”, June 9, 2000
- GSFC 433-RQMT-0005, “GLAST EMI Requirements Document.”
- GSFC 433-OPS-0001, “GLAST Operations Concept”, Sept 7, 2000
- LAT-SS-00047, “LAT Mechanical Performance Specification.”
- LAT-MD-00099, “LAT EEE Parts Program Control Plan,” March 2001
- LAT-MD-00039, LAT Performance Assurance Implementation Plan
- LAT-MD-00033, “LAT Work Breakdown Structure,” May 9, 2001
- LAT-TD-00125, “LAT Mass and Power Allocation Recommendations”

- “Gamma Ray Large Area Space Telescope Instrument Technology Development Program”, NRA 98-217-02, NASA Office of Space Science, January 16, 1998.

4 Introduction

GLAST is a next generation high energy gamma-ray observatory designed for making observations of celestial gamma-ray sources in the energy band extending from 20 MeV to more than 300 GeV. It follows in the footsteps of the Compton Gamma Ray Observatory EGRET experiment, which was operational between 1991-1999. The GLAST Mission is part of NASA's Office of Space and Science Strategic Plan, with launch anticipated in 2006. The principal instrument of the GLAST mission is the Large Area Telescope (LAT) that is being developed jointly by NASA and the US Dept. of Energy (DOE) and is supported by an international collaboration of 26 institutions lead by Stanford University.

The GLAST LAT is a high-energy pair conversion telescope that has been under development for over 7 years with support from NASA, DOE and international partners. It consists of a precision converter-tracker, CsI hodoscopic calorimeter, plastic scintillator anticoincidence system and an associated data acquisition system. The design features a 4x 4 array of identical tracker and calorimeter modules. The modules are ~ 38 x 38 cm. Figure 1 shows the LAT instrument concept.

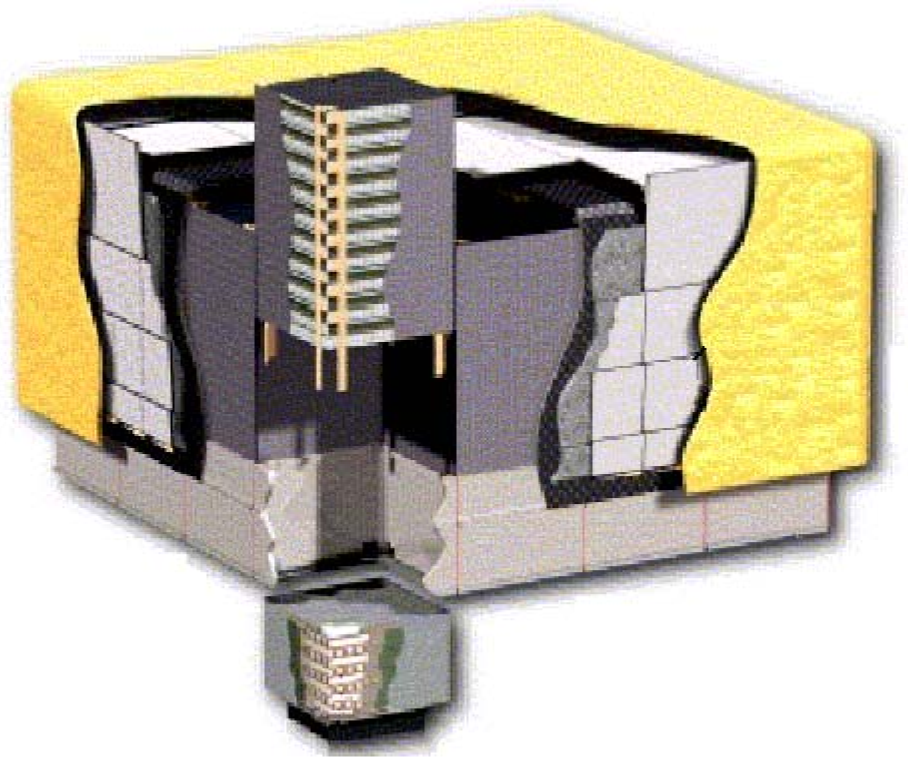


Figure 4.1 View of the LAT Science Instrument
One Tracker tower module and one Calorimeter module are pulled away from the Grid. GLAST is a 4 x·4 array of identical Tracker and Calorimeter modules.

4.1 LAT Science Requirements

The GLAST science requirements are given in the “GLAST Science Requirements Document”. An updated set of requirements, as they pertain to the LAT science instrument, is specified in “LAT Instrument Performance Specification”. General constraints and requirements on the instrument design are specified in GLAST mission documents. The flowdown of the science requirements and instrument constraints to the LAT design is summarized in Foldout-D of our NASA proposal. The requirements that most strongly impact the calorimeter design are those pertaining to the energy measurement domain, the energy resolution, background rejection, and the dead time.

4.2 LAT Technical Description

The LAT science instrument consists of an Anti-Coincidence Detector (ACD), a silicon-strip detector Tracker (TKR), a hodoscopic CsI Calorimeter (CAL), and a Trigger and Data Flow system (T&DF). The principal purpose of the LAT is to measure the incidence direction, energy and time of cosmic gamma rays while rejecting background from charged cosmic rays and atmospheric albedo gamma rays and particles. The data, filtered by onboard software “triggers”, are streamed to the spacecraft for data storage and subsequent transmittal to ground-based analysis centers. The Tracker provides the principal trigger for the LAT, converts the gamma rays into electron-positron pairs and measures the direction of the incident gamma ray from the charged-particle tracks. It is crucial in the first levels of background rejection for providing track information to extrapolate cosmic-ray tracks to the ACD scintillator tiles, and it is important for further levels of background analysis due to its capability to provide highly detailed track patterns in each event. The primary tasks of the GLAST calorimeter are to provide an accurate measure of the energy of the shower resulting from pair conversion of incident gamma rays in the tracker, and to assist with cosmic-ray background rejection through correlation of tracks in the silicon tracker with the position of energy deposition in the calorimeter. The calorimeter also provides triggers to the LAT, particularly for very large energy depositions.

5 Science Analysis Software Conceptual Design

The Science Analysis Software comprises several components

- Data Pipeline
    - Prompt processing of Level 0 data through to Level 1 event quantities
    - Providing near real time monitoring information to the IOC
    - Monitoring and updating instrument calibrations
    - Reprocessing of instrument data
- Performing bulk production of Monte Carlo simulations
- Higher Level Analysis
    - Creating high level science products from Level 1 for the PI team
    - Providing access to event and photon data for higher level data analysis
- Interfacing with other sites (sharing data and algorithms)
    - mirror PI team site(s)
    - SSC
- Supporting Engineering Model and Calibration tests
- Supporting the collaboration for the use of the tools

The IOC and Data Processing Facility are co-located at SLAC/Stanford. The SSC will be located at Goddard Space Flight Center.

Operations during flight will see the telemetered data delivered to the IOC from the ground station. The IOC will packet sort and error correct the raw data, passing Level 0 data to the Data Processing Facility (DPF). An automated server will notice the arrival of new data and pass it through reconstruction, creating Level 1 data, which is input to a database shared with the SSC. This processing will also facilitate monitoring and updating of calibrations as well as providing high level diagnostics back to the IOC Operations crew on a near real-time basis. Higher level science analysis operates from the shared databases and is performed (and shared) by both the LAT and SSC teams.

In addition, the DPF will be able to generate large volumes of Monte Carlo simulations to be used for algorithm development, understanding of instrument performance and for science analysis studies.

This flow of data is shown in Figure 5.1.

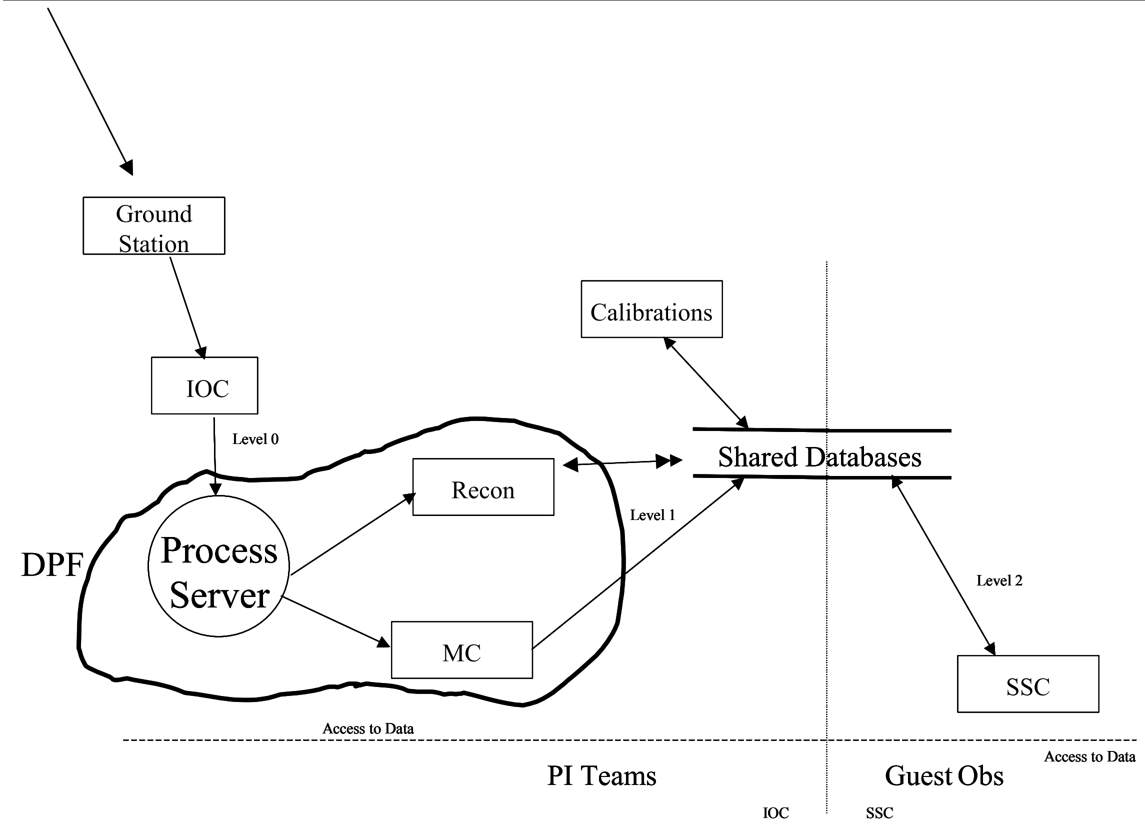


Figure 5.1. Overview of SAS Operation.
Event data passes through the IOC and is handled by a fully automated server, processing the Level 0 data through to Level 1. A relational database maintains the state of the server and the datasets. The output of Monte Carlo and Reconstruction goes into an event database which is shared with the Science Center for higher level analyses.

There are three major components to the SAS: instrument simulation and event reconstruction; an operations facility to do automated event processing; and tools to perform higher level analyses on the processed data. In addition, there are utility elements for access to data, low level analysis tools and support of the user community.

5.1 Instrument Simulation and Event Reconstruction

Event reconstruction uses calibration constants to convert raw data, either from the instrument or from simulations, to physical units and suppress bad readouts. It performs pattern recognition and fitting procedures to interpret the data in terms of interacting particles in the instrument, and determines their identity, direction and energies as well as possible. There is a tradeoff between photon efficiency and purity in this process as selection criteria are applied to the events to suppress charged particle background.

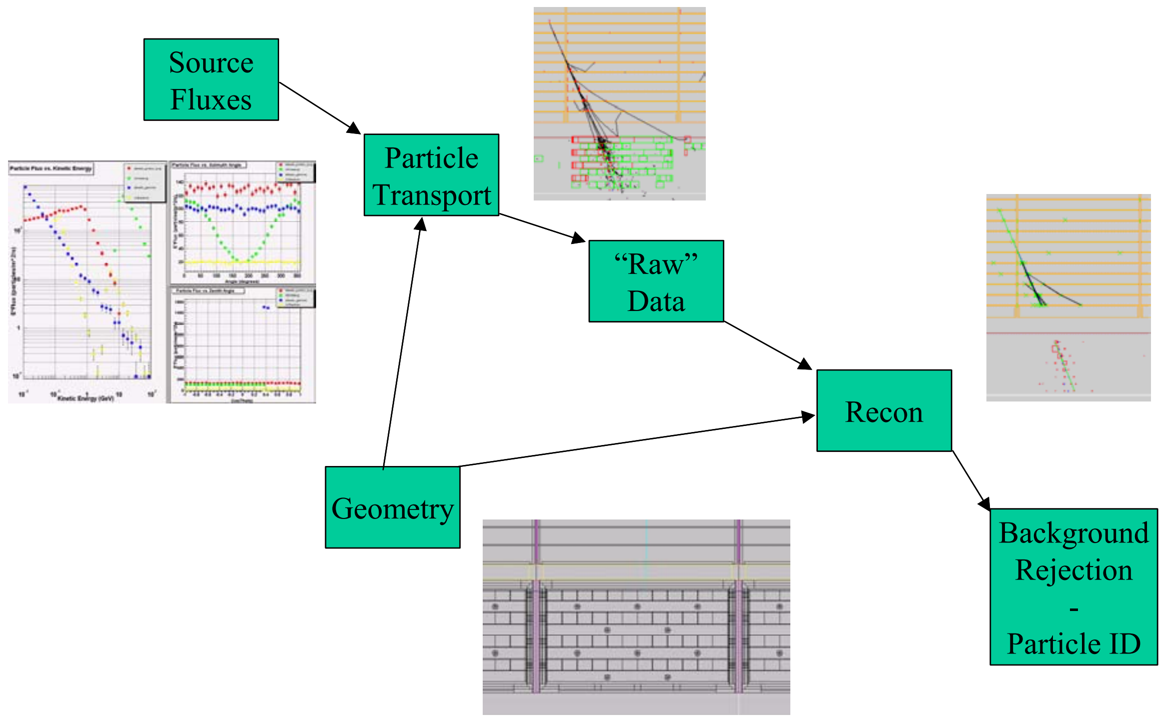


Figure 5.2 Event Reconstruction Chain.
Raw data either arrives via the IOC from the telemetered event data, or as generated Monte Carlo. The data are passed through reconstruction to interpret the instrument response and find gammas and particles. Background selection cuts are applied for final particle identification.

Simulation is an alternate source of raw data for the reconstruction. It takes models of the fluxes of particles incident on the instrument, coupled with a physical description of the device and performs Monte Carlo simulation of the passage of the particles through the materials. The output of this simulation is then formatted as raw data, in the same form as seen from the real instrument. This tool provides the ability to assess the expected performance of GLAST, develop algorithms in a controlled environment, and to estimate efficiencies and purities of the final products.

5.2 Operations Facility

During routine flight mode, there will be downloads of data twice daily from the spacecraft to the ground station and thence to the IOC. The DPF will take Level 0 data (raw data corrected for transmission effects) from the IOC and automatically perform the event reconstruction on it.

The facility will also tag, and make available, appropriate events for performing calibrations (eg high energy non-interacting, heavy cosmic rays. It will keep track of high-level diagnostics (e.g. correlated performance of instrument subsystems that are not available from the raw data alone) and make them available to the IOC Operations staff for monitoring purposes.

Prior to flight, prototypes of the facility will handle engineering model data.

All flight data will be catalogued and made available to the SSC.

5.3 Science Products

The collaboration is responsible for a number of high level data products. The table below summarizes the science data products higher than level 0 that the SAS will produce. Several of the data products will be stored as databases, primarily because they are large datasets that will need to be accessed in ways other than time order. The exact list of products is still subject to revision.

Data Product	**Description**
Instrument response functions	Effective area, energy resolution, energy redistribution, and point-spread function for all gamma-ray event types
Timeline (as observed)	Observing mode and spacecraft position & orientation as a function of time. Command and performance states.
Source catalog	Positions, fluxes, and uncertainties for all detected sources in the sky survey. Includes flux histories, spectral indices, and identifications
GRB/transient alerts	Most initial GRB and bright AGN flare alerts will be generated on the spacecraft; these SAS alerts will provide refined information, or for many AGN flares, the initial notification.
Interstellar emission model	The interstellar emission model is only loosely speaking a data product; it will be refined as necessary using flight data. It is essential for the production of the source catalog, and for likelihood analysis of GLAST gamma-ray data in general, so in any case it is a deliverable.

Table 5.1 Data Products

5.4 Infrastructure, Analysis Tools and User Support

A large component of the software effort is providing the framework within which the various products are developed, the tools to examine the data and the support for users in exercising the software. This will be reflected in the remainder of the document.

Infrastructure refers to the code version and management tools, as well as the rules for the syntax and structure of the code. Version control all permits tracking snapshots of the code through its life. The code is arranged into modules, or packages, to permit fine-grained control. The code management tools provide the capability for enforcing the modularity of packages and providing a standardized build environment for our supported operating systems. The code architecture specifies the rules that the code itself must live by – what standard functions an algorithm must provide and so on. This standardization encourages uniformity throughout the code, allowing easier maintenance.

This organization is also responsible for providing support to the collaboration user community and to the SSC members in their support of the general community. Support comes in a variety of categories, but mostly based on documentation: reference for developers and guides for general users. We support internal documentation embedded in the code itself and then extracted into a web readable format. This is the lowest level of developers manual. We will create an overview manual for developers. For the users, we will prepare extensive guides to accessing the data and to the operation of the software.

6 Deliverables

The SAS will deliver the following:

- Simulation
  - models of the expected particle and gamma-ray fluxes for the flight instrument
  - geometry models of the instrument and spacecraft, and of engineering models
  - simulation of particle transport through those geometries
- Reconstruction of events from data or Monte Carlo for the flight instrument and engineering models. This includes emulation of the trigger, and interpretation of the events in terms of particle content.
- Calibration algorithms and the software to store and access the calibrations, most typically based on time of applicability.
- Detailed and summary output from reconstruction, sufficient to understand the reconstruction results, as well as particle information including type, direction, energy and error estimates.
- Data Production Facility
  - automated server to handle Level 0 to Level 1 processing
  - high level instrument diagnostics to feed back to IOC Ops in near real time
  - database to catalogue the state of the server as well as of input and output datasets
- Science Tools
  - See Table 5.1.
- Infrastructure
  - code architecture and coding rules
  - code development and release management tools, including code repository, code management tool, release management and verification tools
  - low-level (Level 0 and 1) analysis tools with access to data and event display for visualization
- Documentation and support of the collaboration user community for the use of the above deliverables

7 WBS, Timeline and Schedule

7.1 WBS - 4.1.D

4.1.D - SAS

4.1.D.1 - Sources, Simulation and Reconstruction	.1 Sources Japan, UW, SLAC .2 Proto Initial Framework - All .3 Gismo sim UW .4 GEANT4 sim Italy .5 ACD GSFC .6 CAL NRL, France .7 TKR SLAC, Italy .8 Trigger GSFC, SLAC .9 Background Rej - GSFC, SLAC
4.1.D.2 - Analysis Tools and Infrastructure	GSFC, Stanford/SLAC
4.1.D.3 - Engineering Model Support	UW, Italy, Japan, Stanford/SLAC, France, NRL, GSFC
4.1.D.4 - Science Tools	.1 base utils GSFC, Stanford/SLAC .2 analysis tools GSFC, Stanford/SLAC .3 databases GSFC, Stanford/SLAC
4.1.D.5 - Data Production Facility	.1 auto server Stanford/SLAC .2 instrument diags Stanford
4.1.D.6 – Calibrations	.1 Tools SLAC .2 ACD GSFC .3 CAL NRL, France .4 TKR Stanford, Italy
4.1.D.7 – Management	Stanford

Table 7.1 WBS and institutional responsibilities.
Named institutions are taking the lead in these WBS areas.

The following table shows the relationships between the Level 3 requirements and our WBS. The Level 3 requirements are held in document LAT-SS-00020-00.

Requirement #	Description	WBS #	Description
5.1	prompt processing	4.1.D.5	Data Processing Facility
	near-real time monitoring	4.1.D.5	Data Processing Facility
	monitoring & updating calibs	4.1.D.5	Data Processing Facility
		4.1.D.6	Calibrations
	maintain state & performance tracking	4.1.D.5	Data Processing Facility
	high level science products	4.1.D.4	Science Tools

	reprocessing data	4.1.D.5	Data Processing Facility
	access to events	4.1.D.5	Data Processing Facility
		4.1.D.2	Analysis Tools
	perform bulk MC simulations	4.1.D.5	Data Processing Facility
	interface with mirror sites	4.1.D.5	Data Processing Facility
		4.1.D.2	Analysis Tools
	interface with SSC	4.1.D.5	Data Processing Facility
		4.1.D.2	Analysis Tools
	support engineering models	4.1.D.3	Eng Model support
5.2	Code Development	4.1.D.2	Analysis Tools
5.3	Instrument Response Simulations	4.1.D.1	Sources, Simulation and Event Reconstruction
5.4	Event Reconstruction	4.1.D.1	Sources, Simulation and Event Reconstruction
5.5	Environment Logging	4.1.D.5	Data Processing Facility
5.6	Calibrations	4.1.D.6	Calibrations
		4.1.D.5	Data Processing Facility
5.7	Level 1 Processing	4.1.D.5	Data Processing Facility
5.8	Creation of High-Level Science Tools	4.1.D.4	Science Software
5.9	Analysis Platform	4.1.D.2	Analysis Tools
5.10	Longevity	4.1.D.5	Data Processing Facility
		4.1.D.2	Analysis Tools
6.1	Data Formats	4.1.D.5	Data Processing Facility
6.2	LAT Data & Alg Export	4.1.D.5	Data Processing Facility
		4.1.D.2	Analysis Tools
6.3	SSC interface	4.1.D.5	Data Processing Facility

Table 7.2 Cross reference of Level 3 requirements to WBS

7.2 Schedule and Milestones

The figure below shows a high level schedule for the major work elements. The immediate effort involves the simulation, reconstruction and related tools (eg event display), and code infrastructure (eg release management); medium term is the DPF, Calibrations and deciding on event database technology; the long term is occupied by building the science analysis tools and polishing up the short and medium term items. As the Science Tools become better defined, milestones relevant to them will be added to this list.

