Science Analysis Tools Design

Robert Schaefer – Software Lead, GSSC

Design Talk Outline

- Definition of SAE and system requirements
- Use Cases and Requirements
- Use Case Analysis and Design
- Conclusions

Software Design Architecture Delineation

- Summer 2002 - "Standard Analysis Environment" (SAE) defined, resulting in a clear list of tasks and tools for analyzing LAT gamma ray data.
- September 2002 - LAT SAE description document prepared for September software review:
  - http://www-glast.slac.stanford.edu/sciencetools/reviews/sept02/report/html/review_091602.htm
  - System description
  - System-wide analysis use cases,
  - Basic requirements for the individual tools.
- This document superseded by the Tool Requirements document maintained by David Band:
  - http://glast.gsfc.nasa.gov/ssc/dev/tools_doc/
  - Joanne Bogart analyzed the Use Cases in the September review document and concluded that the analysis tool and database definitions were consistent with an Object Oriented design point of view - she did have some concerns about the utilities definitions.
    - http://www.slac.stanford.edu/~jrb/glast/sciTools-use.shtml

System Wide Requirements

- The GSSC LAT software working group defined system wide requirements:
  - Software Development specification:
    - Core tools in C++
    - Use LAT development environment
    - SLAC cvs repository during development.
  - Software Distribution Requirements
    - Support (at least) Intel Linux and Windows.
    - Software must be free for the user.
    - Minimize number of 3rd party packages
    - No large or complicated 3rd party packages unless necessary (from a cost-benefit analysis point of view) e.g., plplot vs. ROOT for plotting
  - NASA - HEA Requirements
    - Support HEA multi-mission analysis capability effort.
      - Tools will be atomistic (FTOOL - like)
      - Tools will pass parameters in standard FTOOL text format (use PIL)
      - Tools will be able to read and write data products in FITS format
    - Must make source code available for distribution.

Hardcore Software Design

- Next need to proceed with design of software infrastructure.
  - Detailed software requirements needed
  - Identify common classes and interfaces
  - Identify all methods for the common classes.
- Standard software design starts with Use Cases as a way to define requirements. (Note: System use cases were supplied with SAE definition document).
- Why Use Cases?
  - Easy for anyone to write (including non-programmers)
  - Way to discern “real” requirements - if you don’t use it, you can lose it.
  - Form the basis for test cases.
- January 2003 - Tool by tool Use Cases writing effort launched.
- Templates were made for Use Cases (available in LaTEX and MSWord)
- Web site set up for access to use cases converted to HTML.
  - http://glast.gsfc.nasa.gov/ssc/dev/soft_dev/LAT_use_reqs_page.html

Use Case Format (High Level)

Template URL - http://glast.gsfc.nasa.gov/ssc/dev/soft_dev/LATtoolsdev.html

Tool ID. Use Case Name (should start with a verb, e.g., Read Event Data.)

High Level Use Case

Actors: Examples are GI or external component
Goal: What the actor is trying to achieve (a 1 line summary)
Trigger: What starts the use case. (Usually an action by the main actor, but could be some timed event).
Description: Terse paragraph describing what happens. E.g., The program reads the user specified data file. The paragraph should not include details like file format (FITS data file) unless this format has been required externally.
References: list of requirements that this use case addresses based on the numbers of the headings in the requirements document: 1.1, 2.1, etc. If the requirement comes from a different tool or a general requirements document then there will be a prefix for the requirement number which indicates where the requirement comes from. E.g., U14.1.1 means that it is requirement 1.1 form the U14 tool document. (These references will be added when requirements are written; the next step after the high level use cases.)
Use Case Template version: 1.2

Expanded Use Case Format (nitty gritty)

Expanded Use Case

(This is to be done after the High Level Use Cases and rudimentary requirements are written.) I have put the expanded use case template after the high level use case because they should be stored together (on the same page in the document we finally produce).

Preconditions: What must be true in order for the action to begin
Successful End: What happens if the use is a success (e.g., data read into program).
Failed End: What happens if the use case utterly fails and alternative courses do not resolve the problem (e.g., program terminates.)
Priority: How important this use is to the system. (levels 1-3: 1 means critical for proper tool operation, 3 means, just a nice to have feature)

