Status of the Anaphe Project

ACAT 2002, Moscow, Russia, June 26 2002

Max Sang

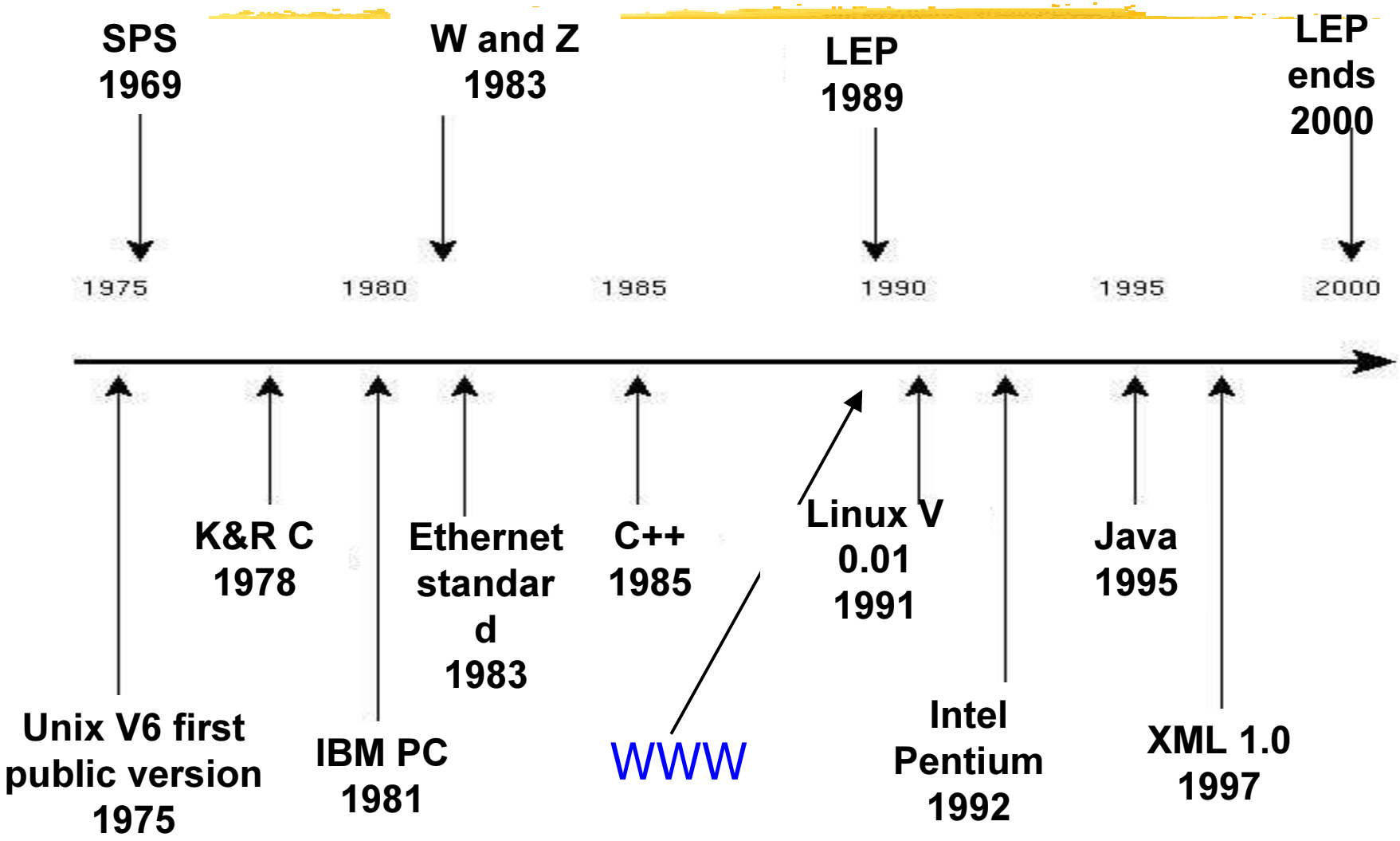
CERN IT/API

max.sang@cern.ch

What is Anaphe?

- An project in CERN IT division to provide a modular OO alternative to CERNLIB
  - Histograms & Ntuples
  - Persistency
  - Plotting and visualisation
  - Fitting and Minimisation
  - Interactive analysis

Prepare for the Long Term!