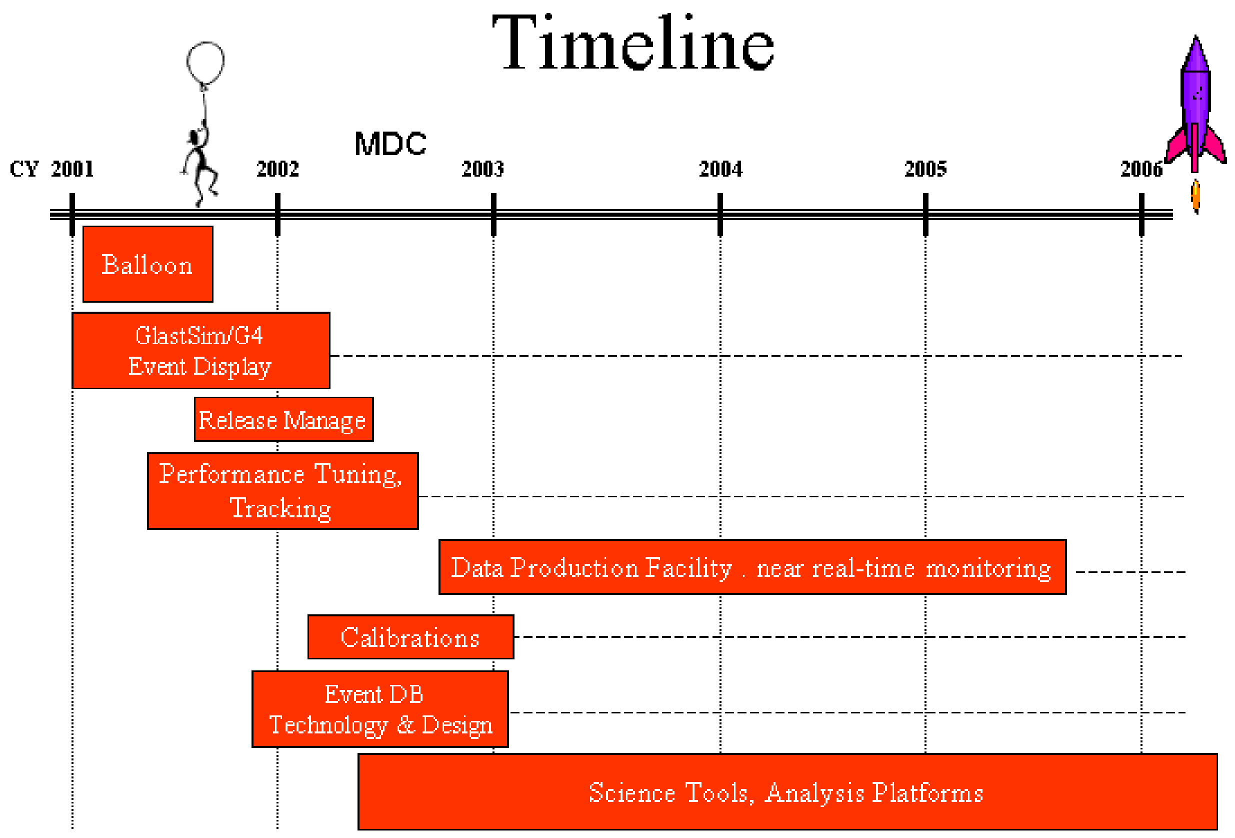


Figure 7.1 Effort timeline.
Short term activities involve balloon support; upgrades to simulation and reconstruction; and code release management. Longer term efforts will involve calibrations, science tools and instrument performance tracking.

Milestone	Date
Science Analysis Software (SAS) Requirements Review	04/20/01
Start PDR Instrument Performance/Backgrounds Evaluation	05/01/01
SAS PDR	08/17/01
LAT Instrument PDR	01/08/02
Release Management & Verification in place	03/02
Simulation/Reconstruction 1st iteration complete	4/02
Low-level calibration algorithms in place	6/02

LAT Instrument CDR	8/02
SAS CDR	9/4/02
Simulation/Reconstruction 2nd iteration complete	10/02
Annual Instrument Performance Evaluation	12/02
Photon Database technology implemented	1/03
Generate Instrument Response Function	5/03
Annual Instrument Performance Evaluation	8/03
Prototype of Data Processing Facility	1/04
Annual Instrument Performance Evaluation	8/04
Annual Instrument Performance Evaluation	8/05
All required Science Tools in place	12/05
Production version of Data Processing Facility	1/06
End to End pre-launch test completed	1/06

Table 7.3 SAS Milestones

7.3 Manpower

Shown below is the manpower estimate vs time. At present, most of the effort is going into simulation/reconstruction, analysis tools and user support. As we move into the DPF, Calibration and Science Tools areas, we will need to expand the effort. This will be further discussed in the Open Issues section.

Here are the assumptions made for the manpower estimate. SAS occupies a unique position in the collaboration in that not only does it have deliverables, but it is also a service organization. This implies an element of ongoing support for users, as well as for upgrades, maintenance and support of the software products.

7.3.1 Sources, Simulation & Recon

- 1 FTE GEANT4 during the development phase and 1/2 FTE ongoing (Italy)
- 1/2 FTE sources (Japan)
- 2 FTE CAL - simulation and reconstruction (NRL, France)
- 2 FTE TKR - simulation and reconstruction (SLAC + Italy)
- ½ FTE for ACD - simulation and reconstruction (GSFC)

- 1 FTE combined for Trigger, Background Rejection studies (there may be odd scientists contributing as well)

7.3.2 Analysis Tools & Infrastructure

- 1 FTE for tools development (GSFC)
- 1 FTE package & user support (SLAC)
- 1 FTE code release & verification (SLAC)
- 1 FTE event display (6 months, Italy)

7.3.3 Engineering Models (BTEM, BFEM, 4 Module test 2003? …)

- 1 FTE extra for duration of each test

7.3.4 Science Software

- 3 FTEs to support 7 scientist programmers (estimated 40 MY work). Starting in FY '02.

7.3.5 Data Processing Facility

- 1 FTE automated server (~1 year) – main activity in FY02, followed by a burst a year before launch
- 1 FTE instrument diagnostics – build up 1.5 yr before launch
- 1/2 FTE support of DPF pre-launch

7.3.6 Calibration (starts 3/2002)

- 1 FTE for machinery for calibrations
- 2.5 FTE for subsystem algorithm development (perhaps more – from NRL, France & Italy).

7.3.7 Management

- 1 FTE code architect
- 1 FTE manager

In more detail, this results in the effort vs time chart shown below. Note that, due to budget cuts, we cut 1 FTE from User Support in FY'02 and delayed the start of the Data Processing Facility until the beginning of FY'03.

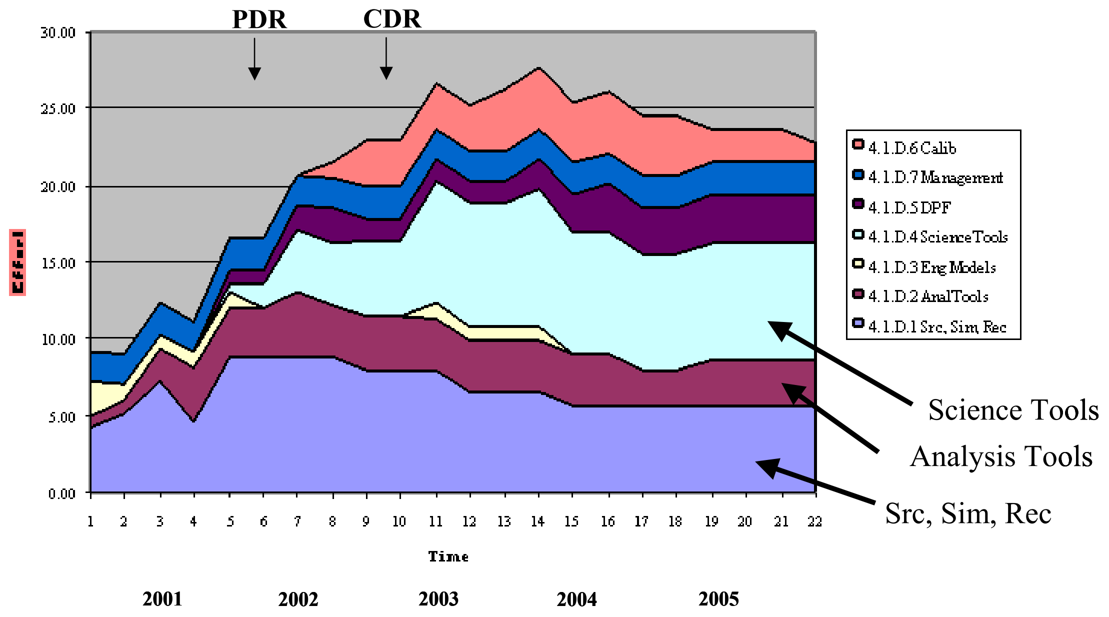


Figure 7.2 Effort (FTE) vs time.
Our effort is expected to peak around 25 FTEs as we develop the science tools.

8 Organization

Science Analysis Software is performed primarily at Stanford (SLAC and campus), GSFC, U Washington, NRL, IN2P3 (France), INFN (Italy), and Hiroshima and ISAS in Japan. Richard Dubois (SLAC) is the subsystem manager. Toby Burnett of U Washington is instrumental in acting as code architect. Heather Kelly at GSFC oversees the low level analysis tools and infrastructure. Seth Digel at GSFC oversees the development of high level science analysis tools. Karl Young at SLAC is managing the development of the Data Processing Facility prototype as well as developing the Code Release Management tools. NRL and France contribute CAL subsystem code; SLAC and Italy contribute TKR code; and GSFC contributes ACD code. Italy oversees the Event Display and GEANT4 simulation package work, while Japan is responsible for the physics models of particle fluxes.

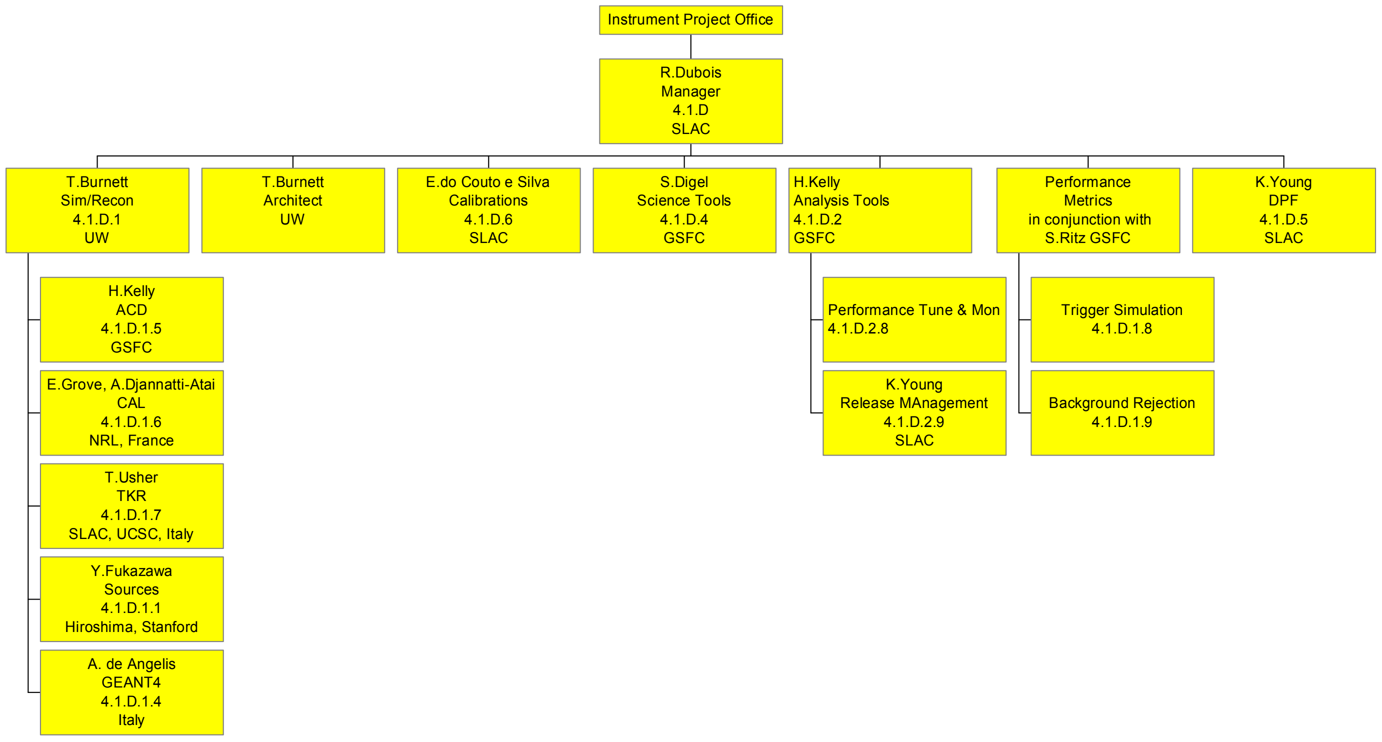

Instrument Project Office
R.Dubois Manager 4.1.D SLAC
T.Burnett Sim/Recon 4.1.D.1 UW
T.Burnett Architect UW
E.do Couto e Silva Calibrations 4.1.D.6 SLAC
S.Digel Science Tools 4.1.D.4 GSFC
H.Kelly Analysis Tools 4.1.D.2 GSFC
Performance Metrics in conjunction with S.Ritz GSFC
K.Young DPF 4.1.D.5 SLAC
H.Kelly ACD 4.1.D.1.5 GSFC
E.Grove, A.Djannatti-Atai CAL 4.1.D.1.6 NRL, France
T.Usher TKR 4.1.D.1.7 SLAC, UCSC, Italy
Y.Fukazawa Sources 4.1.D.1.1 Hiroshima, Stanford
A. de Angelis GEANT4 4.1.D.1.4 Italy
Performance Tune & Mon 4.1.D.2.8
K.Young Release MAnagement 4.1.D.2.9 SLAC
Trigger Simulation 4.1.D.1.8
Background Rejection 4.1.D.1.9

Figure 8.1 Organization Chart

Our group is geographically dispersed, so we maximize our connectivity by

- static use of the web, with a central software home page with collected information on projects, status, meetings, etc. Everything we do can be seen from http://www-glast.slac.stanford.edu/software.
- web conferencing: our meetings are all held via CERN's VRVS system, with weekly general and core meetings and meetings as needed for the subsystems. Minutes and presentations are archived as well.

- instant messaging: we use the ICQ IM tool to facilitate chat and quick online point-to-point contact.
- we hold three software workshops per year, wherein all areas of software are discussed, including detailed reviews of projects.
- between the general workshops, we hold working meetings of the core (ie non-subsystem) group.

9 Prototyping Experience and Design Decisions

There are several areas in which prototypes were developed before proceeding with design and development of the SAS.

9.1 ROOT I/O and Analysis in TB99

One of the casualties of the transition from Fortran to C++ was built-in data persistence: C++ has not native, process independent binary form of complex data structures suitable for I/O. Two examples of persistent object store mechanisms have come into recent use: ROOT and Objectivity. Objectivity requires too much support and maintenance for our resources. ROOT is freely available and has a growing user community. It was successfully used for the 1999-2000 BTEM test. It satisfied our needs for a structured output of objects. We also made use of ROOT's capabilities for analysis (access to the data, histogramming, fitting etc). This was also shown to fit our needs, though our lack of experience with the product showed itself in the limited ways we were able to use ROOT. See Sec 10.1.3 for a fuller description.

9.2 Gismo Simulation

Early in GLAST's development, it was decided to commit to the C++ computer language. This also included development and support of a C++ simulation package for following particle trajectories and interactions in complex geometries, since none existed at the time. In essence, a package was constructed to supply similar features to those provided by GEANT3, with some improvements in the geometry handling. This new package, Gismo, has been in use for several years now, and has proved adequate to the task of designing the instrument and interpreting test beam data. See Sec 10.3.3 for a fuller description.

9.3 GEANT4 Simulation for BFEM

While Gismo has proven adequate for the instrument design and is in reasonable agreement with test data, it suffers from a major deficit: it is unique to GLAST. As it stands, it is supported (and understood) by a single person and is mostly undocumented. GEANT4, the replacement for GEANT3, is in the late stages of development now and is sure to become the world standard in this area, as its predecessor was. It is much better documented, supported by a large collaboration, and is coming into wide use in the community. It was decided to use G4 as the simulation package for the 2001 BFEM program as a prototype. We have seen that it can well describe the instrument and have done validations, as described in http://www-glast.slac.stanford.edu/software/PDR/SAS/g4prot.htm, indicating that G4 should be up to the task.

9.4 Recon

As for the simulation tools, a reconstruction package has been part of the GLAST development toolkit for some years.

The event reconstruction can be broken down into three coupled components: finding tracks and photon candidates in the TKR; estimating the energy and enhancing the background rejection in the CAL; and rejecting charged particle backgrounds in conjunction with TKR tracks in the ACD.

For the CAL this means optimizing the energy resolution and maximizing the effective area. Additional background rejection power can be obtained from discriminant variables based on shower moments, topology and clustering. The CAL can operate in concert with the TKR (in its guise as a sampling calorimeter) to improve the PSF at low energies. Finally the CAL can provide energy and direction information for high energy CAL-only interactions. More details can be found in Sec 10.3.

9.5 Gaudi Code Architecture

For a code system as large and complex as needed for the simulation and reconstruction, a code framework is essential. It supplies basic rules of behavior for modules; philosophy for dealing with data and algorithms; interface strategy for in-memory ("transient") classes versus on-disk ("persistent") versions; and services for useful utilities (like messaging, random number handling and so on). A promising candidate – Gaudi, developed for the LHCb experiment at CERN's upcoming LHC collider - has appeared. It has now been extended for use by several collaborations and is Open Source, so it is no longer specific to LHCb. As for GEANT4 and ROOT, it now has a wide community of users and is well documented. Our initial prototype made use of Gaudi's data and algorithms base classes; used ROOT for a persistent store (using supplied conversion macros); and adopted the messaging and random numbers service. These were incorporated into a revised version of the simulation and reconstruction code. More details are presented in Sec 10.1.2.

9.6 MySQL relational database for constants handling

When interpreting the raw data, the reconstruction process needs to convert electronic readouts to physical units. This involves scale factors (gains) and offsets (pedestals) in the CAL and ACD, and strip positions (alignment) and hot/dead strip lists in the TKR. These conversion and alignment factors will change over time as the instrument ages. A system must be devised to allow access to these constants, using time as an index. An initial prototype was devised to use a relational database (MySQL) to hold meta-data describing times of validity and pointers to calibration files. The issues involved were the richness of the database to describe the meta-data and the ability to access it from running processes. Such databases are in wide use in HEP experiments, so it was no surprise that this mechanism should work for us too. Details are presented at http://www-glast.slac.stanford.edu/software/PDR/SAS/prototyping_MySQL.

10 Architecture Description

10.1 Infrastructure

A very important, and often overlooked, aspect of software development is the development environment. Our approach has been to simply adopt the standard packages that will be described below: CVS, CMT, and Gaudi. An important requirement for each was support for all supported operating systems, namely Windows NT/2000, Linux, and Solaris.

These three were selected to

- ensure that we are not dependent on a single vendor
- expose problems that might not be apparent under a single OS and compiler
- exploit the excellent code development environment on Windows
- exploit the popularity of Linux
- allow us to use the OS that comprises most of the SLAC batch farm

10.1.1 File management: CVS

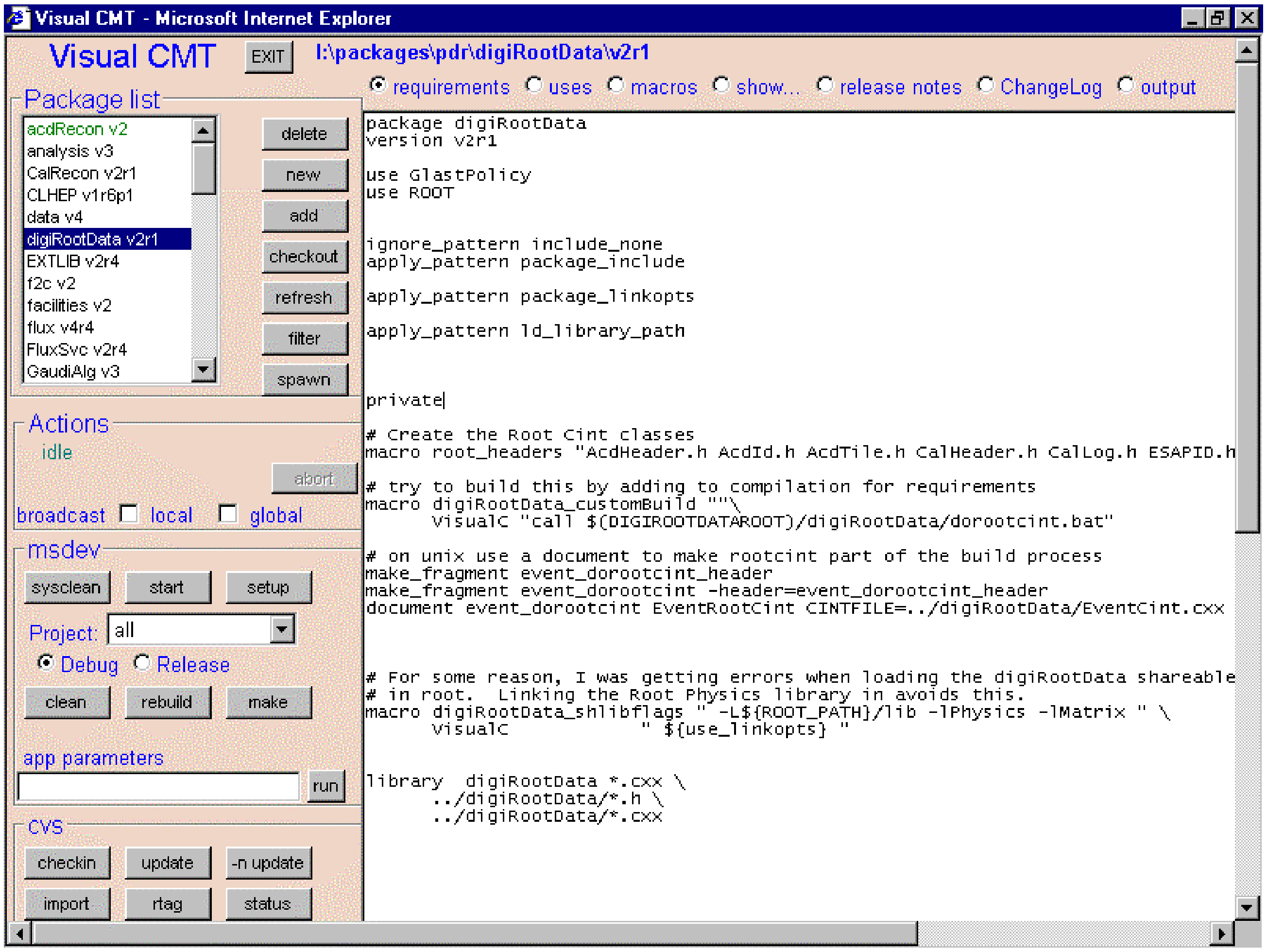
We use the industry standard Concurrent Versions System (CVS) for archiving and keeping track of versions of the code. While it is in some sense an independent decision, it must interoperate smoothly with the code management system. For more information about CVS, please see http://www.cvshome.org.