Typical Course of Events

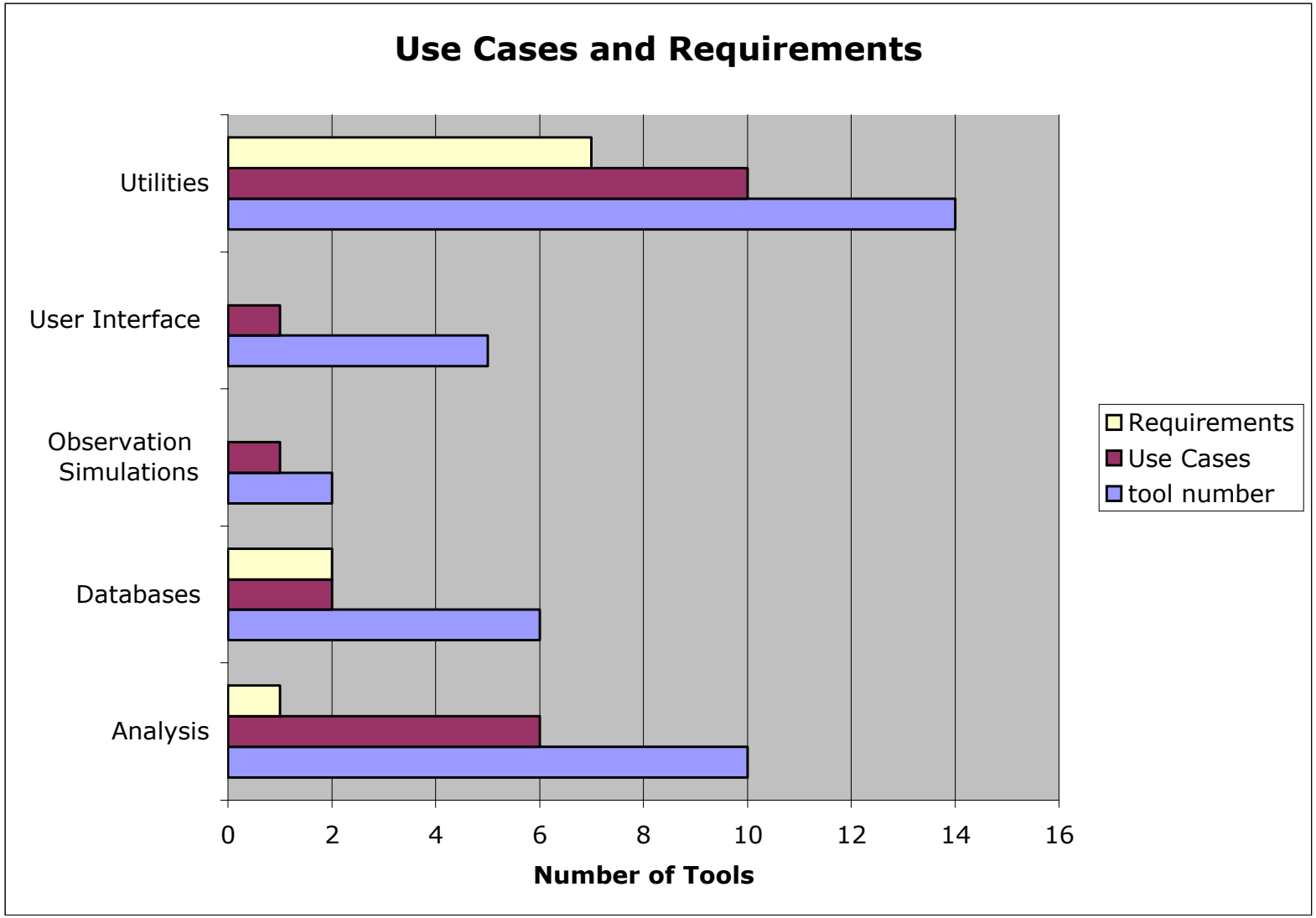
	Actor Action		System Response
1	GI specifies file name on command line	2	read filename from CL
		3	Open file.
		4	read data in file
		A1	When alternate courses are developed, we can expand on the sequence here.,
A2	Alternate course causes actor to do something		
		B1	

Alternative Courses
A. Line #: e.g. error conditions, optional courses, like "Line 3: File not found, prompt user for new file."
B Line #: A separate sequence started by a different alternative course.

Use Case/Requirements Status

- Use cases and requirements have been written for many tools:
  - Use Cases for over half of identified tools have been written
  - Requirements are lagging - we have them for only about 1/4 of tools.