The CERNLIB Legacy

- Maintenance of CERNLIB still a burden in 2002 (after >20 years). Why?
  - Big scope - (necessarily) home-grown solutions to every problem; graphics, mathematical libraries, data storage...
  - High coupling (despite heroic efforts!) - f77 has few language features to help reduce it
    - Fragility has increased with time
    - Very complex - only real experts can maintain it

1) Scope - does it need to be home grown?

- Example: mathematical library
  - Must be high quality and fast (CPU time = money)
  - CERN IT division shrunk a lot since 70s - who writes it? who maintains it?
- But is it HEP specific?
  - No - and neither are graphics (HIGZ/HPLOT), memory or code management (ZEBRA, PATCHY), etc. etc.

Anaphe Approach I: Scope

- If we can find a stable, high quality open-source product, we use it.
- If we can't, we pay for a commercial solution.
- If it just doesn't exist, we write it.
- → More efficient use of manpower

2) Coupling?

- Fragility (inter-dependency) leads to *much higher* manpower costs over the s/w lifetime - but takes careful design to avoid
- Modern languages help by providing
  - Classes (1980s)
    - help to insulate clients from implementation
  - Interfaces (1990s)
    - protocol "classes" - no concrete types in client code (programming by contract)
    - Run time choice ("component architecture")

Anaphe Approach II: Coupling

- Much work to minimise dependencies
  - Strict partitioning into independent packages which communicate *only* through interfaces.
  - → *component architecture* with NCCD ~ 1 (see L.Tuura, these proceedings.)
  - Normal in 'real world', but less so in HEP. Anaphe team is 3/7 computer scientists → big help (see M.Kasemann, these proceedings).

AIDA (I)

- Abstract Interfaces for Data Analysis (see V.Serbo, these proceedings).
- Protocols defined (so far):
  - fitting
  - plotting
  - histograms and ntuples
  - XML format for exchange of histos/ntuples/...
  - Facade to hide management/persistency

AIDA (II)

- Three AIDA2.2-compliant tools (Anaphe, JAS, OpenScientist)
- Cross-tool (and cross-language!) uniform protocol for data analysis
- Next release 3.x (September 2002)
  - large improvement in functionality and flexibility