10.1.2 Code management: CMT

Following a survey of code management systems in use by high energy physics experiments, we decided to standardize on the Code Management Tool (CMT). It provides a consistent definition of a package of software; and a way to define and manage versions and dependence on other packages; and the means to generate the associated binaries. A package is a set of related components, grouped into a unit for versioning and interface purposes. Unlike any of the other systems we examined, CMT has support for three types of inter-package dependence:

- compile-time: specifying paths to find include files.
- Link-time: specification of link options, locations of binaries
- Execution-time: passing the definitions of environment variables to the executable.

CMT provides a command-line interface. While this interface provides all of the functionality necessary, a GUI interface would provide an easier mechanism for users. A prototype GUI has been developed in JavaScript. This program, called VCMT, is currently in use by our Windows users. Work is underway to provide a JAVA version of VCMT that is usable on UNIX platforms as well. Fig 10.1.1 is a screen shot of the VCMT GUI.



```
package digiRootData
version v2r1

use GlastPolicy
use ROOT


ignore_pattern include_none
apply_pattern package_include

apply_pattern package_linkopts

apply_pattern ld_library_path



private

# Create the Root Cint classes
macro root_headers "AcdHeader.h AcdId.h AcdTile.h CalHeader.h CalLog.h ESAPID.h

# try to build this by adding to compilation for requirements
macro digiRootData_customBuild ""\
      VisualC "call $(DIGIROOTDATAROOT)/digiRootData/dorootcint.bat"

# on unix use a document to make rootcint part of the build process
make_fragment event_dorootcint_header
make_fragment event_dorootcint -header=event_dorootcint_header
document event_dorootcint EventRootCint CINTFILE=../digiRootData/EventCint.cxx



# For some reason, I was getting errors when loading the digiRootData shareable
# in root.  Linking the Root Physics library in avoids this.
macro digiRootData_shlibflags " -L${ROOT_PATH}/lib -lPhysics -lMatrix " \
      VisualC              " ${use_linkopts} "


library  digiRootData *.cxx \
      ../digiRootData/*.h \
      ../digiRootData/*.cxx
```

Figure 10.1.1 VCMT GUI

Each of the buttons visible on the screen generates commands to CMT, CVS, or the development tool msdev, usually dependent on the package that is selected. For example, the “make” button in the msdev section will make the project selected in the Project window. If an application project is selected, it can be simply run with the “run” button. One of the more useful functions is the “start” button that starts up msdev itself with a workspace based on the selected package.

CMT also provides a seamless interface to CVS and is supported on all of our required platforms. CMT has been a considerable improvement over our previous unstructured setup. We are still learning how to use it most effectively. For more information concerning CMT, please see: http://www.lal.in2p3.fr/technique/si/SI/CMT/CMT.htm.

10.1.3 Software framework: Gaudi

One of the more important recent decisions was to adopt a well-designed and documented framework, Gaudi. It is an application framework designed to facilitate event-oriented analysis, with a lot of attention to allowing modular development and deployment of processing algorithms. The LAT simulation/analysis program that was used for the proposal to NASA for the LAT suffered from the lack of attention to this requirement. We learned that a system that can be managed easily by a few people may not scale to being accessible to many.

The Gaudi framework provides standard interfaces for the common components necessary for event processing and analysis. The key components of the Gaudi framework are Algorithms, Data Objects, Converters, and Services. Converters translate data between representations, usually a persistent representation and a transient one. Services are common utilities that may be used by many algorithms. Fig 10.1.2 is a diagram of the Gaudi framework as in use by the LHCb project.

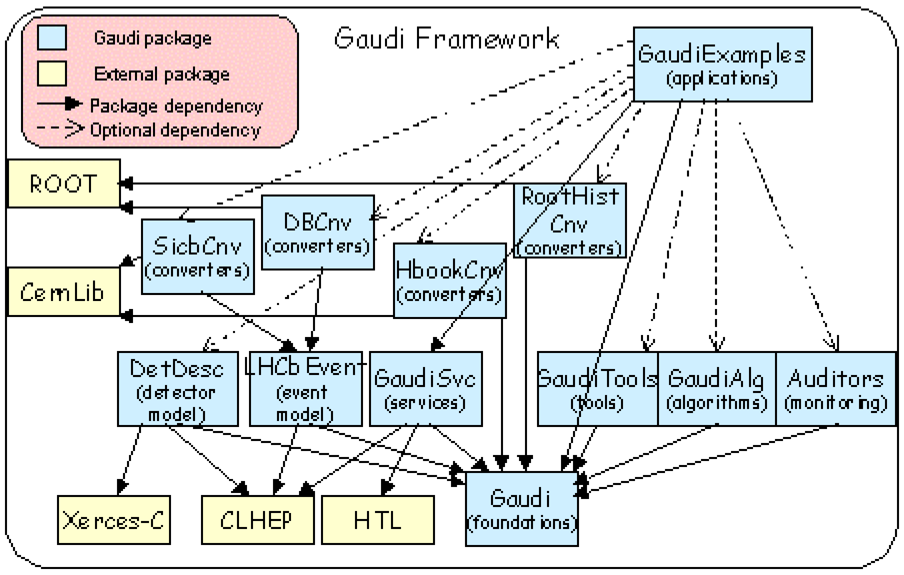


Figure 10.1.2 Gaudi framework for LHCb.

Gaudi embraces the notion that data and algorithms are separate, and provides base classes encapsulating data-like or algorithmic objects. This facilitates flexibility in that different algorithms can manipulate the same data. In the evolution from an informal system used by only a few developers to a formal, well-documented system which must serve a large pool of developers, the data structures for Monte Carlo "truth", raw data, and reconstructed quantities need to be clearly specified and documented. Gaudi provides a very nice model for this with the concepts of the transient data store (TDS), and converters to persistent representations of the data. The transient data store provides a shared memory mechanism allowing algorithms to share data.

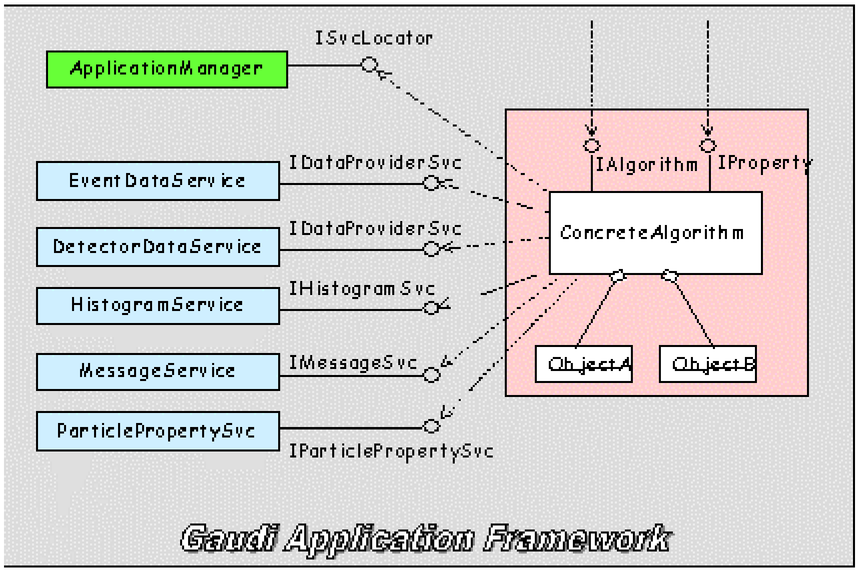


Figure 10.1.3 Gaudi algorithm and its relationship to the data services.

Fig 10.1.3 demonstrates how an algorithm interacts within the Gaudi framework to access data. The Event Data Service provides the interface between algorithms and the transient data store. It also illustrates how a class, *ConcreteAlgorithm* in the above, implements interfaces (the lines connected to circles), and requests services (the lines ending in arrows).

Why Gaudi? It satisfies the requirements for flexibility, modularity, and extendibility. First it provides flexibility, in that at run time users can determine which components are required. For example, it is trivial to replace one set of reconstruction algorithms with a different set. Gaudi uses dynamic linking extensively; thus at runtime, unused components are never loaded. The Gaudi framework provides common utilities and modularity to our simulation and analysis code. The persistency mechanism allows one to choose the persistency format at run time. The service which handles persistency is also a good example of the features that Gaudi provides.

The Gaudi Event Persistency Service handles reading in and writing out data in a persistent format. For example, the raw detector data available can be ingested into the Gaudi framework through the Gaudi Event Persistency Service. This service reads in data available in some persistent format, in this case ROOT. The data is then stored in the Transient Data Store (TDS), available to all algorithms and services that desire to use the data. Hence, the same reconstruction algorithms can be used on any of test beam, balloon, simulation, or flight instrument data. The ROOT I/O and data flow within the Gaudi framework is illustrated in Fig 10.1.4:

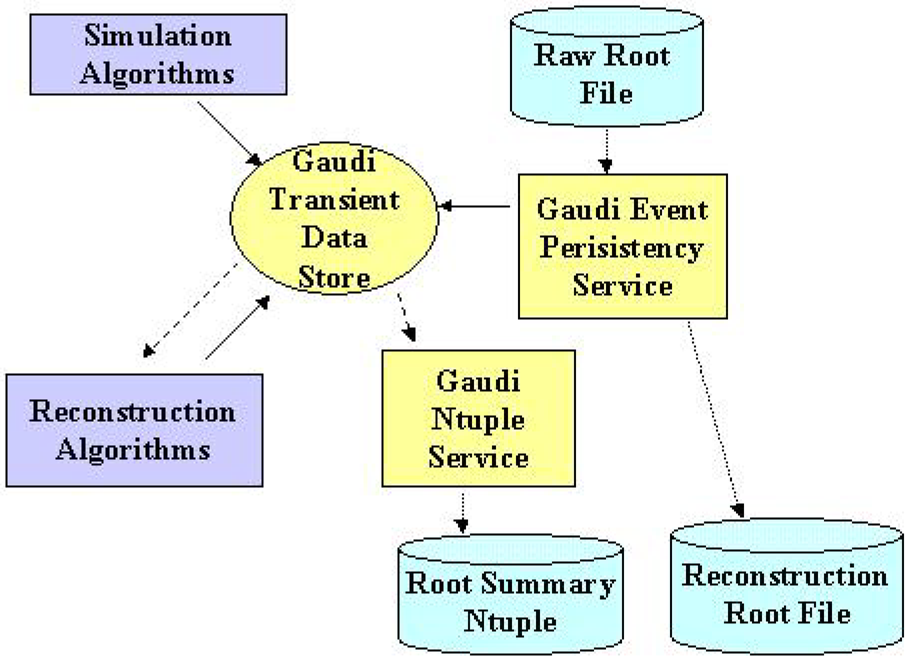


Figure 10.1.4. Data flow within the Gaudi framework.

Our intention is to avoid tying ourselves to a particular I/O mechanism. The Gaudi framework provides a layer between simulation and reconstruction algorithms and the I/O services. If at some later time we decide to pursue other I/O options new services will be added to our framework with no impact on our existing data manipulation routines.

Gaudi is supported and developed in the context of two of the major LHC experiments, LHCb and ATLAS. The current design clearly separates experiment-independent parts, which are managed in a shareware mode. Clearly, details of the event data definitions and geometry structures are experiment-dependent, but such code resides in separate packages. Using LHCb's code as a model, we have defined our own GLAST-specific packages for event data definitions and geometry.

Migrating to Gaudi has not come without some cost. There was a steep learning curve. The documentation, while good, has improved as we have learned more about the system. The Gaudi developers are available to exchange ideas and provide guidance when we encounter problems. Much time has been spent modifying our existing code to conform to the format that Gaudi requires. While this may sound like a heavy penalty, much of our code did not conform to our requirements for flexibility, modularity, and extendibility. A complete re-write was necessary. In fact using the Gaudi framework saved coding time, in that we used the common interfaces that Gaudi provides. We have completely migrated our simulation and reconstruction code.

For more information about the Gaudi framework, please see http://proj-gaudi.web.cern.ch/proj-gaudi/.

10.1.4 ROOT

For object I/O, we have chosen ROOT. It is an object oriented framework designed to store and manipulate large amounts of data using a self-describing machine independent format. In addition, ROOT offers analysis tools such as display creation, histogramming, and function fitting. See http://root.cern.ch for complete information on the ROOT system.

10.1.4.1 I/O Features

There were a number of reasons we chose to use ROOT for I/O:

- ROOT files are machine independent.
- ROOT files are self-describing – files created today will still be readable years from now.
- On the fly compression – ROOT uses an algorithm based on gzip.
- Support for object I/O – the detailed structure of our data is preserved for analysis.
- ROOT supports schema evolution (the change of class definitions with time).

More detailed information about the ROOT file format is available in our document "A description of ROOT for GLAST", http://www-glast.slac.stanford.edu/Software/root/howto/.

10.1.4.2 ROOT Object I/O

ROOT supports object I/O, meaning we can store the detailed tree class structure of our data directly in ROOT files. ROOT has the feature of branched I/O, wherein one can read in a single branch of a tree, and read in the rest of the event only if there is something of interest found in the original branch. This can give a tremendous savings in I/O time. An example would be to read the TKR branch, and only read the CAL data if there was a gamma found by the tracker. ROOT's support for object I/O is clearly advantageous during analysis, versus the use of flat files. It is also consistent with our use of C++ as our primary programming language. ROOT is now commonly used by many high-energy particle physics experiments, and it has a growing, supportive community of users.

To use ROOT's object I/O, class definitions are provided in C++. Some time has been spent developing our own class structures. Clearly class definitions may change over time. However, ROOT provides a mechanism to track schema evolution and the files are self-describing. This guarantees that old files can always be read by new versions of the class libraries. This point is imperative, as over time we expect that there will be extensions and modifications to our own class definitions.

For all data, whatever its source - test beam, balloon, simulation, or flight instrument – our intent is that the corresponding ROOT data files have the same internal structure. Hence, I/O and low-level analysis routines can be shared. This will greatly minimize the programming effort, as the same functions will not have to be rewritten for each data source. We currently store detailed Monte Carlo truth, detector digitization, and reconstruction data in ROOT tree files. The following figure illustrates the logical tree structure for the raw detector data:

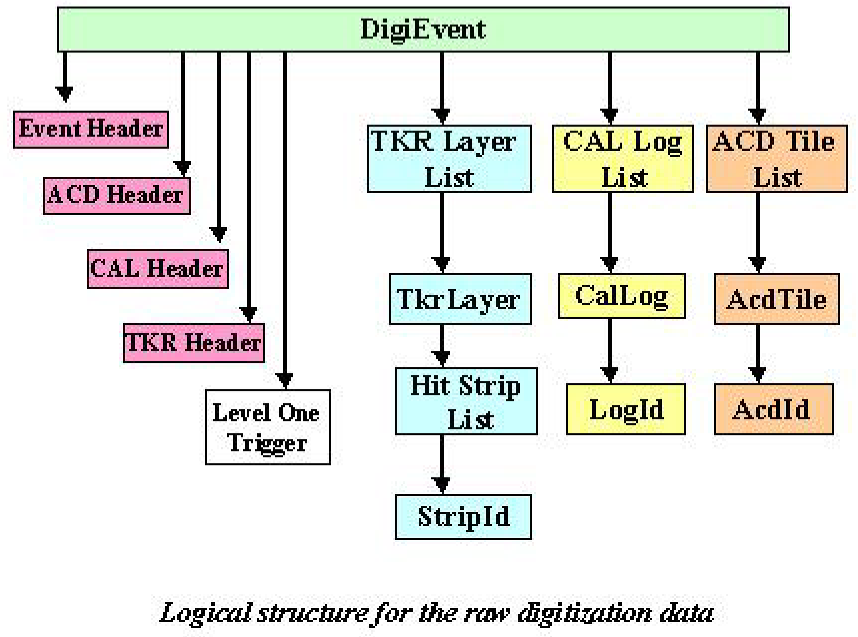


Figure 10.1.5. Logical structure for the raw digitization data.

GLAST first used ROOT during the TB99 run. Raw data files were converted to ROOT. The test beam reconstruction program output is in ROOT as well. Since the test beam, our experience and knowledge of ROOT has grown. We have taken this time to improve the internal structure of our ROOT files, now taking advantage of the many optimizations provided in the ROOT system. The 2001 GLAST balloon flight has its raw data converted to ROOT.

Since the test beam, work has been completed to expand our use of ROOT. The Monte Carlo simulations, running within the Gaudi framework, output reconstruction data into a summary ROOT ntuple and as well a full ROOT tree file. The ROOT tree contains detailed information about the results of reconstruction, including all tracks found by the tracker reconstruction. In the past, we have not had this level of detail available as output. This forced those interested in such details to introduce analysis code directly in the reconstruction algorithms.

10.1.4.3 ROOT as an Analysis Framework

In addition to I/O routines, ROOT also provides an analysis framework that runs on all major platforms. Common analysis functionality is provided including display creation, histogramming, and function fitting. ROOT provides interactive command-line C++ support through the use of CINT, a C/C++ interpreter. Thus, all of the power of ROOT is available through an interactive interface. ROOT also provides a number of graphical user interface tools, e.g. the Object Browser, which allows one to scan open files, loaded classes, etc. Fig 10.1.6 shows the ROOT Object Browser, where the contents of a summary ntuple is displayed:

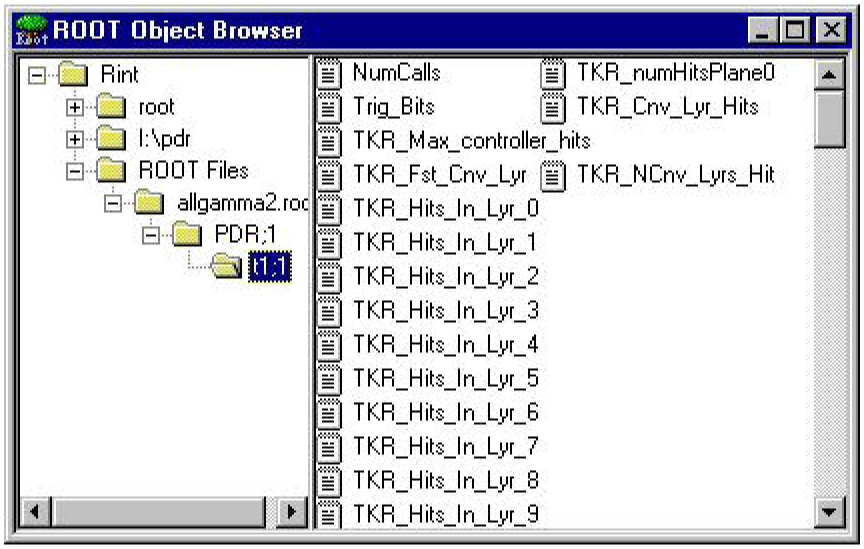


Figure 10.1.6. ROOT's Object Browser

One may browse the contents of open files, and inspect the contents merely by double-clicking on an element - producing a histogram automatically.

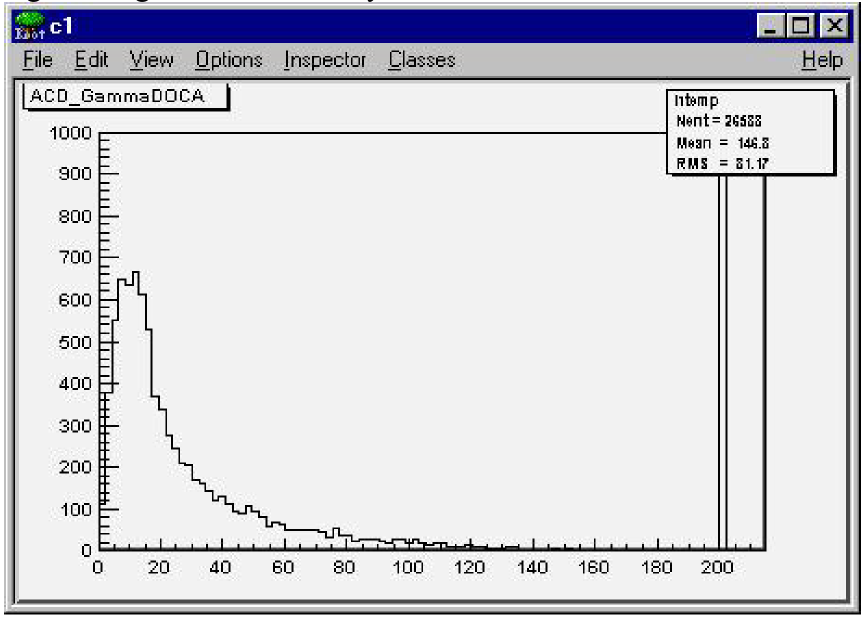


Figure 10.1.7. Histogram of ACD_GammaDOCA, one of the summary ntuple elements.

Plots can be output in Postscript or Encapsulated Postscript, at the click of a button. The drawing options for histograms are extensive, allowing users to customize their plots in any desired fashion.

While ROOT provides all of the functionality required of any analysis framework, we have had some problems. Due to our group's lack of ROOT and C++ experience, using ROOT's analysis toolkit has been a learning experience. The ROOT team, based at CERN, has worked hard to provide more documentation and examples. An extensive user's guide debuted November, 2000. Six hours of instruction are available on a two-CD set, produced by the ROOT support team at Fermi Lab. Within the GLAST software team, we have tried to meet the demand for ROOT support. A GLAST specific ROOT home page has been developed and is continuing to expand, including a Frequently Asked Questions section of common problems. The web page is located at: http://www-glast.slac.stanford.edu/software/root/howto/.