Use Cases by Tool Type



Use Case Analysis

- Collected Use Cases were analyzed
  - Data Objects identified (nouns)
  - Methods identified (verbs)

Object Analysis of Use Cases

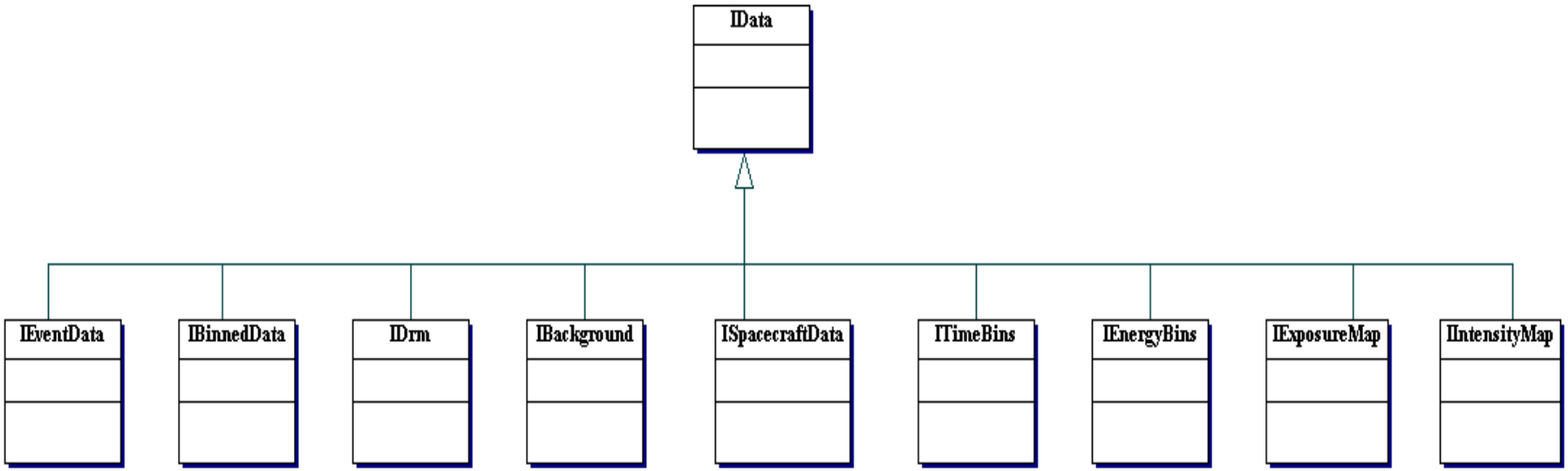
Object	Tool	Functions				
Model Parameters	Likelihood	Calculate				
Region of Interest	Likelihood	Define				
SpaceCraft Data	Likelihood	Read				
	High Level Observation Simulator	Read				
Event Data	Likelihood	Read				
	GRB Event Binning	Read	Bin			
	GRB Rebinning	Read				
	GRB Temporal Analysis	Read				
	GRB Spectral Analysis Tool for Unbinned Energy	Read	Extract			
	Spectral Temporal GRB Physical Modeling	Read				
	Exposure Calculator	Read				
	Map Generator	Read				
	Photon Arrival Time Converter	Read				
	LAT Event Subselection Tool	Read	Extract	Write	Plot	
	High Level Observation Simulator	Read	Create			
	LAT Event Database	Read	Search	Ingest	Send	Write
	LAT Event Database Extractor	Read	Send	Write		
Binned Events Data	GRB Event Binning	Read	Store			
	GRB Rebinning	Read	Store			
	GRB Temporal Analysis	Read				
	GRB DRM Generator Utility	Read	Update			
Exposure Map	Likelihood	Read	Calculate			
	Exposure Calculator	Read	Calculate	Store		
	Map Generator	Read	Calculate	Store		
	GRB Spectral Analysis Tool for Unbinned Energy	Read				
	Spectral Temporal GRB Physical Modeling	Read				
Intensity Map	Map Generator	Read	Calculate	Store		
Model Parameters	Likelihood	Read	Fit	Store		
	Spectral Temporal GRB Physical Modeling	Read	Fit	Store		
Likelihood Surface	Likelihood	Calculate	Plot	Marginalize		
Background Data	GRB Event Binning	Read	Bin	Store		
	GRB Rebinning	Read	Bin	Store		
	GRB Temporal Analysis	Read	Bin	Store		
Time Bins	GRB Event Binning	Read	Create	Store		
	GRB Rebinning	Read	Create	Store		
Energy Bins	GRB Event Binning	Read	Create	Store		
	GRB Rebinning	Read	Create	Store		
	GRB Spectral Analysis Tool for Unbinned Energy	Read	Create	Store		
	Spectral Temporal GRB Physical Modeling	Read	Create	Store		
ARF	GRB Rebinning	Read	Bin	Store		
	GRB DRM Generator Utility	Read	Create	Store		
RMF	GRB DRM Generator Utility	Read	Create	Store		
Temporal Analysis Results (???)	GRB Temporal Analysis	Perform Analysis	Store	Plot		
Spectral Analysis Results (???)	GRB Spectral Analysis Tool for Unbinned Energy	Perform Analysis	Store	Plot		
Flux	Spectral Temporal GRB Physical Modeling	Calculate	Store			
Observed Counts	Spectral Temporal GRB Physical Modeling	Calculate	Store			
Database Query	LAT Event Database	Read				
	Pointing, Livetime and Mode History Database	Read				
	LAT Event Database Extractor	Read				
Pointing, Livetime and Mode History	Pointing, Livetime and Mode History Database	Search	Send	Ingest		
	Exposure Calculator	Read				
	GRB DRM Generator Utility	Read				
Photon Arrival Time	Photon Arrival Time Converter	Convert	Store			
Source Position Coordinates	GRB DRM Generator Utility	Convert				