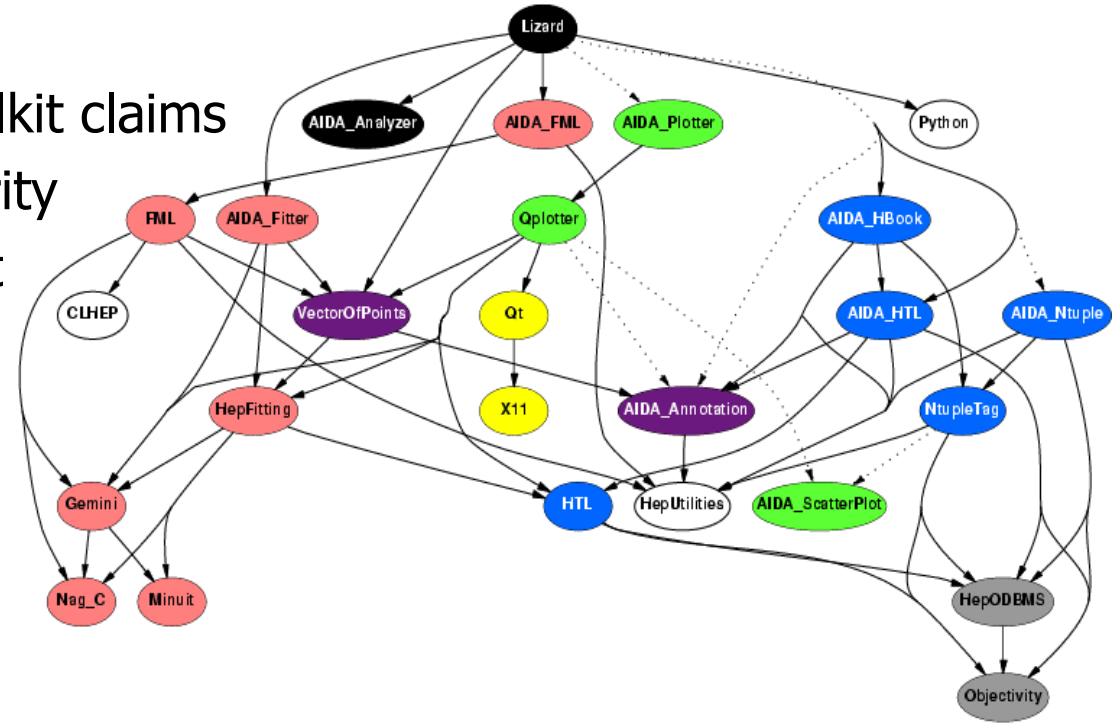
Anaphe Approach III: Layering

- Lizard uses SWIG-generated Python classes which shadow...
- ...the AIDA interfaces which are implemented using...
- ...the Anaphe wrappers which are implemented using...
- ...the Anaphe foundation libraries, which use... CLHEP, Qt, CERNLIB, STL, etc.

Ignominy Analysis of Anaphe

Thanks to Lassi Tuura (CMS)

- Distribution of tools and utilities for LHC era physics
  - Combination of commercial, free and HEP software
  - Claims to be a toolkit
- Seems to live up to its toolkit claims
  - Good work on modularity
  - Clean design is evident in many places
  - Dependency diagrams often split naturally into functional units



Anaphe 3.6.1 Direct Logical Dependencies:
Lizard by Purpose

History

- Anaphe (then LHC++) started 1997
- Foundation libs developed 1997-2000
- Interfaces, Wrappers and Lizard prototyped in time for CHEP 2000
- First production version Summer 2001
- First open-source version Autumn 2001
- AIDA 2.2 compliant version Summer 2002
- AIDA 3.x compliant version Autumn 2002

Core Home-Grown Components

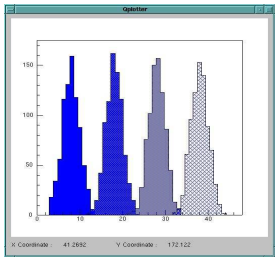
- Histogramming (HTL)
- Plotting (Qplotter)
- Fitting and Minimisation (FML)
- Ntuples (NtupleTag)
- Wrappers implementing the interfaces in terms of other libraries
- Interactive framework (Lizard)

“External” Packages

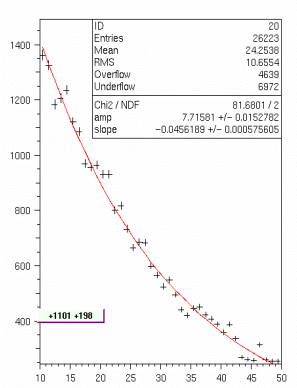
- Commercial licensed packages
  - Objectivity (OO database), NagC
    - deliberately limited dependencies - will be replaced if and when it makes sense
- HEP products
  - HBOOK, MINUIT, CLHEP, HepODBMS
- Open source
  - Qt, OpenInventor, Python, SWIG, expat, Doxygen, ...

Anaphe Components I

- Histogram Template Library (HTL)
  - High performance
  - Flexible treatment of binning, errors
  - Transient and persistent (HepODBMS) versions

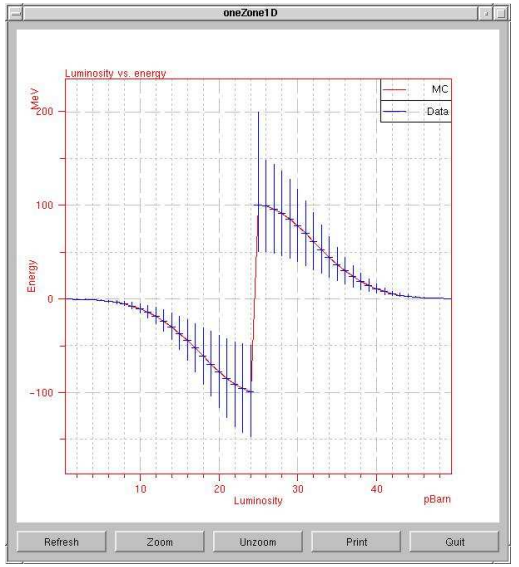


- Fitting and Minimisation Library (FML)
  - Powerful and flexible OO interface
  - uses Gemini (thin wrapper for NAG-C, MINUIT, or other future libraries)