A key area where most users need assistance is in manipulating the detailed ROOT tree files. Example ROOT macros to manipulate the tree files have been provided. The macro handles the mundane tasks of opening and accessing event data in the ROOT files. One function, called Go( ), contains an event loop and all user defined analysis code. Hence, the user only needs to worry about the specific aspects of his/her own analysis. Those interested solely in summary data can use the ROOT summary ntuple to see the basic results of a particular run. The ntuples are easy to digest and manipulate, while the full tree files are provided for those who desire to take a closer look at the data.

10.1.4.4 Support for other Analysis Frameworks

There are other low level analysis frameworks available besides ROOT. It would be short-sighted to allow just one analysis framework for data analysis. In fact, there are a number of GLAST collaborators who are experienced IDL users. IDL is a commercial analysis framework, used extensively in the astrophysics community. The decision to use ROOT as our I/O mechanism should not preclude the use of alternative data analysis frameworks.

As mentioned previously, ROOT files are self-describing. Reading and manipulating ROOT files using a framework other than ROOT has been demonstrated. The Java Analysis Studio written by Tony Johnson (SLAC) can now read in ROOT files. The Java based ROOT file reader is independent of the ROOT framework. Detailed information about this Java library is available at: http://java.freehep.org/lib/freehep/doc/ROOT/index.shtml

Similarly, other ROOT file readers may be used in conjunction with other analysis frameworks, such as IDL. IDL has the capability of calling external routines in shared libraries. If this library of routines is set up appropriately, the routines will be available within IDL just as if they were regular IDL system routines. The interface for the user is seamless.

A prototype library called Root2IDL has been developed. This library provides routines to read in ROOT data into IDL directly. The original prototype was developed during the 1999-2000 SLAC beam test, before ROOT was self-describing. Hence, the library currently handles beam test ROOT files only. A generic Root2IDL which can handle any ROOT file is quite feasible and is under development.

10.1.5 Documentation

No software is complete without adequate documentation, both for developers and users. During the past year, we have made great strides in terms of documentation. Our efforts have focused on two fronts: web pages and code documentation. Ultimately, our documentation collection will be used to form our and users' and developers' guides.

The GLAST ground software home page is: http://www-glast.slac.stanford.edu/software/. A web server has been set up at SLAC, providing a central location for GLAST collaborators to create and maintain web pages. A series of web pages have been written demonstrating how to get set up to use GLAST software. Much time was devoted to detailing the specifics for both Windows and UNIX.

Web pages for each of the main software applications have been developed. These pages are designed specifically for users with no knowledge of the code. An example of one such page is available at:
http://www-glast.slac.stanford.edu/software/balloon/analysis/bfemApp/

To aid in code documentation, Doxygen was chosen as our documentation system. Doxygen extracts documentation direct from the source code files, generating either on-line HTML documents or an off-line reference manual. Doxygen provides an easy mechanism to keep up-to-date developers' guides. A variety of file formats are supported including PDF, Postscript, etc. For more information about Doxygen, please see http://www.stack.nl/~dimitri/doxygen/index.html.

Package-wide documentation is supported through the use of *mainpage* files. Each package in our CVS repository contains a *mainpage.h* file. During the course of adding Doxygen comments to the code, a specific comment can be added to the package's *mainpage* through the use of Doxygen's “\mainpage” command. Thus developers are encouraged to provide links from the mainpage to primary classes.

10.1.6 Coding Standards

Coding standards help produce code with a consistent look and feel, increasing code readability. Naming conventions aid developers and users, easing name recollection. An initial list of coding standards has been developed, using the standards of other software projects as a model. Our current collection of coding standards is available at http://www-glast.slac.stanford.edu/software/CodeHowTo/codeStandards.html

While it is useful to have our coding standards documented, this does nothing to insure adherence to those standards. We are interested in providing an automated system that would check code to see if it satisfies the coding rules we have set forth. Checking would occur as code is checked into our CVS repository.

10.1.7 UW Terminal Server and SLAC's ground distributions

Many users of GLAST software are not interested in the details of obtaining the code, compiling the binaries and then running the applications. Instead, many people would prefer to obtain the code compiled and ready to go. In fact, most users would prefer a stable location set up for them. Additionally, a number of changes have occurred during the past year in terms of compiling and running the code - specifically the introduction of CMT/VCMT. Learning how to use these new tools can be daunting to those who would prefer to immediately run the code.

The GLAST software team has set up two stable and central sites that contain up-to-date release versions of our applications. One is the University of Washington (UW) Windows Terminal Server and the other is located on AFS space on the SLAC UNIX cluster.

The UW Windows Terminal Server, as the name implies, provides applications running under Windows 2000. A freely available client runs on any flavor of Windows, providing a speedy windows interface into the server. The server is much faster than most user’s desktop computer, in fact. All tools are available globally, and the configuration is kept up to date. Naturally, VCMT is readily available, and configured properly from the start.

SLAC provides applications running on Red Hat Linux and Solaris, with access to the powerful batch system.

10.2 Code Release Management and Verification

The primary function of Code Release Management and Verification is to provide a suite of diagnostics for system wide verification and testing. The diagnostic suite shall provide the following functionality:

- perform automated system wide builds
- perform automated system and unit tests
- automatically notify (e.g. via email) a designated list of management team members of build and test results
- provide summary and diagnostic information based on test results and comparison with test results for previous releases

In addition the facility shall provide the following automatically performed functions:

- check for new release versions (tagged versions) of all major SAS packages (release packages)
- attempt to build, in all supported operating systems, each newly tagged package found
- attempt to run tests, in all supported operating systems, for each newly tagged package that builds successfully
- perform above steps for all packages that newly tagged release packages depend on
- log a summary of results of above functions to a database
- attempt a daily build, in all supported operating systems, of the development version (head) of each package
- inform package maintainers and/or designated contact of the results of above functions

The science analysis software (SAS) for LAT consists of a large number of interdependent, generally independently maintained software packages. Packages that provide major functionality (e.g. performing Monte Carlo simulations of detector response) are referred to as release packages and consist of sets of packages maintained both by various LAT collaborators and developers external to LAT.

The method of tracking changes and maintaining proper synchronization for this large set of interacting packages, sometimes referred to as software release management, is primarily determined by the version and configuration management systems that LAT has chosen to use (Concurrent Versions System – CVS Sec 10.1.1, and Configuration Management Tool – CMT Sec 10.1.2). At the lowest level, via the use of CVS, a code repository is maintained (currently containing on the order of 100 independently maintained packages) that holds the current code base and tracks all changes to a package since it was placed in the repository. CMT allows users to define and maintain dependencies between packages and various versions of packages. When it is determined that a set of packages (including their versions) that perform some function (e.g. Monte Carlo simulations of detector response) is ready for release, the set of packages is collected into a single package called a release package and "tagged" with a unique label identifying that particular release. Individual packages are also tagged when it is determined that they are ready for general use; it is in fact these tags that are used in the specification of a release package in terms of its components.

A Release Manager has been built to allow the software management team to perform automated, system wide, builds and tests to determine when software is suitable for release.

Given that LAT code developers work on a variety of platforms it is often difficult for individual developers to track the effects of their development in all supported environments in which the release packages are expected to run. An automated release management facility is useful in this context; it helps to track potential problems and inform managers and developers as the problems occur. In addition to attempting to build dependent packages and run developer designed tests for all dependent packages in a release, the Release Manager will be designed to build and run tests for development code (referred to as in the head of the repository) as well. This can provide useful diagnostics to developers before it is decided to release a package.

Finally the Release Manager will log the results of the build and test stages for release packages in a database, as well as informing developers and/or designated contacts of build and test results for both release and development packages in addition to performing the system-wide diagnostics.

A prototype of the Release Manager has been written in Perl. Perl is a reasonable choice for the Release Manager due to its well tested libraries for network support and ease of extensibility. Current functionality includes checking the repository for new tags of release packages, checking out and attempting to build newly tagged packages, and informing designated contacts, via email, of the results. The Release Manager currently only works in Unix; current plans are to design it to run the build and test steps on a Windows server as well.

10.2.1 Testing

Currently the testing facility provided by the Release Manager simply attempts to locate (using a determined name convention and location) and run, if found, a user supplied test function for each package, and inform the developer and/or designated contact(s) of the results. This should suffice for post-release testing, as package maintainers are the most qualified for determining proper diagnostics for their package. Some general tests that are to a large degree package independent shall be added to the Release Manager to implement the system-wide diagnostics. These will provide both the system-wide and unit test capability and generally fall into standard software testing categories. They include:

- regression tests - i.e. whether the current release generates different results than the previous release e.g.: comparison of various histogram statistics, chi squared tests, Kolmogorov-Smirnov tests
- conformance tests - i.e. whether the code satisfies various conventions outlined by the collaboration
- performance tests

Another reason for using Perl is the availability of the Perl facility Test.pm which provides useful extensions for software testing and would be a convenient way to implement any additional testing functionality for the Release Manager.

10.2.2 Notification and Logging

Package contacts are currently notified by email of build and test failures for release packages, with a summary describing the failure. Current plans are to log this summary to a database as well as generate more extensive summary information to email developers regarding build and test failures on various operating systems for development code.

10.3 Sources, Simulations, and Reconstruction

This section describes the physics modeling of the incident particle fluxes and the traversal of particles through GLAST, and the reconstruction and interpretation of the instrument response to those particles.

As the LAT is a pair-conversion telescope, the detection of photons is done on an individual basis: the trail of ionization in the instrument from each conversion and subsequent showering undergo pattern recognition and fitting stages. These stages serve to identify the particle, and estimate the direction, energy and quality of the reconstruction.

Simulation is required for several purposes: instrument design, algorithm development, and estimation of performance, e.g. efficiency vs. purity for photons. A model of the incident flux is needed for the performance estimates to map out the full space of energies, angles and background contaminations that the instrument is expected to encounter.

10.3.1 Sources: Incident Flux

An obvious requirement for simulation is to provide flexible sources of incident particles, corresponding to the "event generators" used in accelerator-based detector simulation. The sources must meet several needs: illumination of the entire detector or only a portion; incident angles or ranges of angles specified with respect to the detector, or with respect to the local zenith, or finally with respect the to sky. The rates of incident particles must be a property of the source. This allows composite sources to be constructed that determine the relative fractions of the components according to the total flux of each component. Parameters must include absolute time and the orbital position, for geometric transformations from local coordinates to celestial ones, and for cosmic ray sources that depend on geomagnetic latitude. Finally, the orientation of GLAST with respect to the local zenith must be taken into account: in our scanning mode, as opposed to pointing, we expect to “rock” the instrument between ±35 degrees about a N-S axis (or above and below the orbital plane), to provide uniform coverage of the sky.

Our choice of a design that facilitates implementation of the above requirements involves abstraction of the properties of a source, which can be satisfied by actual instances of the variety of sources that we want to be available. For example, for studying the response of the detector to specific particles as a function of direction and energy; we want to specify an angle, or range of angles with respect to the instrument; to understand the rates and potential contamination from cosmic ray background, we need sources that represent the particle composition, energy dependence, zenith angle dependence, and dependence on geomagnetic latitude corresponding to the observed cosmic rays. Finally, the abstract definition of a source, along with global parameters representing the instrument orientation, orbital position, orbital orientation and absolute time, must accommodate the representation of gamma rays emitted by astrophysical objects, such as AGN's and gamma-ray bursts.

While we must provide a library of sources sufficient for the intended uses of the simulation, there must be a mechanism for users to easily add new sources, either via modifications of the parameters used to create instances of existing sources or entirely new code, without the need to modify existing code. This is to be implemented with the same mechanism used for the built-in code, that is, use of the abstract definition, or interface, to represent sources, and the use of a "library" of available sources to which a user can add a new module.

The diagram shows some of the elements of this concept. The Flux Service box represents the interface to the other elements. It has access to a library of sources, from which the selected source is chosen, and to which external source descriptions can be added. It manages a description of the orbital parameters and GLAST orientation (the oval labeled Orbit), on which the selected source may depend. For flexibility and extensibility the Source library is partly implemented by entries in an XML document.

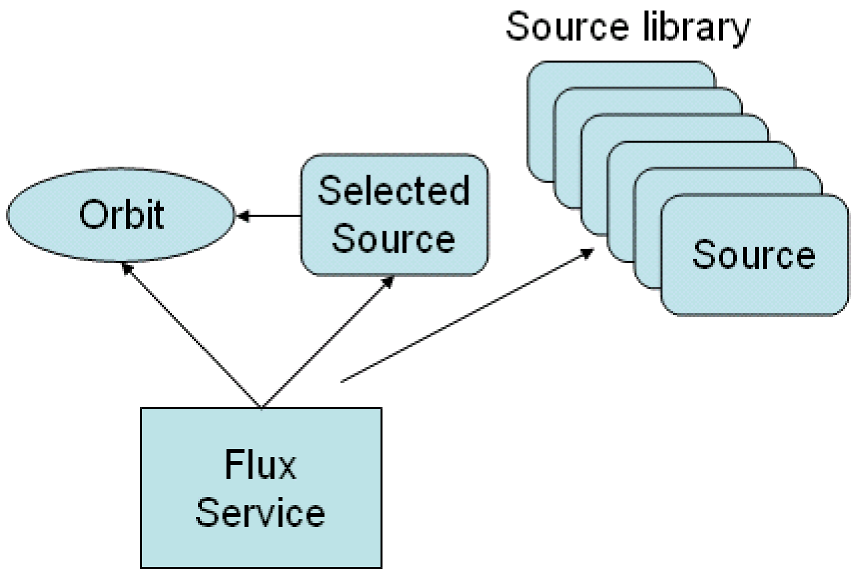


Figure 10.3.1. Elements of the source design.
The "*Flux Service*" provides an interface for use. *Orbit* maintains the parameters associated with the position and orientation of GLAST, and the *Source library* maintains a list of available sources that can be selected to generate the actual particles bombarding GLAST.

10.3.2.1 Rootplot

We plan to have a facility, called for now "rootplot", to easily make plots of the source energy spectra or angular distributions. This allows quick development of new sources, and verification of rates. The following is an example showing the components of a proposed background mixture:

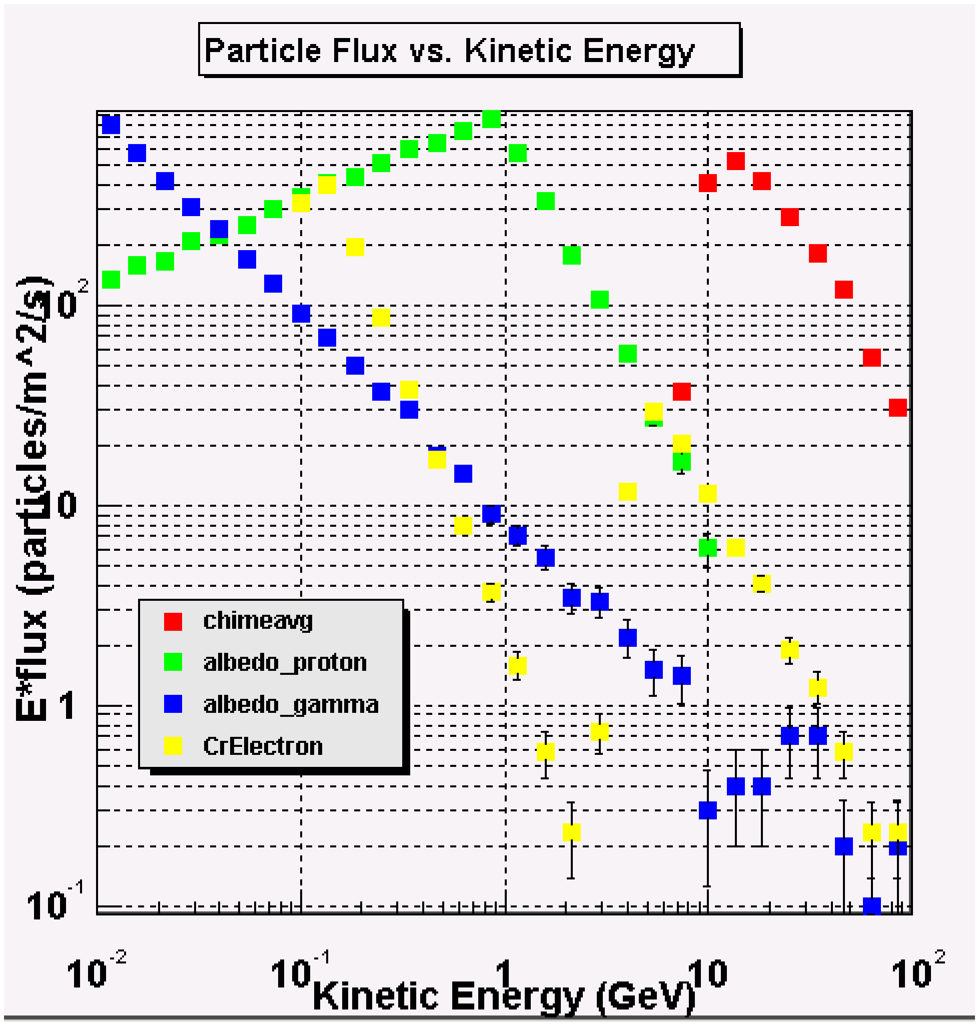


Figure 10.3.2.1 Plot of the energy spectra for various components of a proposed background mixture, including: (1)chimeavg, representing a average rate for the CHIME model of primary proton cosmic rays; (2) albedo_proton, the spectrum of albedo and reentrant protons corresponding to recent measurements; (3) albedo_gamma, secondary gammas from the horizon, and (4) CrElectron, a mixture of primary and secondary electrons and positrons. The abscissa is the kinetic energy of the particles (gamma, proton, or electron) in GeV, and the ordinate the flux times energy integrated over angles, in particles/(m^2 s).

10.3.3 Event Simulation

Given an incident particle, the task of the simulation is in principle simple to state: transport the particle, using Monte Carlo techniques of sampling from a variety of distributions (ionization losses, interaction and decay probabilities, interaction daughter products, etc), through the instrument and record the effects of the interactions on the sensitive detectors of the instrument. These effects are generally the ionization losses of charged particles in the detector elements. Once these losses are tallied per detector element, there is a digitization phase in ehich the losses are converted into expected digital outputs (hit strips, ADC values in the CAL and ACD). These digitizations are then formatted to look identical to the real data, so that the reconstruction process can be blind as to whether the data it is working on is real or from simulations.

It is vital to have a 3D graphical representation of simulated events, so that one can understand the geometry, correlate particles with the responses and reconstruction, both to understand the processes involved, and to search for errors. Our prototype system uses a simple but effective object-oriented

3-d graphics interface with a GUI. For the longer term, we will use either ROOT or a Java-based system called WIRED. Both have better support and active user communities. An example picture follows:

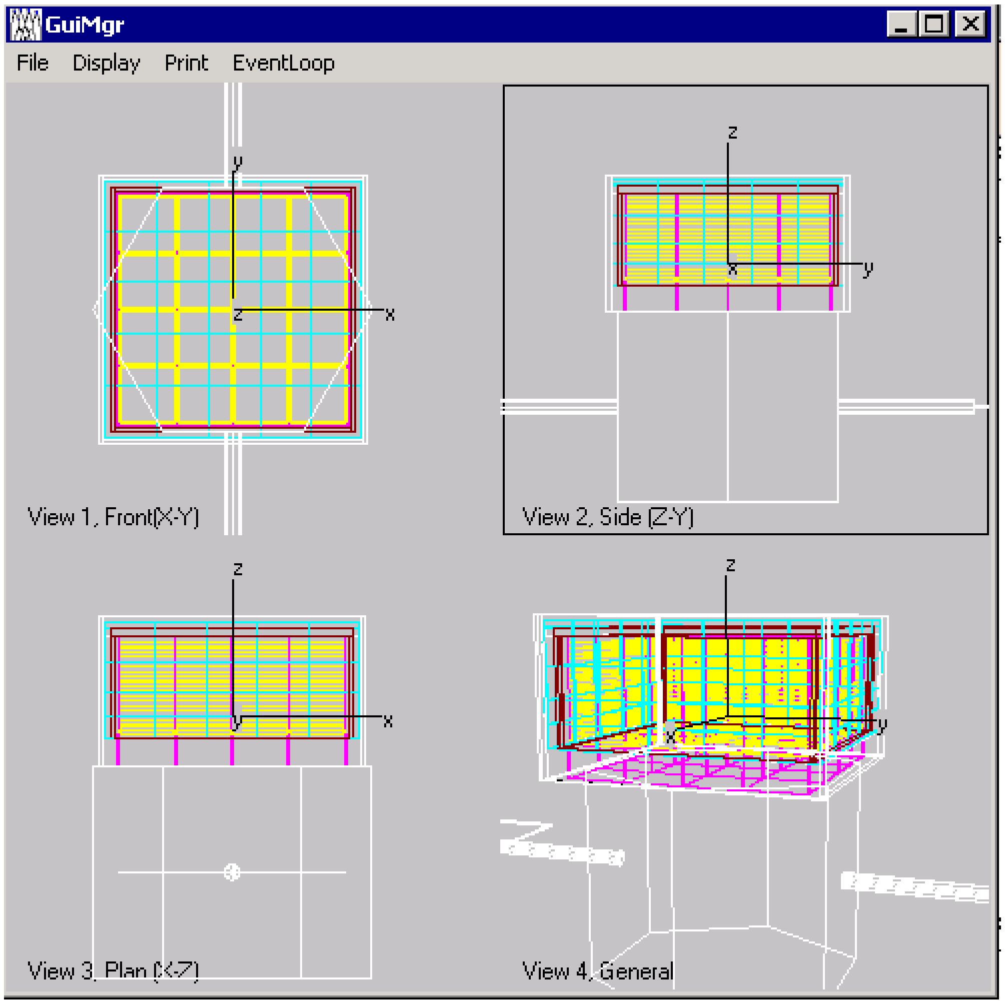


Figure 10.3.2.1. The full GUI display, with pull-down menus for control of the job, and four projections of the instrument, showing the outer shield (white outline), ACD tiles (blue rectangles), TKR system (yellow), CAL (below the TKR). The Spacecraft is represented by a hexagonal solid. The solar panels are shown as well.

The display also provides indications of the detector response, given that the geometry, particle trajectories, and instrument responses can all be overlaid selectively on the same display. The following plot, Fig. 10.3.2.2, shows the simulation of a 3 GeV gamma ray shower.