Other Design Input

- Abstract interfaces are a good design element to keep code separated by functionality.
- Data classes should be independent of the format of the file they are stored in.

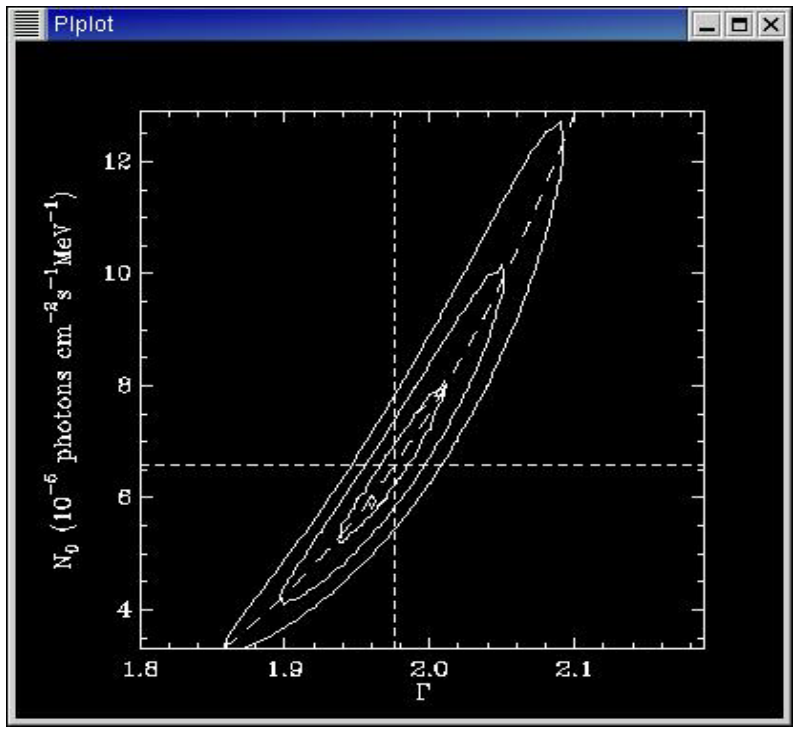
The bulk of defined methods for the objects in the use cases that were written data IO. This resulted in the definition of the GOODI library (See James Peachey's talk)

High Level GOODI Class Diagram



Plotting Library

- Note: For the plotting, a picture = a thousand words, so Jim Chiang created plot/data use cases.
  - http://www.slac.stanford.edu/~jchiang/UI/Plotting/
- 9 different types of plots were identified for the tools plotting interface (see J.P.'s plotting talk)
- E.g.,



Conclusions

- We have come a long way in defining and designing analysis tools - we expect to have common libraries for data classes, file IO, plotting IO, and parameter exchange available well before DC1.
- Time to start developing more detailed use cases and then code for individual tools - Use cases have been a useful way to make progress on the software infrastructure.
- Future: We need the rest of the Use Cases and requirements to:
  - Finish specification and design of common tools
  - Design individual tools - this implies getting into the science algorithms as well (very little of this now). [Use Cases or Science requirements?]
  - Bring the current SAE description-like documnet into a full requirements document
    - We need this for the Ground system reviews.
    - We need this to manage software development
  - Define our test cases to verify our software does what it needs to.

Use Cases -Advantages and Disadvantages

- Advantages
  - Design of a really useful common library is made possible by having a systematic set of use cases and/or requirements documents.
  - Library has been created (see James Peachey's talk to learn more about this library).
  - Provides easy way for defining software requirements
  - Provides List of software test cases
  - Provides at least some design documentation
- Disadvantages
  - Only slightly easier to get people to write than straight requirements documents.
  - No easier to read or update than requirements docs.

Suppl. Use Cases by Development Area

Development Area	Number of Tools	Tools with Use Cases	Tools with Requirements
Analysis	*10*	*6*	*1*
Databases	*6*	*2*	*2*
Observation Simulations	*2*	*1*	*0*
User Interface	*5*	*1 (unofficial)*	*0*
Utilities	*14*	*10*	*7*