Anaphe Components II

- Qplotter
  - Based on Qt Free 3 - very powerful C++ graphics & GUI library released under GPL
- Several layers of interface complexity
- Stand-alone version available (request from US medical physics group)



Anaphe Components III

- NtupleTag
  - Allows transparent navigation from 'tags' (small number of vars) back to original data
  - Designed for use with OODB
  - Faster - tags better clustered than full data
  - Original version used HepODBMS/Objectivity
    - Read/write of HBOOK RWN with same interface
    - Version based on LCG persistency scheme will be produced (when defined)

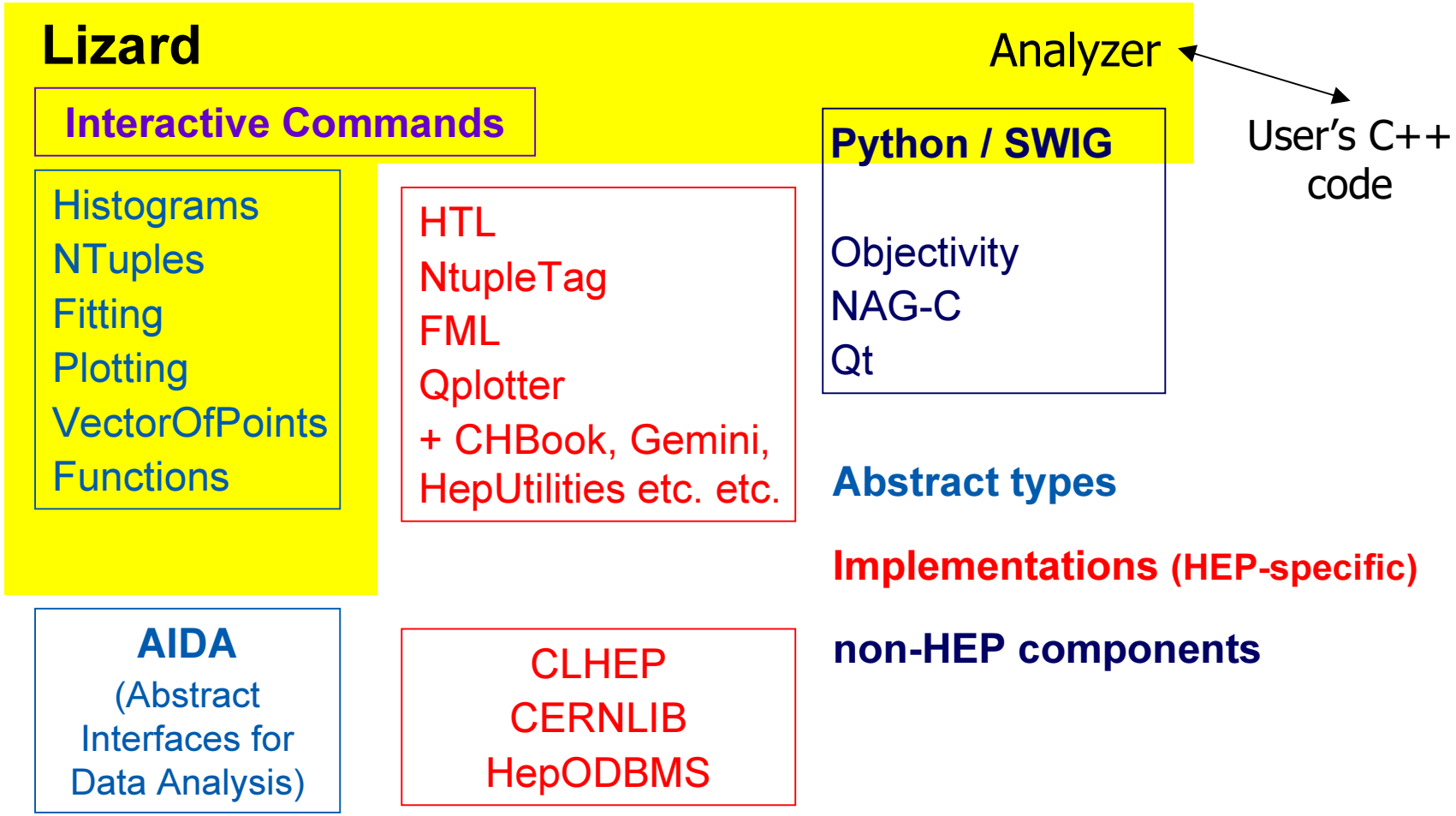
Anaphe Components IV

- AIDA Wrappers
  - Wrapper classes to implement the AIDA interfaces using functionality from the foundation libraries
  - Alternatives available where appropriate - just link to whichever you need
    - Ntuple & Histo : Objectivity or HBOOK
    - Fitter : NAG-C or MINUIT