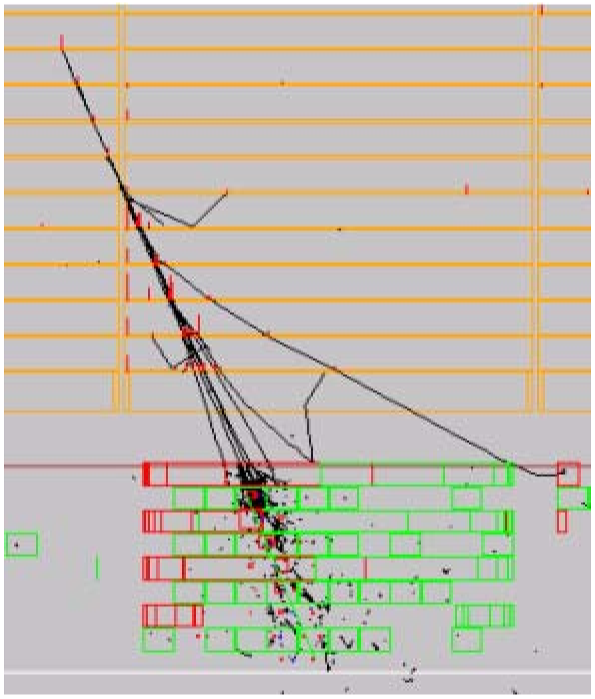


Figure 10.3.2.2. A projection of the 3D display, of a generated gamma-ray event, showing outlines of the TKR trays, charged particles (black lines) and responses of detector elements. The black lines are the charged particles (electrons and positrons). The red ticks in the upper tracking section represent silicon strips that have had enough energy deposited to register as "hits". These are drawn at the end of the respective strips, and so line up with the track in one view. The hits in the perpendicular view can be seen along the edge of the tower. Energy deposited in the CsI logs is used to determine the light output seen at each end of the crystal: these are drawn as red or green boxes.

There are four major components involved with the simulation: geometry description; particle transport; bookkeeping of energy depositions in the instrument; and conversion of those depositions into detector response. These are described in the subsections below.

10.3.4 Geometry

The simulation requires the most information about the geometry of all its clients. It must be able to handle a complex physical setup, forming a hierarchy of many shapes and materials. The goal of the design was to provide sufficient flexibility to describe the breadth of devices likely to be used (all engineering models plus the flight instrument), and to give equal access to all clients. The implementation made use of XML to provide both a human-readable specification and one rich enough to describe a hierarchical geometry.

A number of advantages of the design follow:

- volume description mechanism was borrowed heavily from ATLAS, which in turn was designed to map easily to the GEANT/simulator point of view
- Evaluation/substitution mechanism makes data files more maintainable and facilitates automatic documentation as shown at http://www.slac.stanford.edu/~jrb/glast/xml/constsDoc.html

- abstraction of geometry information into objects of C++ classes, collectively known as detModel, which insulates applications from the XML format. A variety of clients have already been (quickly!) written using detModel. GEANT4, for example, can deduce the geometry from the XML LAT description, with no need for hard-wired code, as shown in Fig 10.3.4.1.
- tools for assigning identifiers to volumes via XML constructs finish the job of keeping the description, particularly those parts of it which are of interest to more than one application, out of code.

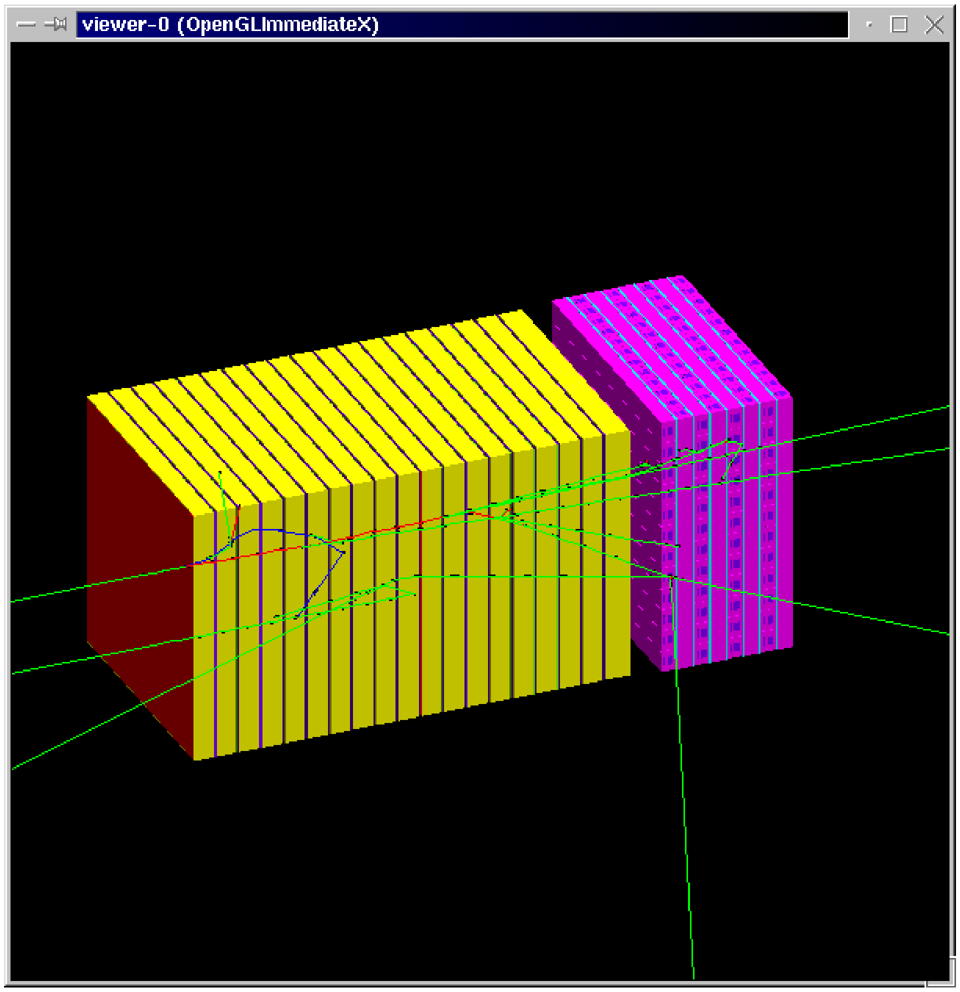


Figure 10.3.4.1 GEANT4-simulated particle interaction.

10.3.5. Particle Transport

This area has seen tremendous effort in the past twenty years, with the trail blazed by the EGS and GEANT projects. Our current Gismo, and shortly GEANT4, step particles through materials, accounting for interactions, decays, and geometric boundaries to limit the step size. When a step is taken, appropriate energy losses and multiple scattering are applied. If an interaction or decay is deemed to have occurred, the particle is terminated and one or more new ones started at the interaction/decay vertex.

10.3.6 Bookkeeping of energy depositions in the instrument

We wish, as an option, to track all energy lost anywhere, in order to tune reconstruction algorithms with knowledge of the “truth”. This means that energy lost in insensitive materials must be recorded. In addition, the energy deposits must be traceable to their primary parents, bywhich we mean the e^+ or e^- in a photon conversion or to the original particle otherwise.

10.3.6.1. CAL

- the full energy accounting must be segmented; it cannot just be one number for all energy seen in inactive media.
- a calorimeter crystal, or “log” will be treated as a single volume for the simulation, but energy deposition will be segmented to allow the light collection digitization stage to deal with the distribution of energy throughout the log. In addition, it is planned to register energy sum and energy-weighted longitudinal position moments.

10.3.6.2 TKR

- the dead material energy loss must be segmented at least by plane
- energy loss in the Silicon must record the volume ID, position of the step, direction cosines and MC parent particle. It may need to record the exact particle type that made the deposit, and the total energy of that particle.
- For each hoit, must identity the particle, primary or a daughter, must be recorded.

10.3.6.3. ACD

- TBD how such energy loss must be segmented
- energy loss in the scintillator must record the volume ID, position of the step and MC parent particle.

To respond to these requirements, we propose to notionally separate the instrument into 'volume integrating' and 'single-step' components. In any case, all volumes are marked as 'sensitive' to the simulation package (e.g., Gismo, G4). Volumes will carry an additional property indicating whether they are 'sensitive' for digitizations.

10.3.6.4 Single-step

All steps are recorded in all volumes; energy lost, position vector, volume name and MC parent are recorded per hit. The hits are recorded up to and including the termination point or exit from the world volume.

10.3.6.5 Volume integrating

All steps until each particle interacts are stored, just as for the single-step volumes. Once a particle interacts, it is tagged as showering and all its daughters' contributions are assigned to it.

10.3.7 Digitization

The hit information is sufficient to simulate the instrument digitizations, in which the simulated energy deposit is gathered up per sensitive element and transformed into a readout. For the TKR, this is a list of hit strips and TOT per layer; for the CAL and ACD, these are pulse heights from the various photo-diodes and photo-multipliers respectively.

Random noise is added in all subsystems (which can add or subtract from the counts above threshold). Charge is shared amongst hit strips in a geometric fashion. Studies are underway for a more realistic handling of the sharing and of modeling of the TOT.

Depending on the readout mode, the best or all four PIN diode readouts are simulated for each log end of the CAL and include the light output taper from end to end of the logs. Future upgrades will include electronic non-linearities and optical gains.

10.3.8 Event Reconstruction

The event reconstruction takes the raw readouts from the detector elements, converts them to physics units (e.g. energies in MeV, distances in mm), performs pattern recognition and fitting to find tracks and then photons in the tracker; finds energy clusters in the calorimeter and characterizes their energies and directions and uses the ACD to veto events in which a tile fired in the vicinity of a track extrapolation.

10.3.8.1 Tracker

The Tracker reconstruction is initially done in separate *x* and *y* projections. The projections are associated with each other whenever possible by matching tracks that pass from one Tracker module to another. This significantly improves the power of the reconstruction for complex events.

The converter foils, needed to produce the interactions, introduce an unavoidable error due to the multiple Coulomb scattering (MS) in the trajectory of the particles. It is crucial to understand how the multiple scattering affects the reconstruction of the particle trajectories.

The presence of non-negligible multiple scattering complicates the fitting procedure and the pattern recognition problem. The covariance matrix becomes non-diagonal in order to take into account the error correlation between different planes; thus it becomes larger and requires more computing time to invert it. The Kalman Filter (KF) technique alleviates both problems elegantly.

The power of the KF to handle the track fitting problem when multiple scattering errors are involved comes from its iterative property. KF considers only one measurement each time, introducing it independently into the fit. This property facilitates the decision of adding or removing a given measurement to the track, therefore aiding the track finding. It also permits the introduction of random errors (as it is the case of the multiple scattering) in a natural way. Now, one has to consider only the multiple scattering error produced between two measurement planes. This simplifies greatly the problem of the MS error.

The KF also allows us to compute the precision or resolution on the track parameters at the vertex position, since it provides the parameters and their covariance matrix of the track at each measurement location and, most importantly, at the first plane. Extrapolating this covariance matrix to the interaction vertex, one obtains the resolution of the track parameters. With this technique one can quantify the multiple scattering effect, and the relation with the other detector parameters.

The track model is a straight line and the measurements are a set of periodic hit positions. The distance between planes, the resolution and the amount of MS per plane, are known for a given energy. This makes the application of the KF simple and straightforward.

The figure shows the result of the KF algorithm applied to a simulated high energy muon.

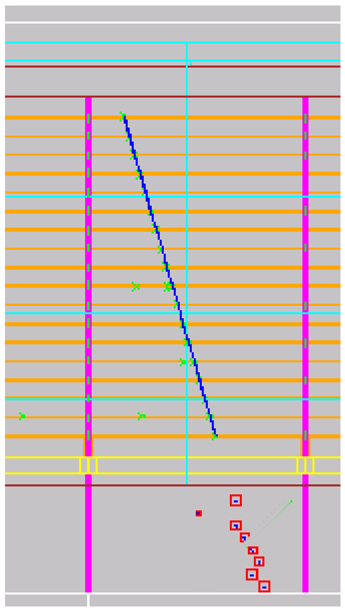


Figure 10.3.7.1 KF fit to a simulated high energy muon (blue line).
Reconstructed energy centers in the calorimeter (red boxes) are also shown.

An important result of the track fit is an estimate of the direction of the incoming particle. We use the MC to estimate the resolving power, or point spread function (PSF), of the instrument, by comparing the reconstructed direction with the direction of the simulated incoming gamma ray. There are two factors that limit the precision possible: intrinsic measurement resolution, basically the strip pitch, and multiple scattering due to the passage of the converted electron and positron particles in the converter foils. Experience has shown that the performance is close to the theoretical limits imposed by these effects, except for misidentifications that contribute to “tails” of the distribution. An example plot of this quantity follows:

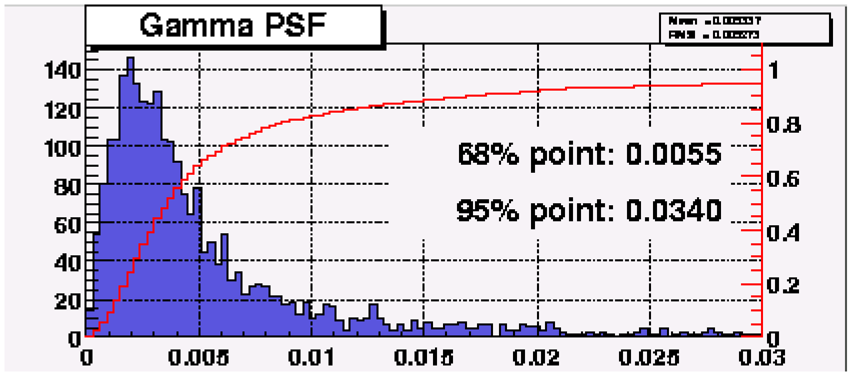


Figure 10.3.7.2. Histogram of the angle,
in radians,between the incoming and reconstructed photon direction. The red curve is an integral used to identify the 68% and 95% angles..

The tail on this particular distribution is in fact quite broad, indicating a need to impose tighter restrictions on sources of misidentified tracks.

10.3.8.2 Calorimeter

The calorimeter consists of 16 modules of 8 layers of 12 CsI(Tl) crystals in a hodoscopic arrangement, this is to say alternatively oriented in X and Y directions, to provide an image of the shower. It is designed to measure energies from 30 MeV to 300 GeV and even up to 1 TeV.

However, the calorimeter is only 8.5 X_0 thick and therefore cannot provide good shower containment for high energy events, though these events are very precious for several astrophysics topics. Indeed, the mean fraction of the shower contained can be as low as 30% at 300 GeV, normal incidence. In this case, the energy observed becomes very different from the incident energy, the shower development fluctuations become larger, and the resolution decreases quickly.

Two solutions have been pursued so far to correct for the shower leakage. The first is to fit a mean shower profile to the observed longitudinal profile. The profile fitting method proves to be an efficient way to correct for shower leakage, specially at low incidence angles when the shower maximum is not contained. The resolution is 18% for on axis 1 TeV photons, which is a 50 % improvement compared to the raw sum of the energies recorded in the crystals.

The second method uses the correlation between the escaping energy and the energy deposited in the last layer of the calorimeter. The last layer carries the most important information concerning the leaking energy: the total number of particles escaping through the back should be nearly proportional to the energy deposited in the last layer. The measured signal in that layer can therefore be modified to account for the leaking energy.

The methods presented significantly improve the resolution. Up to 1 TeV, the resolution on axis is better than 20 %, and for large incident angles (more than 60 degrees) it is less than 4 %. It should be noted that the last layer correction is more robust since it doesn't rely on a fit, but its validity is limited to relatively well contained showers, making it difficult to use at more than 70 GeV for low incidence events. There is still some room for improvements, especially by correcting for losses between the different calorimeter modules and through the sides.

10.3.1 Background Rejection

Background rejection performs the function of particle identification, defining whether the incoming particle was a photon or not. With a 10^5:1 charged particle background to signal ratio, shower fluctuations in background interactions can mimic photon showers in non-negligible numbers. Cuts are applied to the events to suppress the background. Studies done to date use the following properties of the instrument and particle interactions to achieve the suppression:

- use fitted tracks to extrapolate to the ACD and CAL. Photons will not fire tiles in the vicinity of the extrapolation. There should be energy depositions in the calorimeter near where the track extrapolates to it.
- hit patterns: photon and hadron showers yield different topologies of hits in the TKR and CAL.
- track quality: ensure that reconstructed events in the tracker are of sufficient quality
- spacecraft induced events: suppress particles entering the instrument via the bottom (ie coming in through the calorimeter).

The cuts will continue to be honed as reconstruction algorithms and understanding of the instrument and background improve.

10.4 Science Tools

The design for the high-level analysis software, i.e., the software for analysis of gamma-ray data after reconstruction and background filtering of events, is driven by three principal considerations:

1. High-level analysis of GLAST data will be fundamentally statistical; limited numbers of photons and relatively poor angular resolution mean that quantitative analysis will be via model fitting.

2. The characterization of the LAT is complicated and compounded by a scanning observation mode, a large FOV, and the need to reject the intense background of cosmic rays as well as albedo gamma rays from the earth's limb. The PSF, effective areas, and energy resolution depend on energy and direction of the gamma ray, location of the conversion in the LAT, and on the background rejection cuts employed.

3. After events are reconstructed, data access will be principally by direction on the sky and time range. (For cosmic rays used in monitoring calibration, access will be principally by direction in instrument coordinates and time range; see Calibration section.) The data analysis system must support efficient spatial access.

10.4.1 Interstellar emission model

Consideration (1) implies that a model of the direction and energy dependence of the interstellar emission of the Milky Way is needed for the high-level analysis. This emission is from cosmic-ray collisions with interstellar gas and photons and it is present in any direction on the sky although most intense from directions close to the plane of the Galaxy. More than 60% of the celestial gamma rays detected by EGRET were interstellar emission. The emission can have structure on fine angular scales, and an accurate model will be useful for distinguishing low-latitude point sources from unresolved diffuse emission and for accurately determining positions and fluxes of point sources. (In addition, an accurate model of interstellar emission at high latitudes will be essential for estimating the truly diffuse extragalactic emission.)

The development of models of interstellar emission is fairly well understood after more than two decades of application in high-energy gamma-ray astronomy. Advances that we will take advantage of for the LAT include higher-resolution surveys of the interstellar medium than were available for EGRET analysis, and modern calculations of cosmic-ray production and propagation that include constraints from cosmic-ray isotope abundance ratios and other local measurements.

10.4.2 Likelihood analysis

Also regarding (1), model fitting in high-energy gamma-ray astronomy has long used the likelihood function as the measure of goodness of fit (e.g., Pollock et al. 1981). Variations of the likelihood function (which defines the likelihood of the data given the model) with respect to the various parameters of the model can be used to quantitatively determine confidence ranges.

For EGRET and preceding missions, the likelihood analysis was based on binned maps of photons, i.e., by comparing the predicted and observed numbers of photons in bins of energy and region of the sky. Information is lost in binning, and in principle the most sensitive analyses can be performed using unbinned implementations of the likelihood function, where the contribution to the likelihood function of each photon is treated individually, using the response functions that apply to that photon. Unbinned analysis is much more computationally intensive, and is less well-behaved numerically, with results often being the small difference between two large numbers.

Regarding consideration (2), for GLAST, we intend to perform a trade study on the degree of binning acceptable (or maybe to implement both analysis options). Regarding binned likelihood analysis, fine binning in energy and inclination angle are likely most important to limiting the loss of information.

10.4.3 Exposure calculation

The calculation of instrumental exposure is fundamental to obtaining calibrated fluxes and spectra. The exposure is a function of time range, energy and direction on the sky. It also depends on the spacecraft position and orientation, because directions near and below the earth's limb must be excluded. Exposure calculations, complicated as they are, must be performed rapidly in order to support the multiple all-sky analyses that will be undertaken daily. Our implementation of the LAT analysis software includes an optimized algorithm that can quickly and accurately generate exposure matrices by factoring the problem. Much of the calculation is in accumulating livetimes, and is independent of instrumental response functions. These accumulations can be made quickly on a predefined (sufficiently fine) grid on the sky.

10.4.4 Technical assumptions

The volume of Level 1 data will be too great (1–2 Tbyte/yr), and searching the data too computationally intensive, for the entire dataset to be distributed to each LAT investigator or guest investigator. The Level 1 and associated databases for high-level analysis (see below) will be accessed via server computers at a few sites. These sites are envisioned to be the LAT IOC, remote analysis sites of coinvestigator institutions, and the GLAST SSC. High-level analysis modules will be run on client computers, not necessarily co-located with the servers, that query the servers for data. This division obviates the need to distribute the whole LAT data set as part of the analysis environment, spreads the overall computational load for analysis, and enables a single analysis environment to be supported across the collaboration and within the SSC. (The LAT team is required to produce an analysis environment that can be used by outside investigators supported by the SSC.)

10.4.5 Data Flow

The flow of data from Level 0 through the highest levels of processing is diagrammed in Figures 10.4.1 and 10.4.2. The databases and processing steps for Level 1, i.e., the Event database and higher processing are described in the subsections below.

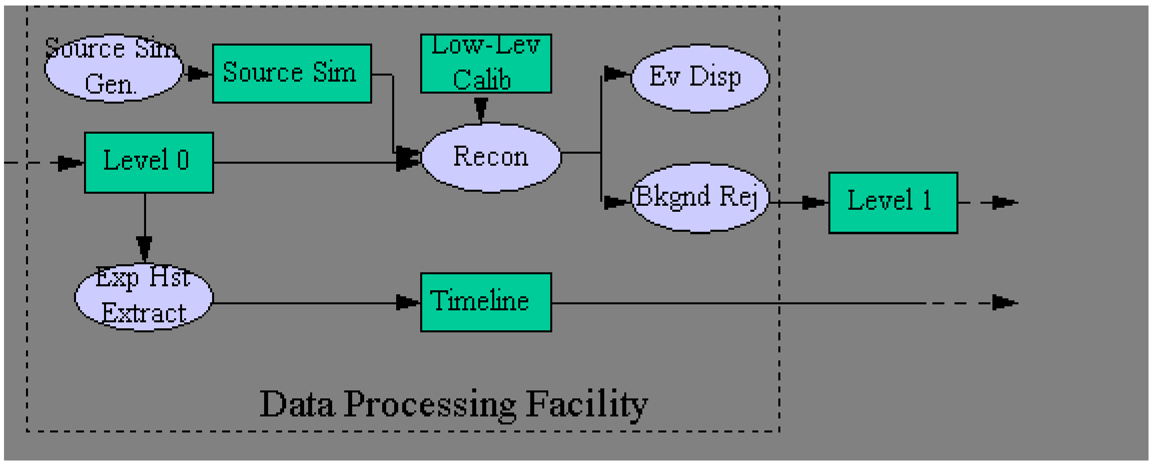


Figure 10.4.1 Level 0 to Level 1 processing

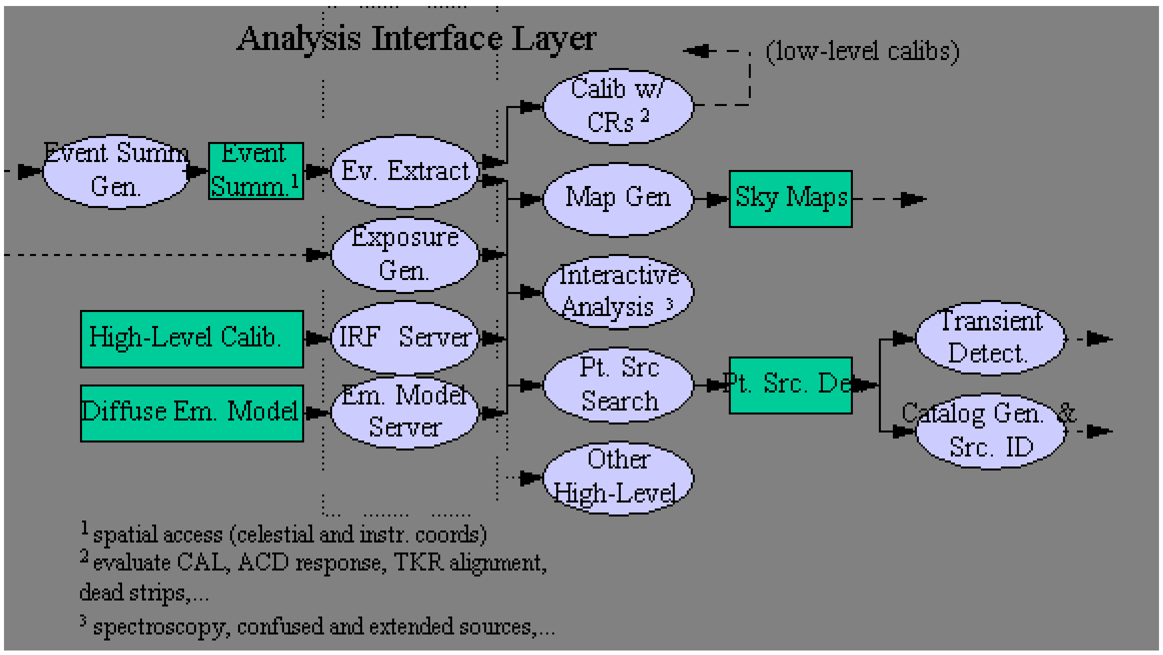


Figure 10.4.2 Post Level 1 processing

The analysis interface layer outlined in Figure 10.4.2 extracts data, calibration, and emission model information from the databases and passes it to the higher-level analysis modules. The layer is necessary to prevent the higher processing routines from being tied directly to the extraction routines used by the TBR database technology. Hence the higher level processing algorithms interface directly to the a static layer whose interface will remain the same. If we later decide to migrate to a different database technology, this will have no direct impact on the higher level processing.

10.4.5.1 Estimated file sizes for a typical analysis

As an example, consider analyzing a year's worth of data for a point source. Upon receipt of a request for the data for a region of the sky around the source (for a set of background/PSF cuts, energy range and zenith angle cuts), the Event Extractor would retrieve high-level information for the photons. The high-level information passed back to the client would have the energy, direction, inclination, azimuth, plane of conversion, quality flags, time for each photon, about 40 bytes minimum for each photon, and approximately 1 million photons (for a 10° radius selection region). The corresponding exposure matrix produced by the Exposure Generator would have exposure tabulated for a grid of energy, direction, inclination, azimuth, plane of conversion. This could be fairly large, approx 1000 (ra,dec) x 10 (inclination) x 10 (azimuth) x 10 (energy) x 18 (TKR planes) = 18 million entries. So 1 Gbyte or so would have to pass from the server to the client before analysis began. The appropriate instrument response functions for the time range and event classes selected would be generated by the IRF server. (TBR. The IRF server would have response functions for a predefined set of background rejection/PSF enhancement cuts; new cuts would require new response functions to be generated from the calibration Monte Carlo events.) The interstellar emission model for the corresponding region of the sky would probably have to be requested from the Emission Model Server as well (specifying, e.g., the coordinate system and binning), but this would be much smaller. In addition, the point-source catalog should probably also be queried to assist in defining the overall (background + sources) emission model for the region surrounding the source under study.

10.4.6 Analysis Environment

The high-level likelihood analysis of LAT data will have interactive (graphical user interface) and batch (command line or script driven) modes. Much of the LAT team's routine analysis of the gamma-ray data will not be interactive. For example, all-sky searches for point sources (to flag sources that are flaring) will be made for short time scales (typically hours), and so will be run many times per day.

10.4.6.1 Infrastructure

The infrastructure of the high-level science analysis system includes the Analysis Interface Layer described above (see Fig. 10.4.2) and the software and databases needed to provide the low level services of the Analysis Interface Layer. In particular, exposure calculation, event summary generation, high-level calibration database, and the diffuse emission model are part of the infrastructure. These modules and services are the core of the high-level analysis system.

Not explicitly discussed elsewhere, but essential to the high-level analysis system is a tool for map generation and for displaying images and plots. Maps can be generated, e.g., from a list of photons or from an exposure matrix. Images can be displayed with full coordinate information, with reprojections if necessary, and overlays.

10.4.6.2 Data Export

All processing steps that produce image or tabular output will have the capability to write the output in TBD formats to facilitate subsequent display or processing outside of the LAT SAS system.

Database	**Contents**	**Access Criteria**	**Used by**
Event	full info. for each event, including reconstruction (Level 1 database)	time or event number	Event Summ. constructor, event display, low-level calib monitoring
Event summary	energy, direction (celestial and instr. coords), time, plane/tower/log layer of conversion, event id and bkgnd rej/quality flags	energy, direction, time range, event flags, event ID	high-level map generation and analysis, CR event selection
High-level calib	instrument response functions as functions of energy, angles, plane, time,...	energy, angles, time, ...	Exposure gen, high-level analysis
Exposure history (timeline)	S/C position, orientation, LAT mode, and livetime for regular ~30s time intervals	time range	Exposure gen.
Source sim.	Monte Carlo equivalent of Level 0 data, perhaps already as 'digis', with truth info, and run/config. ID	?	Recon
Pt. Src. Detection	Position, flux, spectral hardness and associated uncertainties, time range	coordinates, time range	Transient Src search, Pt. Src. Catalog Gen.
Pt. Src. Catalog	Summary of Pt. Src. Detection, flux histories and candidate source IDs	coordinates, spectral hardness, variability index,....	Catalog access interface?
Pulsar Ephem	(radio) Timing parameters for known pulsars, contemp. With GLAST mission	pulsar name	Barycenter corrector
GRB	?	?	?

Table 10.4.1 - High-level databases.

10.4.7 High-level analysis tasks

The high-level analysis tasks planned for development are described in Table 10.4.2. Most of them derive their inputs from the Analysis Interface Layer, i.e., all of the inputs that they require are in the Level 1 and associated data (see Fig. 10.4.2 and Table 10.4.1). Other tasks require Level 2 data, i.e., the output of another high-level task. Some of the tasks are related to ancillary science goals for the LAT and will be developed as level of effort undertakings.

Name	Function	Inputs	Outputs
Point-source detection	Analysis of a given region of the sky for point sources	Analysis interface layer	locations, fluxes, significances, spectrum or spectral hardness);
Point-source spectroscopy	Model fitting with flexible definition of spectral models; possibly developed as part of the general likelihood analysis capability described below (Extended sources and confused regions)	Analysis interface layer	Model coeffs and uncertainties
Source variability	Flare detection (short term, for issuing alerts), pt. source vs. extended source determination (longer term, for quantifying variability)	Point source detection database	Flux histories, estimates of variability
Extended sources and confused regions	'Custom' model fitting. Interactive analysis largely will be model fitting (parametric), allowing flexible specification of source – multiple point sources, spectral models, arbitrary extended sources	Analysis interface layer	Model parameters, confidence ranges
GRB time profiles	Construction of time profiles for user-defined event selection criteria	Analysis interface layer (Event Summary)	Time profile histograms (perhaps normalized by IRFs, with periods outside FOV indicated), tables of events associated with a burst
Source identification	Quantitative definition off probabilities of associations of LAT pt. srcs. with srcs. in other astronomical catalogs	Point source catalog	Point source catalog
Pulsar phase calculation	Assignment of pulsar phases to a set of photons based on the timing params for the pulsar, to allow phase-resolved analysis for most of the analysis tasks, like spectral meas., and phase binning - for histograms and	Analysis interface, Pulsar Ephemerides	Phase assignments by event number (?)

	maps.		
Pulsar periodicity searches	Searches for pulsations in data for a point source	Analysis interface*	Ideally, position, period, period derivative,...
High-resolution spectroscopy	Finding narrow-line emission at high energies	Analysis interface	Line energy, flux, or upper limits
Inflight calibration	Monitoring effective area via fluxes of pulsars, monitoring PSF via phase-selected photon distributions around pulsars.**	Analysis interface	Flux histories, PSF profile plots, tables

* Also may need a tool to display times when target was in FOV to select intervals with greatest continuous coverage.

** Gains, alignments, hot/dead strips, etc., are part of the lower-level calibration monitoring described in the Calibration section)

Table 10.4.2 - High-level science analysis tasks.

Other potential analysis tasks (potentially level of effort):

• Multiple-gamma events - this may be a lower-level analysis issue - after reconstruction need to define a flag or a set of flags that indicate multiple pairs of tracks may be present. What would be most interesting is multiple pairs of tracks with the same apparent arrival direction. [What would be the approximate rate of multiple gamma events of any kind - just from closely-spaced arrival times of otherwise unrelated photons? 2.5 Hz avg rate, 20 μs separation?]

• Nonparametric algorithms for detection of point sources and extended sources without models (either for point sources or interstellar emission). This includes wavelet analysis - application for quick detection of transients.

• Polarization of point sources - the measurement will be hard (possible?), need to measure the plane of the e+/e- pair

10.4.8 Interstellar emission model

The interstellar emission model will be refined, most likely iteratively, based on LAT observations during the sky survey. The models for cosmic-ray production and propagation in particular are most constrained by the gamma-ray observations themselves. Some aspects of the EGRET findings, in particular the 'GeV excess,' need to be verified and investigated in more detail with LAT data. Also, in special directions, the 3-dimensional distribution of interstellar gas is especially difficult to determine from spectral line surveys of H I and CO, and models for different distributions consistent with the radio/mm observations may have to be tested against LAT data.

No particular tool has been identified for validating and refining the model. The most useful input would likely be a point-source subtracted map of the sky.

For LAT data analysis, the model will be precomputed for a grid of directions and energies on a grid finer than the angular and energy resolution. There's no particular advantage to generating the model on the fly for arbitrary directions and energies. The nature of the calculation (line of sight integration of the products of cosmic-ray and interstellar gas or photon densities) makes precomputing the maps straightforward and efficient.

10.4.9 Observation simulators

Two are needed: low-level (generates events that are passed through Recon and Bkgnd Rej) and high-level (based on instrument response functions and the exposure calculator). The former will be important for developing and testing the SAS system (mock data challenges) and the latter will be a proposal preparation and observation planning tool.

10.4.10 Other considerations

The high-level analysis software for the LAT is to be validated using Monte Carlo simulations of observations. Also useful for validation, and for scientific analysis, would be the EGRET data imported into the LAT analysis environment. The mapping of the EGRET summary database files into the approximate LAT equivalent of the event summary database would be straightforward. Translation of the timeline files into the LAT equivalent for calculating exposures would not be quite as straightforward, but could be done. The complication is that the trigger modes (and hence the effective area matrices) were changed (to limit the number of triggers from earth albedo gamma rays) as the earth entered and left EGRET's field of view during every orbit.

Low-level processing (event reconstruction and initial identification) is to be done at the LAT IOC, but all data, Level 0 and higher, are to be provided to the SSC. In our proposal to NASA, this was to be done via database mirroring. The SSC and LAT teams agree that this is desirable and a workable implementation is being sought. Such a system would also permit establishment of internal LAT-team mirror sites. The database system will have to be implemented in some way to protect proprietary data rights. Although the LAT team will monitor the data for transient sources and to maintain calibration, access for other purposes must be restricted during the 3-month (TBR) validation period that guest observers (and LAT team members with winning proposals) will have for their data.

10.5 Data Processing Facility

This facility has five major functions:

- automatically process Level 0 data through reconstruction (Level 1)
- provide near real-time feedback to IOC
- facilitate the verification and generation of new calibration constants
- produce bulk Monte Carlo simulations
- backup all data that passes through

First, it is instructive to examine the scale of the processing problem. The downlink rate of 300 kb/s, results in a daily rate of some 3 GB of data, or approximately 1 TB per year. Products generated from the raw data will perhaps double or triple this volume. Over a 5 year period, this comes to some 15-30 TB, reasonably modest and fairly easily all held on disk.

The average event rate in the telemetry is expected to be 30 Hz, split between signal photons and background cosmic rays. Our current reconstruction algorithm consumes about 0.2 s per event on a 850 MHz Linux processor. Assuming processors at 4 GHz by launch time (a conservative estimate), then this rate drops to 0.04 s/event, allowing a single processor to keep up with incoming data on a daily basis. To turn a full day's downlink around within 6 hours, we would require perhaps 3-5 processors. The gist of the message is that disk and CPU time are not drivers for the LAT's Level 1 analysis.

These disk and CPU needs represent perhaps 1% of SLAC's Computing Center capacity, so even a gross under-estimation of rates and volumes is easily accommodated within the existing facility. Figs 10.5.1 and 10.5.2 show estimates based on current simulations and reconstruction CU times and filesizes.

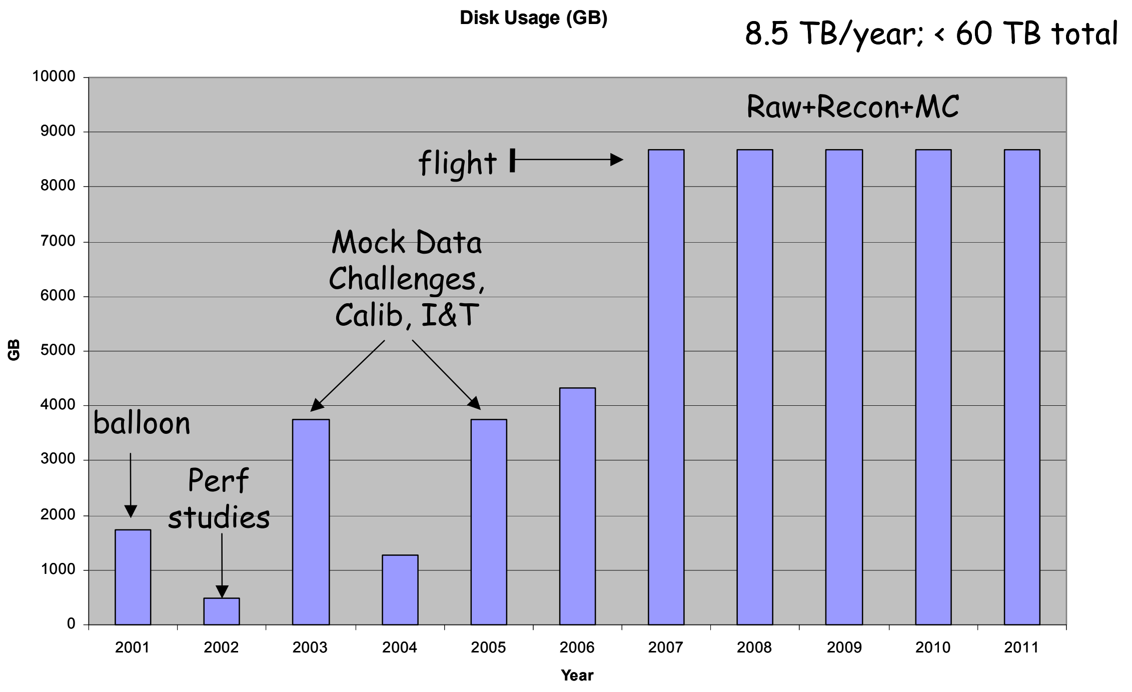


Figure 10.5.1 Projected disk needs.
The projected total through 5 years of flight comes to less than 50 TB, and includes MC simulations as well as instrument data processing.

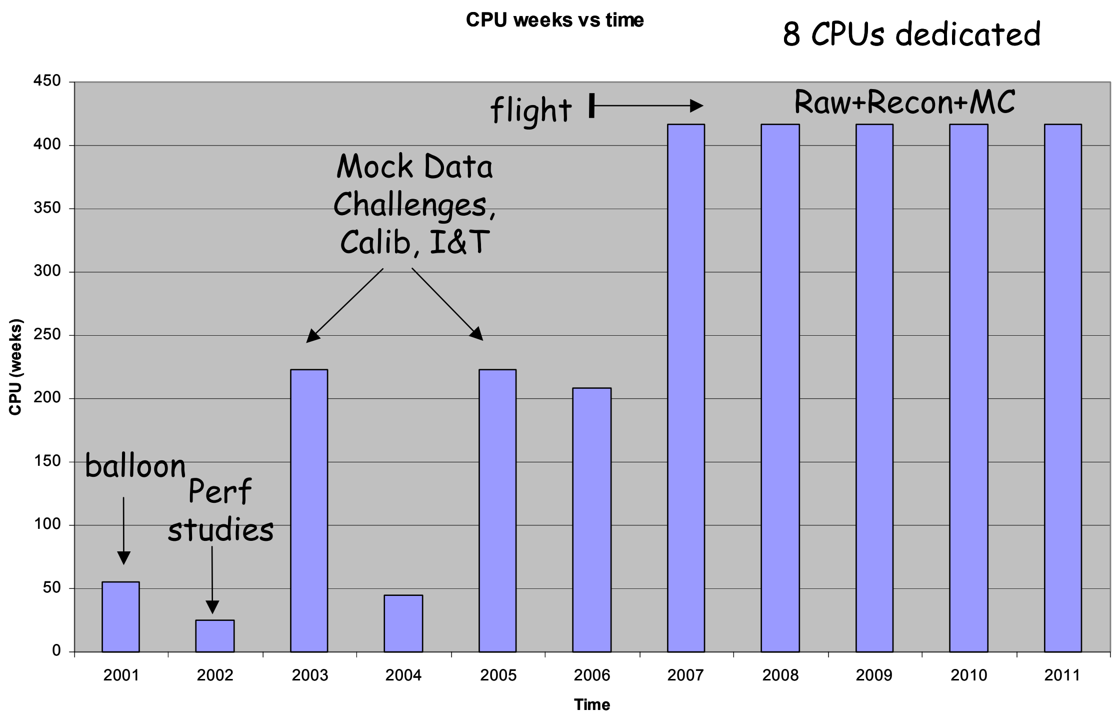


Figure 10.5.2 Projected CPU needs.
The projected needs come to less than 10 processors dedicated at any time, and include MC simulations as well as instrument data processing. Estimates are for current CPU speeds.

These estimates can be inflated by requiring the capacity to reprocess data and generate Monte Carlo simulations concurrently with prompt processing. An estimate of the maximum computation and storage capacity required is a few 10's of processors and 50 TB of disk over the life of the mission. The SLAC Computing Center is committed to supplying, at no explicit expense to GLAST, these disk and CPU resources.

We will need to have a sensible backup scheme for the data and a well designed database which can handle the state of the processing (for all of prompt and reprocessing, and MC generation) and description of the resulting datasets. The database will be the heart of the operation. From it, a fully automated server can completely handle the data processing, with a minimum of human intervention.

10.5.1 Processing Database

The processing database will be used to track the progress of any given dataset - most likely a single file initiated from a Level 0 file - through its life in the system, from arrival at the IOC through to Level 1 processing output. The database is dataset oriented: it must keep track of the properties of the file (location, size, and any needed metadata), as well as its state (completed, pending, failed, etc) and how it got into that state.

Such a database is being prototyped for GLAST use, based upon experience from a similar data pipeline used for the SLD experiment at SLAC. An entity relationship diagram is shown at http://www-glast.slac.stanford.edu/LAT/balloon/data/db2/DB2ERM1.htm. This database is designed for use in the engineering model tests as well as for flight mode. The tables are divided into three categories:

- dataset property description
- processing status
- metadata about the dataset.

In addition, datasets can be grouped, so that like datasets can be easily linked together. An example would be a particular set of Monte Carlo simulations which require many individual files all with the same setup conditions.

10.5.2 Processing Server

It is assumed that the IOC will create entries in the database for new Level 0 datasets. An automated processing server can poll for new entries and take immediate action when it finds them. For event data, we currently envisage a single process acting on an input file with a single output file. It may be conducive for calibration tasks if a separate file of candidate calibration events is created; alternatively, those events could be tagged in the standard output file. Either way will work and is supported by the database. Similarly, the database will support multiple processes operating on a dataset.

The server's life is considerably ameliorated by having all datasets on disk all the time. There is no urgency for backups, and the server does not need to be responsible for doing them

Questions of programming language and database technology are somewhat interconnected. A mainstream interpretive language like Perl is a good match to this kind of work. The database is assumed to be relational, and sql-based. SLAC has an Oracle site-licence, so that seems like a natural choice. Perl has a good interface to Oracle, so that combination of Perl and Oracle will be a good match to the needs.

Since datasets are independent, the server can make use of a load balancing batch system (SLAC uses the LSF batch system) to handle dispatching the processing jobs. So, assuming the daily Level 0 data are broken into a number of smaller files, then the server can submit them to separate processors to achieve parallel throughput. Each process can then communicate its results directly to the database or to the server which would perform the updates.

We will also need web interfaces to the server, both for watching its progress and for interacting with it. These interactions will involve communications in both direction with the server (restarts, etc) and with the database (eg altering the state of a dataset to set an OK flag to resume processing).