    - Others as they become available (GSL, LCG persistency solution etc.)

Anaphe Components V: Lizard

- Python framework for interactive analysis
  - All classes and methods from AIDA wrappers mapped into Python commands - plotting, fitting, ntuples, histos.
  - User modules can be plugged in as required
  - Analyzer module provides on-the-fly compilation and running of user's C++ code
  - Lizard is a 'Facade' to the Anaphe libraries
    - Unified interface but only at top level

ANAPHE Components

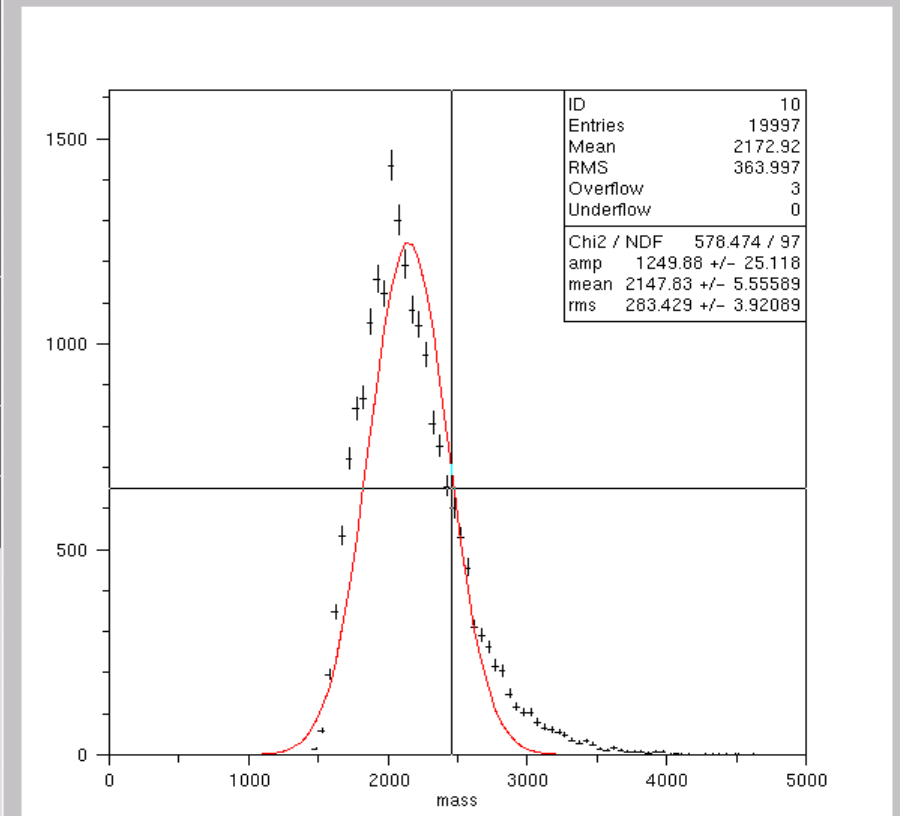


Why Python?

- Object oriented and weakly-typed
  - maps well to OO languages 'underneath', quick and convenient to type
- Easy 'gluing' of components together
  - Mapping from C++ and/or Java is automated (SWIG, Boost, Jython)
- Huge user base
  - low risk of serious bugs (or its disappearance!)
  - lots of free software off the shelf - networking, GUI, OS, scientific etc

(Control Client) | xterm | xterm <2> | xterm <4> | emacs: ntup.py | Qplotter | (xterm <3>)

Qplotter



X Coordinate : 2456.9 Y Coordinate : 648.406

emacs: ntup.py

File Edit Apps Options Buffers Tools

```
ntm.listNtuples()                          # get list of names of all tuples from tupl

nt1=ntm.findNtuple("Charm1")               # retrieve tuple by name
# nt1.listAttributes()                     # prints names and types of attributes

h1=hm.create1D(10,"mass",100,0.,5000.)  # create 1D histo for one of the attribute
h2=hm.create1D(20,"mass for pt1>10",100,0.,5000.)

# for the updating plot, reset the plotter to a single zone
pl.zone(1,1)
h1.reset()
start = 0
num = 1000
for i in range(0.,20.):
    nt1.project1D(h1,"MASS","",start,num)
    vTemp=vm.from1D(h1)
    pl.plot(vTemp)
    del vTemp
    start=start+num

# that's it !
```

d classes initialised

```
:-) exe("ntup.py")

Explorables present:

        Charm1

:-) err()
:-) h1=hm.retrieveHisto1D(10)
:-) vFit = hfit(h1,"G")
+++Some of the bins have zero errors. They will be ignored in the chi-square fit.
+++Use setDefaultError(double) to redefine zero errors.

+++ Performing a chi-square fit.
:-)
```

One | Two | Three | Four | Five | Six | 15:05

News I:
AIDA Compliance

- Refactoring of wrappers to become AIDA compliant - requires change of user interfaces at Lizard level
  - Required no changes to foundation libs
  - Shows flexibility of layering
  - New documentation July 2002
  - Future major releases will be synchronized with AIDA releases
  - Next major release September 2002 (AIDA 3)

News II: Internal Development

- Response to user feedback and (mostly) friendly criticism
  - Improvements in testing and installation
  - Clearer and more informative web site (available July 2002)
  - Lots of work on documentation (examples, tutorials, reference docs) and procedures for responding quickly to user requests
  - We hope you see a difference!

Medium Term Plans

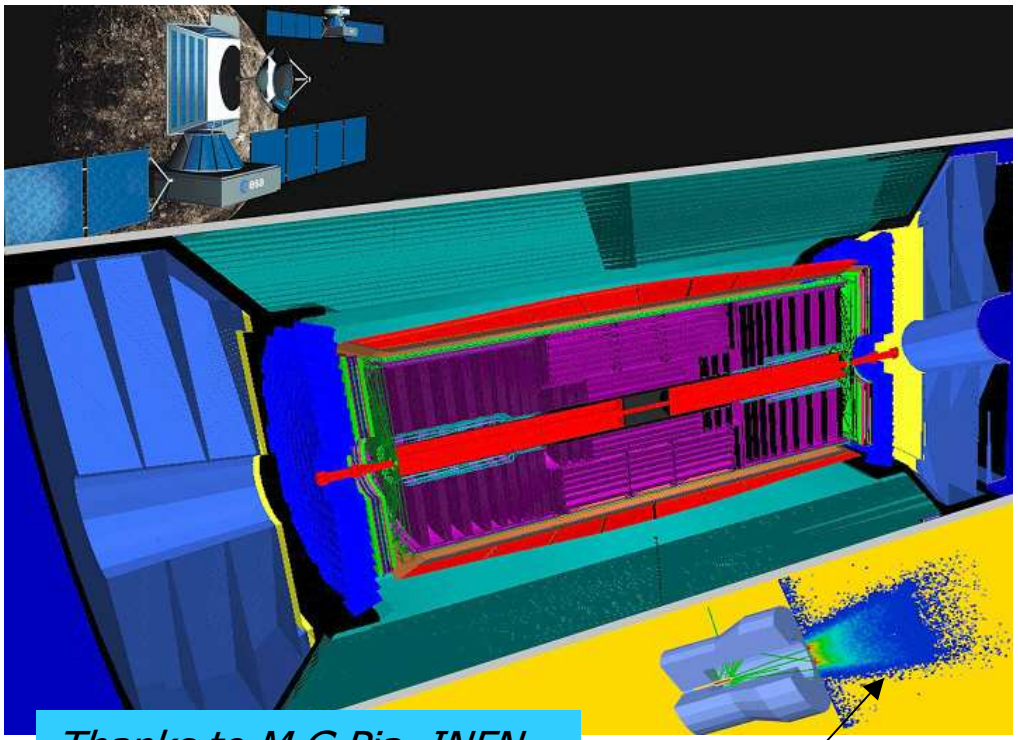
- AIDA 3.x compliant version (September)
  - Exposes much more functionality of underlying libraries
  - Finer-grained control of plotting, fitting etc at user (Lizard) level
- Reading of Root (3.x) files
- HBOOK column-wise ntuples
- Improvements to foundation libs (ask us!)