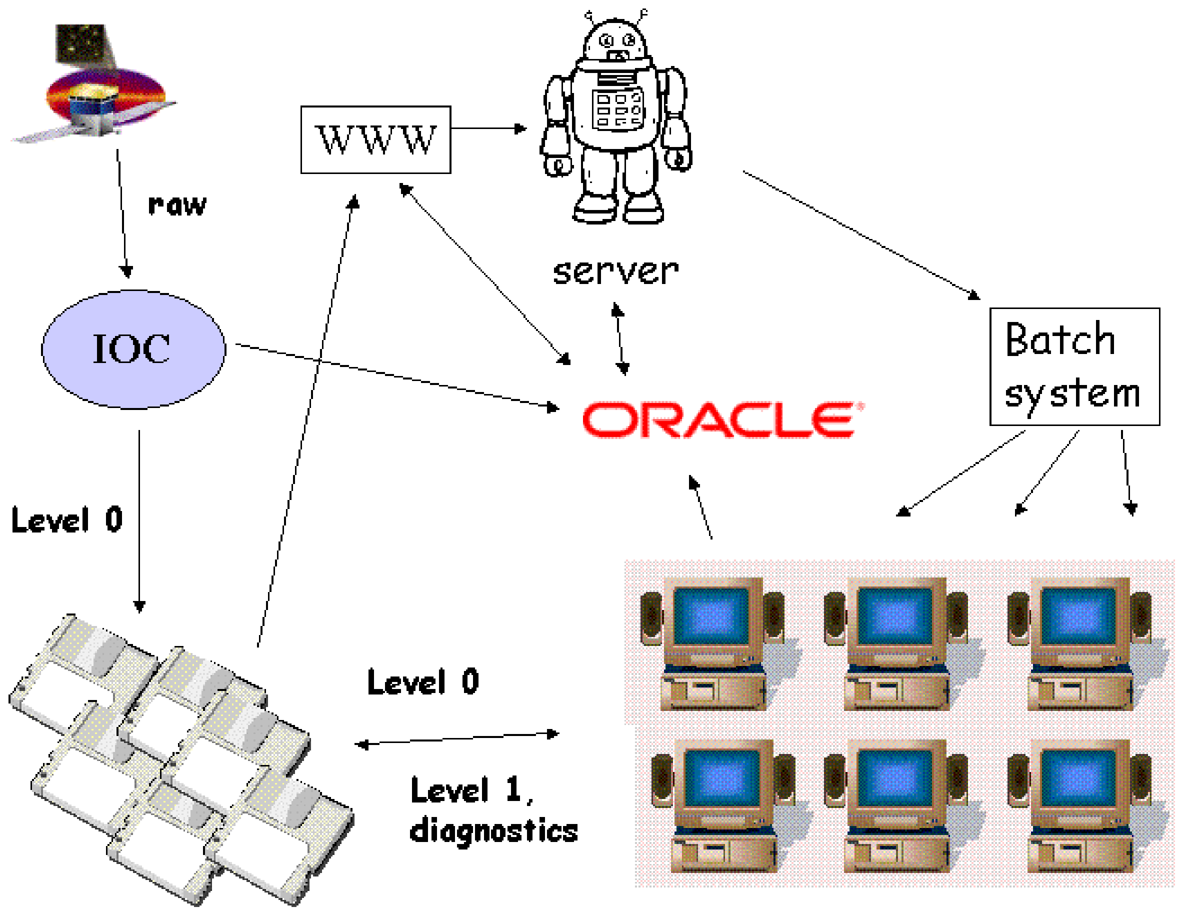


Figure 10.5.3 Automated server.
The server is driven from a central database which contains all information about the datasets that need processing.

10.5.3 Near Real-time Feedback to the IOC

The automated processing provides the opportunity to obtain high level diagnostics from the data and feed that back to the IOC in a timely fashion. Raw data arriving at the IOC can give basic sub-system-centric information on the detector elements, but cannot tie subsystems together, nor give higher level measurements on those subsystems.

Examples are

- measuring ACD tile efficiencies by fitting charged minimum ionizing tracks and extrapolating them to the ACD
- measuring TKR hit efficiencies by looking at the distribution of hits per layer on fitted tracks.
- measuring the amount of the CsI light tapering along the length of the logs, again by extrapolating min-I particles through the calorimeter.

There is a long list of such diagnostics that can be used. These diagnostics will likely take the form of statistics and plots and will be tracked for each dataset, yielding a performance metric as a function of time. These will be compared to standards and as many automated comparisons made as possible. Some of the diagnostics will have to be examined by people, using their expertise to decide whether current distributions are acceptable. All of these diagnostics and standards will be tracked in a database, and viewable over the web.

10.5.4 Calibrations

The DPF will facilitate instrument calibrations by tagging candidate events. An example would be a heavy non-interacting nucleus used for energy calibration of the CAL logs. It will also allow for monitoring of calibrations on a near real-time basis.

10.5.5 Monte Carlo Generation

It is anticipated that large numbers of simulated events will be generated. As much as possible, the machinery that does this generation should make use of the automated server described above, in order to leverage all the benefits of the server and database.

The issue involved is to record the metadata that is unique to MC: the source generator, its parameters, and the configuration and parameters of the simulation package. These are readily handled by the flexible metadata scheme in the database. The code management system also makes code version identification unambiguous.

10.5.6 Data Manager Prototype

Development of a Data Manager (automated server) prototype is currently underway at SLAC. The prototype is intended to demonstrate automation of the data processing steps described above. The prototype and eventually the final product are being developed in Perl, based, as mentioned above, on the availability and reliability of extensively tested libraries providing general network, web, SQL, and Oracle support. These capabilities will be important in providing efficient and convenient access to the data and current processing status.

As well as performing automated processing to Level 1, the data manager, in combination with the database, will provide the logic that allows users to access data sets with similar properties as a group. The data manager will work in tandem with the code management system to provide extensive version information on processing algorithms used for each stage of processing a given data set.

The current version of the prototype is able to generate MC data and run various versions of the reconstruction code on it and will soon have the capability of logging metadata on the simulation and reconstruction specific algorithms to the previously describe Oracle database. A block diagram providing a more detailed view of the interaction of the Data Manager with various SAS components is shown in Fig 10.5.6.1.

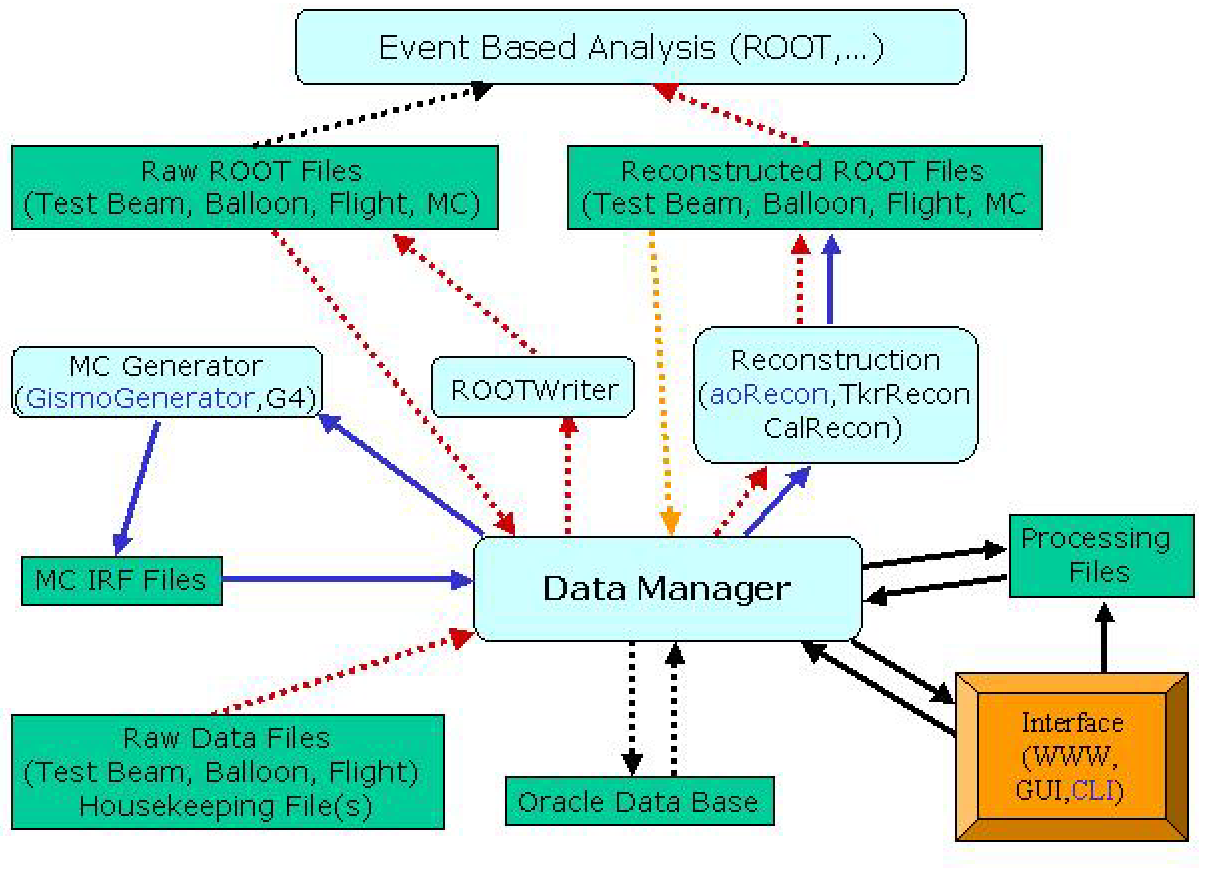

Event Based Analysis (ROOT,...)
Raw ROOT Files (Test Beam, Balloon, Flight, MC)
Reconstructed ROOT Files (Test Beam, Balloon, Flight, MC
MC Generator (GismoGenerator,G4)
ROOTWriter
Reconstruction (aoRecon,TkrRecon CalRecon)
MC IRF Files
Data Manager
Processing Files
Raw Data Files (Test Beam, Balloon, Flight) Housekeeping File(s)
Oracle Data Base
Interface (WWW, GUI,CLI)

Figure 10.5.6.1 The Data Manager relationship to SAS components.
The Data Manager can handle MC generation, data conversion, and full processing through to analyzable output files. Dataset bookkeeping is kept in Oracle.

10.6 Low and High Level Calibration

10.6.1 Introduction

There is no single calibration, nor even much similarity among calibrations, for different subsystems of the instrument. This section is not intended to be a complete blueprint for all calibration designs. Instead, it will concentrate on some of the most important. When sufficient commonality in requirements among different calibrations exists it becomes worthwhile to design common facilities to be used by multiple client calibrations. See the last section of this section for a discussion of data storage and access for calibration results.

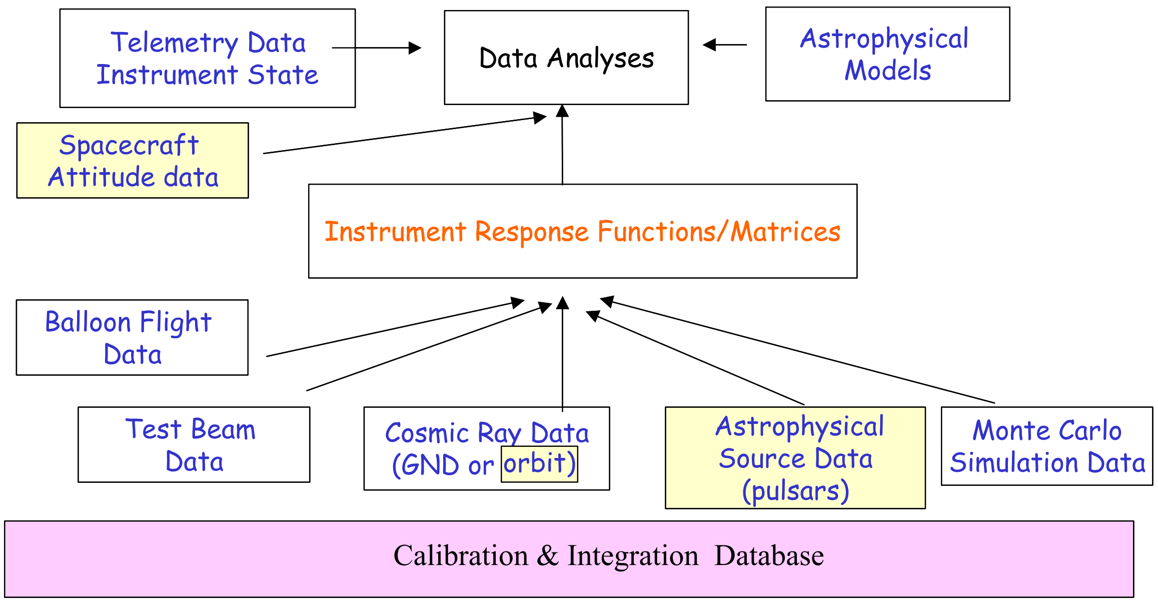

Telemetry Data
Instrument State
Data Analyses
Astrophysical Models
Spacecraft Attitude data
Instrument Response Functions/Matrices
Balloon Flight Data
Test Beam Data
Cosmic Ray Data (GND or orbit)
Astrophysical Source Data (pulsars)
Monte Carlo Simulation Data
Calibration & Integration Database

Figure 10.6.1 Overview of the calibration process.
The IRF is the ultimate goal of the calibration process and takes input from engineering model and construction information as well as flight state and MC simulations. It is all tied together by the calibration and integration database.

10.6.2 Point Spread Function

The Point Spread Function characterizes the angular resolution of the LAT for determination of incident directions of gamma rays.

Determination of the PSF requires the incident photon energy, incident inclination and azimuth; incident vertical position w.r.t to LAT (in the TRK or CAL) and number of tower of conversion; complete description of the state of the LAT, including the states of the towers, hot and dead TKR strips, alignment parameters, diode and phototube gains and pedestals, etc.

10.6.3 Effective Area

Effective Area refers to the effective collecting area for gamma rays after the effects of background rejection/PSF enhancement cuts have been taken into account.

The effective area depends on incident photon energy, incident inclination and azimuth; incident Z position w.r.t to LAT (in the TRK or CAL); number of tower of conversion; background rejection cuts; and PSF rejection cuts. These cuts will depend on types of algorithms (TBD). As for the PSF, the state of the LAT is also required.

10.6.4 Energy Resolution

The resolution is likely best specified as an energy redistribution matrix, mapping true incident energy to the distribution of measured energies.

The energy resolution depends on incident photon energy, incident inclination and azimuth; incident Z position w.r.t to LAT (in the TRK or CAL); number of tower of conversion; pedestals, gains, linearity, light attenuation and rails.

10.6.5 TKR Noisy Strip Identification

The results of this calibration are used by both on-board and on-ground software. On-board, this information provides a way to limit the trigger rate due to hot strips, and to limit the data volume caused by such strips.

On-ground in the reconstruction, the information allows clusters composed entirely of noisy strips to be ignored, and adjacent clusters separated only by noisy strips to be combined, whether or not such strips have been suppressed in the data stream.

Inputs necessary to find these noisy strips can come from any subset of the telemetry data that can be used to provide hits for this measurement. Events with few "real" hits, such as random triggers, are more useful, but this advantage may be outweighed by the limited number of such events available in the telemetry stream.

Lists of hot strips are produced by scoring hits for each strip (884K in the flight instrument), and looking for strips whose hit count is significantly higher than average. The algorithm is straightforward. The limit of the measurement comes from the statistical fluctuation of the hit count, so care must be taken to accumulate sufficient data. In practice, the calibration can be made more sensitive by calculating an "average" using only strips in the vicinity of the strip being examined.

10.6.6 TKR Alignment

For an individual tower, the purpose of the alignment is to decrease the residuals of hits around fitted tracks, with the ultimate goal of reducing the PSF and obtaining accurate absolute directions for gamma rays. For low energy gammas, natural multiple scattering is a much bigger source of error in the measurement, but as the energy gets higher, alignment becomes more and more important.

For the flight instrument, alignment measures the relative orientations of the individual towers, so that data from different towers can be added together without compromising the PSF. It is currently expected that the alignment will remain stable in orbit to better than 10 microns.

Any well-measured track can contribute information for the alignment, but straighter tracks have more statistical weight. Under flight conditions, there will be an abundant source of straight tracks from cosmic ray protons; the number available will be limited only by the efficiency of the on-board trigger, and the constraints of downlink bandwidth.

The misalignment of each element of interest is characterized by a set of parameters. For example, the misalignment of a rigid silicon plane might be specified by 3 translations and 3 rotations. Residuals of hits on tracks to straight lines are parameterized in terms of these parameters, with known coefficients. Some overall measure of quality, for example the total chi-squared of the residuals, is optimized. In practice, the problem reduces to the solution of a very large set of simultaneous linear equations. This set of equations is singular, because there are implicit degrees of freedom in the system. For example, chi-squared is invariant under uniform translations, rotations, and scale changes of the detector.

Mathematically, the solution can be found using the method of singular value decomposition. In practice, much ingenuity is required to render the problem tractable.

This is true for a relatively static detector; in our case, since the primary source of variation is temperature distribution, the detector elements could shift over a fraction of an orbit, and the ultimate resolution may depend on how well we can combine data samples separated in time, but sharing the similar temperature distributions. However, if the requirement of pointing stability of 5 arc-seconds or better is achieved, this complication can be avoided.

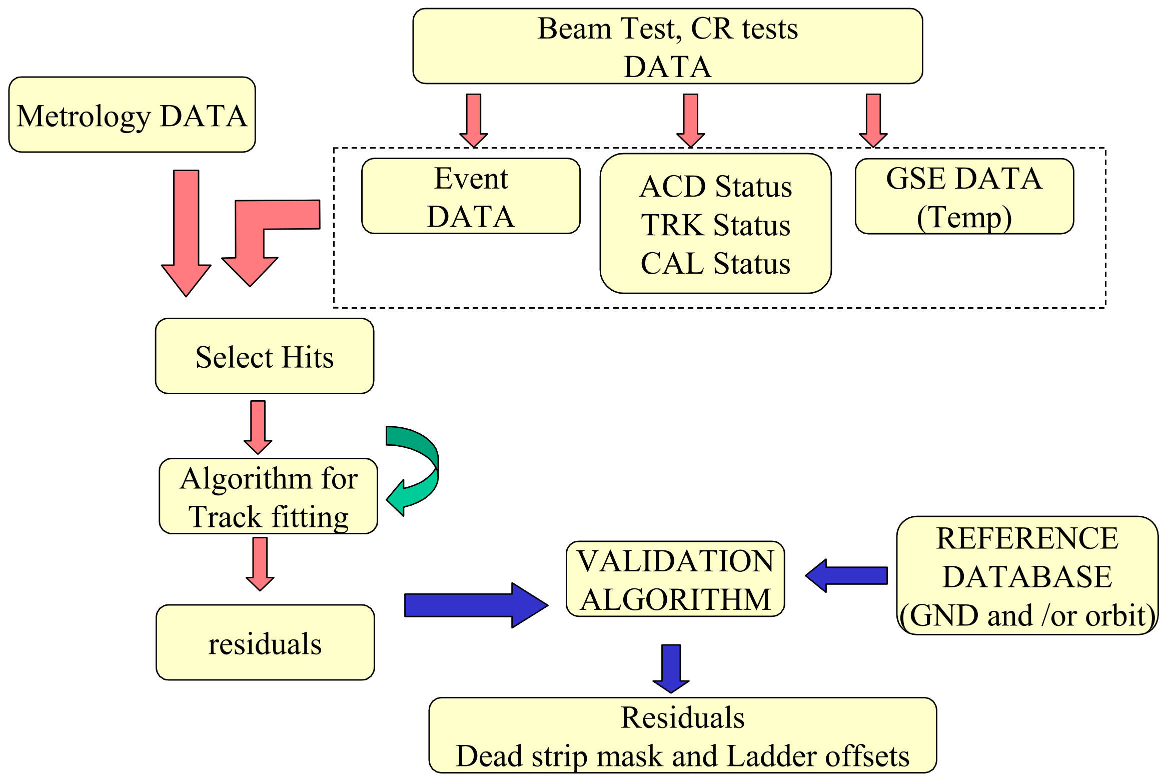

Beam Test, CR tests DATA
Metrology DATA
Event DATA
ACD Status TRK Status CAL Status
GSE DATA (Temp)
Select Hits
Algorithm for Track fitting
residuals
VALIDATION ALGORITHM
REFERENCE DATABASE (GND and /or orbit)
Residuals Dead strip mask and Ladder offsets

Figure 10.6.2 Concept of TKR alignment process.

Fig 10.6.2 shows a concept of the TKR alignment process. Event data from flight or engineering models, and possible external data like optical surveys are used to perform the alignment. The results are validated and checked against a reference database.

10.6.7 CAL Cosmic Ray Calibration

In order for the Calorimeter to measure the energy of incident photons reliably and accurately, an on-orbit calibration of the absolute energy scale of the CAL is required. The high flux of galactic cosmic rays (GCRs) gives a good calibration over most of the full dynamic range of the CAL. A calibration with statistical precision of better than a few percent can be derived each day over essentially the full dynamic range. A detailed discussion is available at http://gamma.nrl.navy.mil/glast/calpdr/technotes/calibrate_on_orbit_fixed_dEdx_.pdf.

Flight software will identify candidate heavy GCRs by their large energy depositions in ACD tiles. These events will be telemetered in when the LAT is in Calibration Mode. The GCR calibration routines shall process Calibration Mode data.

The general scenario is as follows. The ACD will be configured to veto events that deposit some fraction of a MIP to several MIPs. Energy depositions of greater than several MIPs will be flagged. This will flag GCRs from carbon upward in nuclear charge Z, as well as a fraction of Li, Be, and B. The GCRs visible in the GLAST orbit have energies at or above the minimum-ionizing energy and will penetrate the calorimeter (except for highest Z at large angles). For each valid particle, the full LAT will be triggered and the data telemetered in Calibration Mode format. On the ground, trajectories will be precisely determined from the TKR. After correcting for the derived pathlength in each CsI bar, the dE/dx will be accumulated. An adequate calibration can be derived every day from the ~1000 non-interacting CNO-group nuclei that pass through each CsI bar.

Interpretation of the scintillation light measured in the CAL crystals requires understanding of at least two fundamental physical processes: the specific ionization energy loss of heavy cosmic rays ("dE/dx"), and the scintillation efficiency for heavy ions ("dL/dE"). The ionization energy loss of heavy ions is reasonably well understood, and analytic models can be found in the literature (e.g. Ahlen, 1982, Phys. Rev. A, 25, 1856 and references therein). These can be coded in the SAS. Effective, predictive physical models of scintillation efficiency do not exist; however, simple analytic expressions can be derived to describe the dependence of scintillation light output on the charge and energy deposition of the primary particle. The CAL group will measure dL/dE for representative heavy ions in the Engineering Model calorimeter to generate an analytic, phenomenological model, which will be coded in the SAS.

The GCR Calibration Process has the following schematic flow.

1. Extract multi-MIP events from the telemetry stream.
2. Identify likely GCRs and reject obvious junk.
3. Fit GCR track through Tracker, and project track into CAL and ACD.
4. Accept events with clean track through logs. Reject glancing hits or edge events.
5. Identify GCR charge.
6. Identify idenitfy and reject events with charge-changing interactions.
7. Identify and reject events with mass-changing interactions.
8. Fit dE/dx.
9. Iterate steps 5 through 8 until charge identification is stable.
10. Accumulate energy losses and light asymmetries.

10.6.8 Calibration Results Facility

Results from calibration procedures are used to normalize and interpret science data. For most types of calibration these "constants" vary with time (or temperature or...). For such calibrations multiple sets of results must be maintained and must be accessible by specifying cuts on appropriate independent variables.

We expect to perform a survey of recent calibration facilities (SLD and BABAR experiments at SLAC, Chandra's CALDB) to assist in either adopting one or for required functionality.

10.6.8.1 Structure

Services provided may be divided into two categories: query and storage.

10.6.8.2 Query

A relational database management system will keep meta-data records for each collection of calibration data. Such a system provides just the services needed: ability to sort, to select by various criteria, and to link together related records. Each calibration type (e.g., tracker noisy channels) will have a corresponding summary table with one record per calibration of that type. A typical record will include at a minimum

- A unique sequence number.
- Output data format version.
- Calibration procedure version.
- Completion status of the calibration run (OK, ABORTED, etc.)
- Status of calibration results, such as TEST, PRODUCTION, or SUPERSEDED. Only one calibration of a particular type covering a particular time period may be marked PRODUCTION.
- A (logical) pointer to the output data from the calibration. The interpretation of this field could and probably will depend on the calibration type.
- Time when the output data were produced.
- Fields to describe the regime in which the calibration output is valid; for example, start and end times for a time interval of validity.
- A comment field.

Other possible fields, specific to calibration type, include information about input (e.g., time interval over which input data were taken or values of environmental parameters) and calibration-type-specific information about the output (e.g., number of channels calibrated). Indexing can be used to enhance performance for common queries.

10.6.8.3 Storage

Procedures for different calibration types and their output will differ from each other in at least the following ways:

- data volume
- number of data sets (frequency of calibration)
- regularity of output (fixed or variable length, fixed or variable structure)
- environment of clients (hardware and software platforms, interactive or batch, etc.)
- frequency of read access

No single storage mechanism can perform optimally, or even acceptably, under all these conditions. However it should be possible to organize calibration types into a small number of categories such that all types within a group have similar characteristics, and hence can use the same storage mechanism. A modular approach in the design of these mechanisms, separating out data compression algorithms, user interface components, etc., will maximize possibilities for code reuse.

10.6.9 Schedule

A schedule of activities in the calibration area is shown in Fig 10.6.4. This covers engineering model support through to flight operations.

.

When	Where	Module	# Towers	Focus	Does algorithm exist ?	How often?
1999/2000	Beam test	BTEM	1	Do we need metrology ? Are there effects on the PSF ?	Yes	Once
2001	Balloon Flight	BFEM	1	How many triggers are needed ?	Yes	Once
2002	CR tests	EM1,EM2, FMA,FMB	2	Develop automation and database concepts	Yes	TBD
2003-2004	Beam Tests CR tests	Calibration Unit	4	Inter tower alignment	NO	TBD
2005	CR tests	LAT	16	Tune algorithms and database for orbit	NO	TBD
2006-2016	Orbit	LAT	16	Operation mode	NO	TBD

Table 10.6.4 – Schedule for calibration activties

11. Interfaces

The primary interfaces for the SAS are between the IOC and the SSC. The IOC interface delivers data of a known format for the Data Processing Facility to handle, while the SSC interface allows transfer of a variety of data types, plus algorithms. The relations between the IOC, DPF and SSC were shown in Fig 5.1.

11.1 IOC Interface

There are two components to the IOC interface:

- notification of arrival, description, location and format of incoming data
- near real-time diagnostics from Level 1 processing going back to the IOC

11.1.1 Incoming data

The data received from the IOC will be corrected for downlink errors. It will then be made available to the DPF. Our initial concept is that the IOC and SAS will share the database that describes the Level 0 and 1 data. The IOC will update the database as to the availability of new Level 0 data, giving a description of it (command state, etc) and location. From the command state, the data format type can be deduced. The automated server will detect the addition of new data and take the appropriate action on it, updating the database as it goes. It is assumed that the IOC and DPF are co-located, so that the data will be on shared disk.

Consequently, the interfaces required are:

- disk layout
- database layout
- formats of the various command state data paths

Note that these requirements can also be applied to the support of event data taken for Calibration and Integration units.

11.2.2 Diagnostics

The diagnostics will take the form of statistics and plots. These diagnostics will be tracked in the database, and viewable via the web.

The design of the operator interface will depend on directions the IOC takes in its operations software, so not much of this interface can be set yet. The main issue will be how the operators log any variances they see from the DPF diagnostics into their own system.

11.3 SSC interface

The LAT team is required to deliver all mission data to the SSC, both for archiving and in support of the GI program. Because the Event, Event Summary, and Timeline databases will be dynamic and central to the (post-Level 1) analysis system, and because the LAT team is mandated to transfer its high-level analysis environment to the SSC, the interface for high-level data between the SAS and SSC is planned to be via database mirroring. This will facilitate duplication of the high-level analysis environment at the SSC. The same mirroring of databases will also facilitate the planned establishment of additional sites for high-level processing within the LAT team.

The High-Level Calibration database (from which instrument response functions are extracted) and the Diffuse Emission Model database are also central to the high-level analysis environment but they are expected to change infrequently. These need to be provided to the SSC, and the transfer could be implemented as mirrors, but perhaps manual export/import of databases could be used for these.

Other databases used in the LAT analysis environment that should be shared include the Point Source Catalog and the Pulsar Ephemerides. The Point Source Catalog database used by the LAT team may not be the same as that provided to the SSC. At the least, a schedule of release dates for updates will be established. For GI analysis, the Point Source Catalog is better not to be a fluid entity, changing daily. The Pulsar Ephemerides, likely established in collaboration with radio pulsar timing groups, will be needed for barycenter correction of photon arrival times for pulsar studies. The Ephemerides must be current (e.g., to account for timing glitches), but the database will be very small (kbytes) and quite likely just a flat file will suffice.

The lowest-level mission data will likely not be part of standard (post Level-1) analysis, and rarely will be subject to further scrutiny outside of the DPF. The Level 0 data (and housekeeping data) will be delivered to the SSC as flat files.

The specific interfaces between the SAS and SSC for the databases are still being planned. The mechanisms for synchronizing remote databases are being defined; we may borrow from SLAC experience with BABAR and other experiments. In terms of the transfer of the lowest-level data, such as Level 0, the interface is better understood. The Level 0 data will be transferred as it becomes available, on a daily (or 1/2-daily) basis. The SAS will alert the SSC that new files have been staged for transfer. The SSC will transfer and validate the files using pre-computed checksums, then send confirmation to the SAS that the transfer was successful.

12 Quality Assurance

Testing will span the range from

- unit tests - individual component tests
- system tests - collections of units working together, producing statistics and plots representing the performance of entire applications
- instrument performance tests - periodic reassessment of the instrument performance as characterized by the simulation and reconstruction tools
- end-to-end tests - "Mock Data Challenges" which will exercise all the machinery, tools and ability to perform science analysis

12.1 Unit tests

These are stand-alone tests run on the individual code packages. Each package should have a test/ directory with applications having known outcomes. These would be run automatically by the Release Manager (see Sec 10.2) whenever the packages are tagged and when a Release is declared. Failure of the test is reported to the package owner.

12.2 System Tests

These tests are performed when a Release is declared. The entire application is run as a unit and diagnostics generated. See the Release Manager for a fuller description of this process. The diagnostics are generated with releases, tracked from release to release and compared to standards, with variances flagged.

12.3 Instrument Performance Tests

These tests will evaluate on an ongoing basis the basic performance parameters of the instrument and show they meet the relevant LAT Performance Specifications. In particular, we must examine (after all background rejection and resolution cuts)

- the PSF as a function of energy and angle, for front and back sections of the TKR;
- the energy resolution, on-axis and at >60 degrees incidence angle, as a function of energy;
- the effective area as a function of energy and angle (and, hence, the FOV);
- the residual background as a fraction of the accepted high-latitude diffuse flux as a function of energy.

Performing these tests regularly (perhaps annually) and tracking the input and output will allow us to understand the evolution of the code and to verify that no change has taken us far from understood performance. This is really a set of regular tests for the software system itself, as well as a means of obtaining a current understanding of the expected instrument performance.

Such a performance test is underway now for the Instrument PDR, and illustrated by sample plots as shown in Fig 12.1.

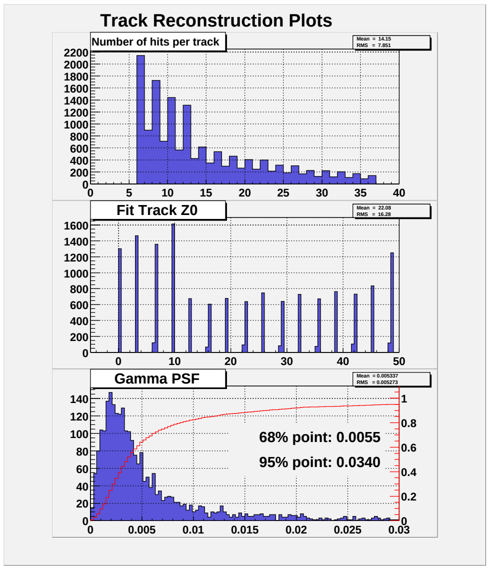


Figure 12.3.1 Sample TKR reconstruction plots.
These show the reconstructed track multiplicity, origin plane of the tracks and the PSF (68% and 95% containment) for gammas.

12.1 End to End Testing - Mock Data Challenges

These are large-scale tests of the entire system: from bulk processing of simulated source raw data, through Level 1 processing followed by Level 2 analysis. In the HEP tradition, the underlying physics put in to the simulations are not revealed to those doing analysis: their job is to find it. The MDC's are large scale efforts involving a good deal of the collaboration, and certainly the LAT Science Working Groups.

It is anticipated that there will be 2-3 such MDCs prior to launch: one in mid 2003 and another toward the end of 2005.

13. Open Issues

13.1 Science Analysis

13.1.1 Event/photon database

There are two major issues involved with the Level 1 data, which will be input to the higher level analyses.

- the form of storage of the data, and how it will perform for the expected types of data requests (temporal and spatial)
- access to this data: is it expected that most of this data will be accessed from a central location, like SLAC, or copied to some or all home institutions
- The disposition of the potentially large volume of MC simulations is still to be determined. MC runs of the simulation and reconstruction will be used extensively to define the high-level calibration of the LAT. MC runs might also be used for specific scientific studies of flight data. The simulations should be preserved but are not likely to be widely accessed.

13.1.2 Event-level analysis platforms

Currently, two analysis packages are being supported: IDL and ROOT. IDL is a commercial package in wide use in the astronomical community; ROOT is a new product out of HEP and is becoming the standard there.

There are three main problems with supporting two packages:

- extra effort in maintaining two systems
- division within the collaboration when developing useful tools and analysis macros: these cannot be directly shared between packages

The factors in favor of two packages are:

- it seems unlikely that either user group will abandon their favored package
- use of ROOT is mushrooming and starting to be noticed in the astrophysics community. It is possible that ROOT will become a standard there by the time of launch and it will have been good for us to have stayed current with it. It is also plausible that ROOT will acquire suitability for Level 2 analysis in the not too distant future as more astronomers get on board and import the functionality needed.

We believe there is really not much choice but to support the two platforms, keeping in mind the downsides.

13.1.3 High Level Analysis environment

The issue is which environment to adopt. At a minimum the environment provides a 'shell' for accessing the data and running the high-level analysis software. It should have GUI and command line interfaces. It should be scriptable and closely coupled with image display and plotting package.

Existing environments under consideration are ROOT and the core of the CIAO (Chandra Interactive Analysis of Observations environment). Both are widely used, although with different constituencies. They are well supported and freely distributable.

A related issue is the communications between the analysis environment and the Analysis Interface layer, which serves data for higher-level analysis. The analysis environment will query the analysis interface for data, exposure, calibration, and interstellar emission model information. The form of what the server returns needs to be established, along with the practical limits of the system in terms of retrieval speed by the Analysis Interface layer and the volume of data transferred.

13.1.4 Representation of instrument response functions

This issue has two aspects. The first is with what detail will we specify them; potentially the effective area A_{eff}, energy resolution, and point-spread function (PSF) could be described as functions of energy, azimuth, inclination, plane of conversion in the TKR or layer of conversion in the CAL, tower of conversion, etc. We need to find out both the practical limit for determining the IRFs from Monte Carlo as well as the point of diminishing returns in terms of science analysis. For example, we may find that no practical scientific gain would be realized by having the instrument response functions defined separately for each tower.

The other aspect to this issue is determining the 'standard' background rejection/PSF enhancement cuts that will be used to select events for high-level analysis. More than one set of cuts will be used, depending on the particular science analysis goal. Probably at a minimum we would have three sets - one useful for GRBs (maximizes A_{eff}, with background rejection and PSF not so important), another for low-latitude point sources (PSF tails minimized, A_{eff} and background rejection not as important), and a third for general analysis (background rejection important, PSF and A_{eff} optimized in a reasonable compromise). For special applications, like very-high energy resolution spectroscopy using wide-angle events in the calorimeter, we may even define additional sets of cuts. We will undoubtedly refine the cuts for each set after launch, but a core set needs to be defined in advance from ground-based calibration and MC simulations. Deriving IRFs for a given set of cuts is a lot of work, and the cuts must be optimized selected carefully. [Where does this go in the schedule? It is an issue that could belong both to instrument simulation and science analysis.]

13.1.5 Implementation of point-source detection/Extended source analysis/spectroscopy

As described in the Science Tools section, the analysis of high-energy gamma-ray astronomy data is fundamentally model fitting, owing to the limited numbers of photons and the limited angular resolution of the measurements. Model fitting can be used to detect point sources or analyze source spectra or extended sources, depending on how the model is defined. The likelihood function, which defines the likelihood of the data given the model, may be used to determine confidence ranges for parameters and to distinguish between different source models. For EGRET and earlier missions in high-energy gamma-ray astronomy, the likelihood function was evaluated by binning the photon data (on the sky and in energy) and comparing the number of gamma rays observed to the number predicted in each bin. The coarser the binning, the less discriminating the likelihood function can be, because in evaluating the predicted numbers of photons the instrument response functions are effectively averaged over the bin. In principle, the unbinned limit (for which the bins are so small that they contain at most 1 photon) maximizes the information usage from the data. In practice, though, unbinned analysis has not been applied extensively because it is more computationally intensive and less stable numerically.

The issue is to decide whether to use unbinned or binned likelihood functions for the routine analysis, and if binned, then what binning. For the LAT, which will be changing its pointing much of the time, binning (in instrument coordinates) must be done very carefully to avoid loss of information from mixing photons with near-axis arrival directions from those far off axis, which generally have less sensitive instrument response. Preliminary indications are that binned analysis can fairly rapidly approach the sensitivity of unbinned analysis if the binning is judicious (e.g., with bins small enough to that the instrument response functions do not vary appreciably within any bin). Ultimately, after the relative performance of the analysis using the two likelihood functions has been established, perhaps both forms of the likelihood function may be implemented, one for speed and the other for maximum sensitivity.

13.1.6 Ground-based alerts for AGN flares and non-triggered GRBs

Another important issue is how we will quickly decide whether a transient (AGN flare or non-triggered GRB) is captured in the most recent data dumps. This is for transients that are not bright enough or brief enough to be noticed onboard. The algorithms to 'trigger' an alert (or follow-up analysis) need to be defined. They may be traditional likelihood analysis as described above, or perhaps something faster will be needed, such as a search for clusters of photons in direction and time or a wavelet filtering of the data to reveal the positions of potential point sources. A related question is how detections in the current sky map are matched against the accumulating point source catalog to decide whether a source is newly detected and/or flaring.

13.1.7 Proprietary Data

During the GI phase of the mission (years 2 and beyond), data awarded to GIs will be proprietary to the GIs for 3 months. During this time the SSC will have to restrict access to these data (primarily by region of the sky and time range). The LAT team will have some ongoing processing rights to the entire dataset (e.g., to search for transients and compile a source catalog. For other uses, though, the LAT team will have to respect the proprietary data rights of the GIs. The open issues regarding this include how the proprietary data protections are implemented. The SSC is nominally responsible for scheduling observations, and may be responsible for tracking data rights. However, this would imply that database mirroring would be two-way (SAS-SSC and SSC-SAS).

Should the Level 0 data be delivered to the SSC by the IOC (where it first arrives at the LAT team) or by the SAS (where it is turned into Level 1 and higher-level data)?

13.2 Support of Engineeering, Calibration Models and Integration Test Units

The LAT has commissioned a study to produce a Calibration and Overall I&T Plan. These will focus on the transition from instrument construction, through test modules and into Integration and then flight. The SAS will be charged with supporting analysis for any event data taken in these activities. The database defined for tracking datasets and the processing server status should serve for all data-acquiring phases.

The issues will be

- to have a database defined which tracks instrumental performance metrics (eg. channels live or dead; gains and pedestals, etc) throughout this entire chain to be available to the final event processing.
- to clearly define what analysis tools are needed for each phase

13.3 User Support

We do not have sufficient manpower at present to maintain adequate user documentation and to answer questions. Answering user queries (especially when documentation is scanty) can pose a serious drain.

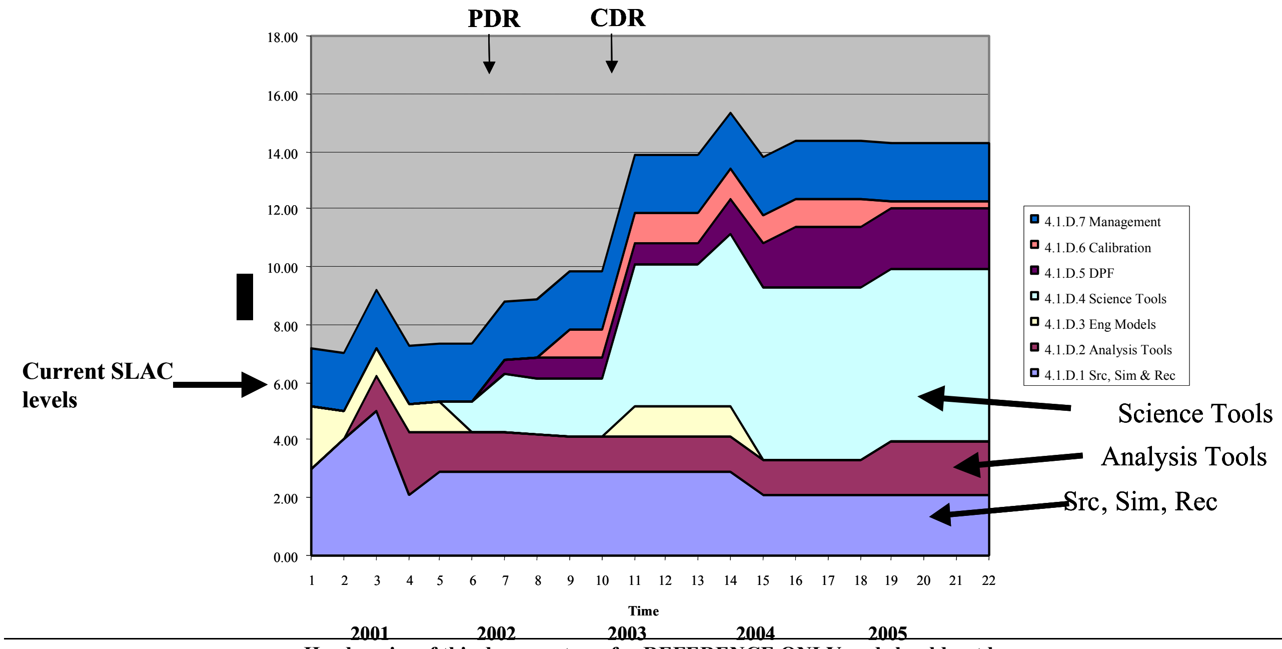
The SLD experiment pioneered a "User Workbook", which was an online tutorial that lead a new user progressively through all the tools and techniques he would need to work the software system. The BABAR experiment followed up on this idea and also created such a workbook. It required one dedicated person for about a year to set up the ideas and recruit a couple of assistants (often graduate students and post-docs) to write the documentation. It requires effort as the system evolves and new features are added or old ones changed, but experience showed it needed perhaps 1/4 FTE after the initial work was complete to keep it up to date. It made a huge difference to the SLD software team, significantly lowering their interrupt rates, and it was much easier for new users to come up to speed.

SLD: http://www-sld.slac.stanford.edu/sldwww/workbook/workbook_prod.html

BABAR: http://www.slac.stanford.edu/BFROOT/www/doc/workbook/workbook.html

13.3.1 Manpower

Eventually some 25 FTEs will be required, with the bulk of the effort going into the Science Tools. Clearly this will involve a build-up of staffing from our current levels. This build-up is indicated in this figure, which shows SLAC + non-assigned effort. SLAC is supplying about 6 FTEs now.



Our top priorities are the simulation/reconstruction and DPF areas. If we do not achieve the necessary manpower levels, our first response will be to approach the collaboration for more manpower from the scientist ranks, and also to negotiate sharing the effort with the SSC scientist. If this fails, we will delay some of the Science Tools, though not the elements that affect sharing data with the SSC. Depending on the severity of the problem, we would delay full implementation of the DPF.