Users I

- User community small but growing
- Some pioneers on CMS & LHCb
- Collaboration with JAS & OpenScientist growing - component sharing is now a real possibility
- Geant 4 has adopted AIDA as a tool-independent analysis standard so some Geant 4 users are coming to Anaphe

Users II

- Data analysis of medical and space physics simulations using Geant 4 - see CERN Courier, June 2002

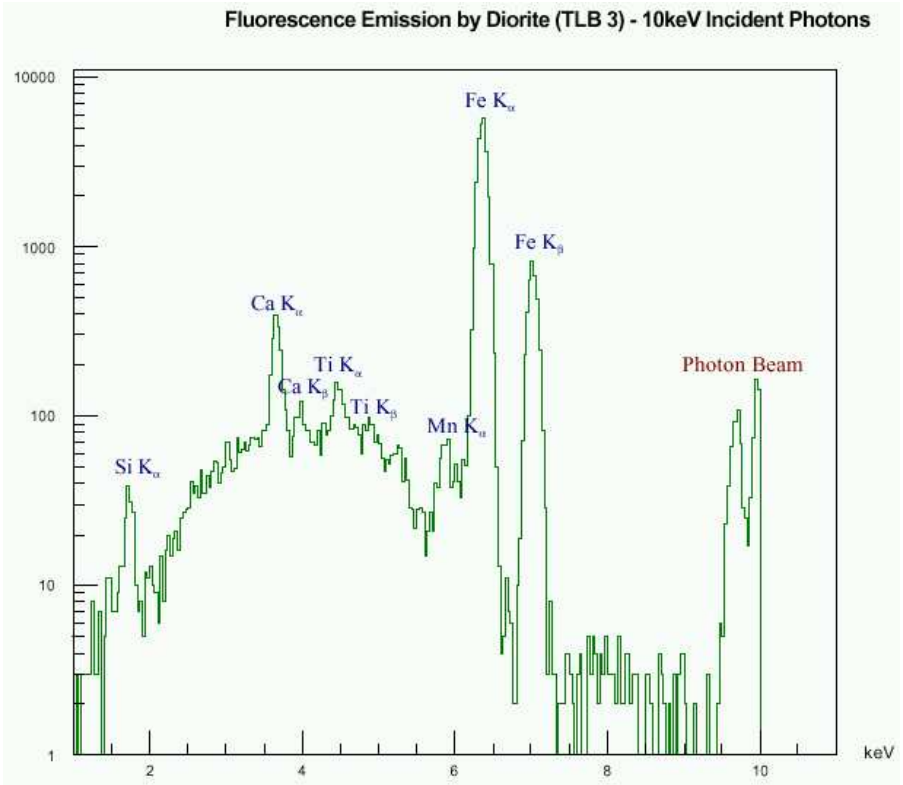


Thanks to M.G.Pia, INFN

Anaphe

Users III

- Planetary physics (X-ray fluorescence of simulated Mars soil)
- UCL group on LISA (space gravitational wave detector)



Thanks to A.Manero, INFN

Distributed Ntuple Analysis

- Would like to do distributed parallel analysis over large data sets with minimal assistance from the end user
  - e.g., distributed, parallelised ntuple analysis and projection into Histograms in Lizard *Analyzer*
  - Abstract Interface hides the complexity, no change in tool(s) or user code needed
- First prototype for very general distributed parallel data analysis available (see J.Mosciki, these proceedings). Special case (ntuple analysis) already developed using Lizard and the Anaphe libraries

Summary

- Anaphe is a layered set of loosely coupled C++ components for data analysis, plus an interactive Python framework (Lizard)
- Only HEP-specific parts written in-house
- Developed and maintained by CERN IT
- Committed to AIDA compliance
- Functionality, stability and user support improving rapidly; gaining users (you..?)

More Information

- http://cern.ch/Anaphe
- http://aida.freehep.org

- **HepLib.Support@cern.ch**